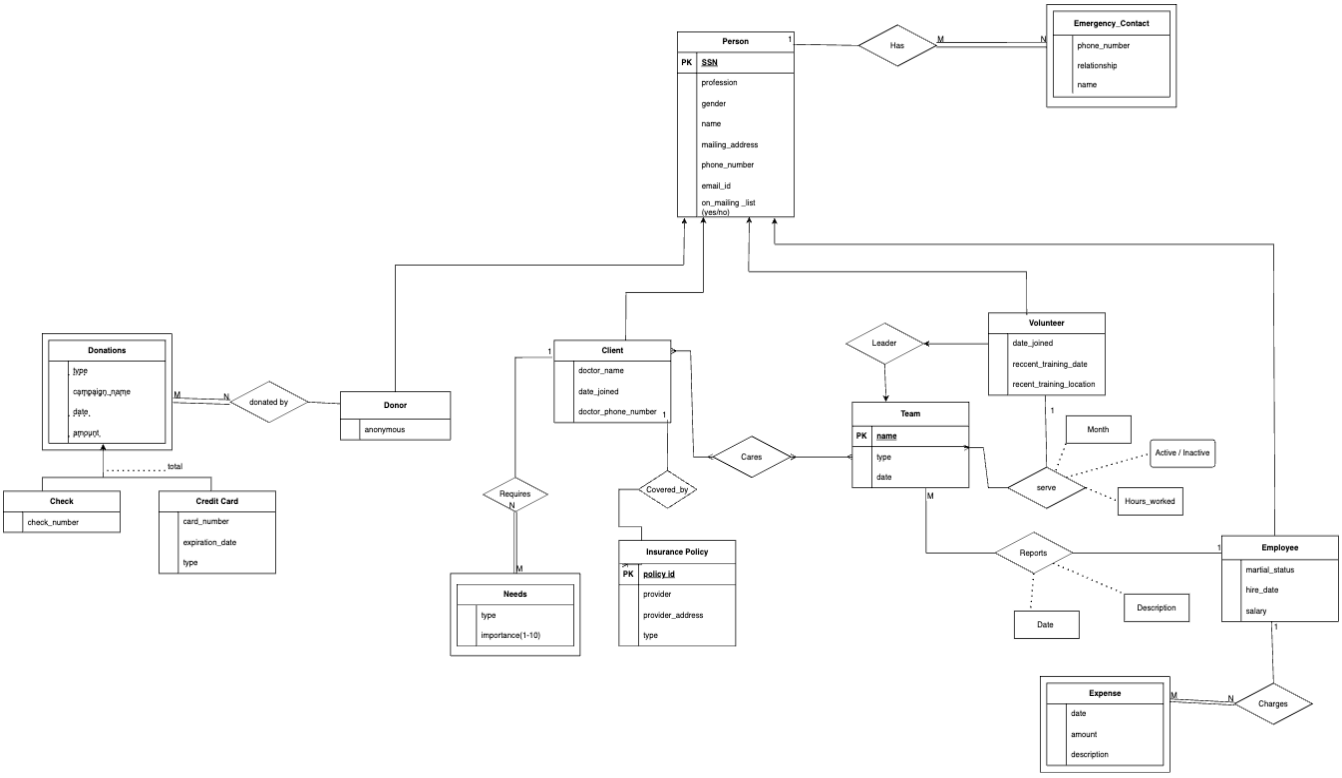
Course name	**:**	**Database Management Systems**
Course Number	**:**	**DSA 4513-001**
Section Number	**:**	**001**
Semester and Year	**:**	**Fall 2024**
Instructor's Name	**:**	**Dr. Le Gruenwald**
Student Name	**:**	**L Srihari Boppana**
Student Id	**:**	**113638264**
Email address	**:**	**l.srihari.boppana-1@ou.edu**
Title	**:**	**PAN Database System**

TASK 1: ER Diagram



TASK-2: Relational Database Schema

- **Person** (**SSN** , profession, gender, name, mailing_address, phone_number, email_id, on_mailing_list)
- **Emergency_Contact** (name , phone_number, relationship, SSN)
- **Client** (SSN , doctor_name, date_joined, doctor_phone_number)
- **Needs** (type , importance, client_SSN)
- **Insurance_Policy** (**policy_id**, provider_name, provider_address, type)
- **Covered_By(**client_SSN, **policy_id)**
- **Team** (**Name** , type, date)
- **Volunteer** (**SSN** , date_joined, recent_training_date, recent_training_location)
- **Leader**(**volunteer_SSN**, Name)
- **Serves** (**Volunteer_SSN, Team_Name** , active / inactive , hours _worked,month)
- **Cares**(Name,client_SSN)
- **Employee** (**SSN** , marital _status, hire_date, salary)
- **Reports** (**employee_SSN, team_name, date** , description)
- **Expense** (employee_SSN,date, amount, description)
- **Donor** (**SSN** , anonymous)
- **Donations** (Donor_SSN, type, campaign _name, date, amount)
- **Check** (Donor_SSN, date, check _number , campaign _name, type, amount)
- **Credit_Card** (Donor_SSN, date, card_number ,card_type, Amount, type, campaign _name,expiration_date,)

TASK 3: 3.1 Discussion of storage structures for tables

Table Name	Queries and Types	Search Key	Query Frequency	Selected File Organization	Justifications
Person	#2 - Insertion, #3 - Insertion, #12 - Random Search	SSN	1/week, 2/month, 1/week	**Dynamic Hashing with search key SSN**	Dynamic hashing enables quick access and efficient handling of high insertions and updates, making it ideal for records with frequent changes.
Emergency_Contact	#12 - Random Search	SSN	1/week	**Dynamic Hashing with search key SSN**	Dynamic hashing is well-suited for emergency contact retrievals associated with SSN, providing fast lookups when needed.
Client	#2 - Insertion, #8 - Random Search, #10 - Random Search, #15 - Deletion	SSN	1/week, 4/year, 4/year, 4/year	**Dynamic Hashing with search key SSN**	This method is efficient for managing frequent insertions, deletions, and quick retrievals based on SSN.
Needs	#15 - Range Search	Importance Score	4/year	**B+ Tree with search key Importance Score**	B+ Tree structure allows efficient range queries on static data, making it optimal for accessing ordered data based on importance scores.
Insurance_Policy	#15 - Random Search	Type	4/year	**B+ Tree with search key Type**	B+ Tree indexing offers efficient searches for mostly static data by type, making it suitable for infrequent lookups.
Covered_By	#15 - Random Search	Policy ID	4/year	**B+ Tree with search key Policy ID**	B+ Tree indexing supports efficient access to insurance data, useful for searches on unique policy identifiers.

Volunteer	#3 - Insertion, #4 - Insertion, #10 - Random Search	SSN	2/month, 30/month, 4/year	**Heap File Organization**	Heap file organization is effective for frequent insertions with minimal need for ordered access, ideal for volunteer records.
Team	#1 - Insertion, #11 - Range Search	Date Formed	1/month, 1/month	**B+ Tree with search key Date Formed**	Sequential organization is ideal for static data, where occasional range-based access by formation date is needed.
Serves	#3 - Insertion, #4 - Insertion, #10 - Random Search	Team Name	2/month, 30/month, 4/year	**B+ Tree with search key Team Name**	B+ Tree indexing provides efficient insertion and random search capability, especially useful for team-related queries.
Leader	#5 - Insertion	Name	Infrequent	**Heap File Organization**	Heap file is effective for infrequent insertions, allowing efficient handling of unordered access for leader data.
Cares	#2 - Insertion, #10 - Random Search	Client SSN	1/week, 4/year	**Dynamic Hashing with search key Client SSN**	Dynamic hashing is efficient for frequent updates and quick retrievals, ideal for managing client-to-care associations.
Employee	#5 - Insertion, #6 - Random Search, #9 - Range Search, #13 - Random Search, #14 - Update	SSN	1/year, 1/day, 1/month, 1/week, Infrequent	**B+ Tree with search key Employee SSN**	B+ Tree structure supports efficient aggregation, retrieval, and updates on employee records, making it optimal for SSN-based access.
Reports	#5 - Insertion, #14 - Random Search	Employee SSN	1/year, Infrequent	**Dynamic Hashing with search key**	Dynamic hashing provides quick access to reports linked to employee SSNs, ideal for

				Employee SSN	occasional updates and searches.
Expense	#6 - Insertion, #9 - Range Search	Expense Date	1/day, 1/month	**B+ Tree with search key Expense Date**	B+ Tree indexing supports efficient date-based retrievals, particularly useful for range searches on expense data.
Donor	#7 - Insertion, #13 - Random Search	Donor SSN	1/day, 1/week	**Dynamic Hashing with search key Donor SSN**	Supports frequent donor record insertions and retrievals without requiring ordered access, making it efficient for SSN-based lookups.
Check	#7 - Insertion (Donation), #13 - Random Search	Donor SSN	1/day, 1/week	**Dynamic Hashing with search key Donor SSN**	Optimized for frequent insertions of check donations with quick retrievals based on donor SSN.
Credit_Card	#7 - Insertion (Donation), #13 - Random Search	Donor SSN	1/day, 1/week	**B+ Tree with search key Donor SSN**	B+ Tree indexing enables ordered access and efficient retrievals for credit card donation records based on SSN.

3.2 Discussion of storage structures for tables (Azure SQL Database)

In Azure SQL Database, a clustered index is automatically created on primary key columns. This clustered index optimizes data retrieval by storing rows in the sorted order of the primary key, ensuring efficient data access. Given that only one clustered index can be created per table, additional optimization can be achieved by implementing non-clustered indexes on frequently queried columns. Here are the storage structure choices I have decided on for a few tables:

- **Team Table -**

  Index Key: Date

- **Volunteer Table -**

  Index Key : RecentTrainingDate

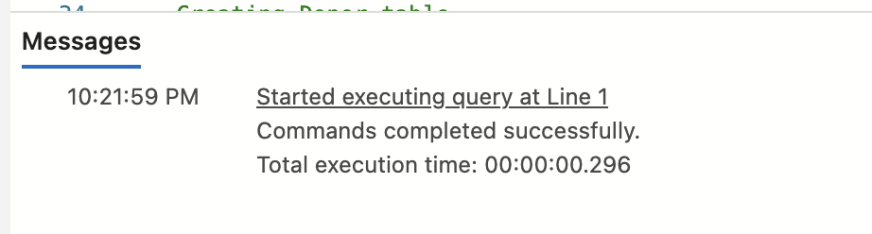
- **Serves Table -**

  Index Key: Month

Task 4 - SQL statements and screenshots showing the creation of tables in Azure SQL Database-

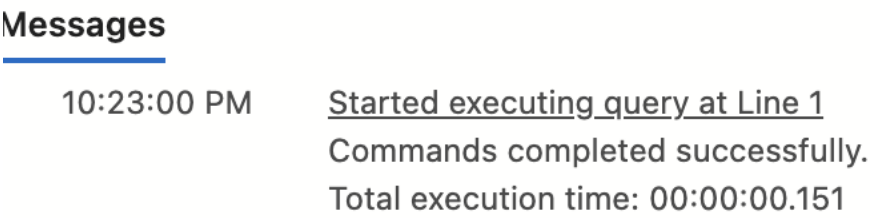
1)Create Person Table:

```
22   -- Creating Person table
23   CREATE TABLE Person (
24       SSN CHAR(9) PRIMARY KEY,
25       profession VARCHAR(50),
26       gender CHAR(1),
27       name VARCHAR(100),
28       mailing_address VARCHAR(255),
29       phone_number VARCHAR(15),
30       email_id VARCHAR(100),
31       on_mailing_list BIT
32   );
33
```



2)Create Donor Table:

```
33
34   -- Creating Donor table
35   CREATE TABLE Donor (
36       SSN CHAR(9) PRIMARY KEY,
37       anonymous BIT,
38       FOREIGN KEY (SSN) REFERENCES Person(SSN)
39   );
40
```



3) Create Emergency_contact Table

```
40
41    -- Creating Emergency_Contact table
42    CREATE TABLE Emergency_Contact (
43        name VARCHAR(100),
44        phone_number VARCHAR(15),
45        relationship VARCHAR(50),
46        SSN CHAR(9),
47        FOREIGN KEY (SSN) REFERENCES Person(SSN)
48    );
49
```

Messages

10:23:49 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.143

4) Create Client Table

```
49    -- Creating Client table
50    CREATE TABLE Client (
51        SSN CHAR(9) PRIMARY KEY,
52        doctor_name VARCHAR(100),
53        date_joined DATE,
54        doctor_phone_number VARCHAR(15),
55        FOREIGN KEY (SSN) REFERENCES Person(SSN)
56    );
57
58    -- Creating index on SSN in Client table
59    CREATE NONCLUSTERED INDEX idx_client_ssn ON Client(SSN);
60
```

Messages

11:28:10 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.767

5) Create Needs Table

```sql
-- Creating Needs table
CREATE TABLE Needs (
    type VARCHAR(50),
    importance INT,
    client_SSN CHAR(9),
    FOREIGN KEY (client_SSN) REFERENCES Client(SSN),
    PRIMARY KEY (client_SSN, type)
);

-- Creating index on importance in Needs table
CREATE NONCLUSTERED INDEX idx_needs_importance ON Needs(importance);

```

Messages

11:30:00 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.186

6) Create Insurance_Policy Table

```sql
-- Creating Insurance_Policy table
CREATE TABLE Insurance_Policy (
    policy_id CHAR(10) PRIMARY KEY,
    provider_name VARCHAR(100),
    provider_address VARCHAR(255),
    type VARCHAR(50)
);

```

Messages

11:31:06 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.142

7) Create Covered_By Table

```
81  -- Creating Covered_By table
82  CREATE TABLE Covered_By (
83      client_SSN CHAR(9),
84      policy_id CHAR(10),
85      PRIMARY KEY (client_SSN, policy_id),
86      FOREIGN KEY (client_SSN) REFERENCES Client(SSN),
87      FOREIGN KEY (policy_id) REFERENCES Insurance_Policy(policy_id)
88  );
89
90  -- Creating index on policy_id in Covered_By table
91  CREATE NONCLUSTERED INDEX idx_covered_by_policy_id ON Covered_By(policy_id);
```

Messages

11:31:45 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.129

8) Create Team Table

```
93  -- Creating Team table
94  CREATE TABLE Team (
95      Name VARCHAR(100) PRIMARY KEY,
96      type VARCHAR(50),
97      date DATE
98  );
99
100 -- Creating index on date in Team table
101 CREATE NONCLUSTERED INDEX idx_team_date ON Team(date);
```

Messages

11:32:31 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.163

9) Create Volunteer Table

```
103    -- Creating Volunteer table
104    CREATE TABLE Volunteer (
105        SSN CHAR(9) PRIMARY KEY,
106        date_joined DATE,
107        recent_training_date DATE,
108        recent_training_location VARCHAR(100),
109        FOREIGN KEY (SSN) REFERENCES Person(SSN)
110    );
111
112    -- Creating index on SSN in Volunteer table
113    CREATE NONCLUSTERED INDEX idx_volunteer_ssn ON Volunteer(SSN);
114
```

Messages

11:33:21 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.179

10) Create Leader Table

```
115    -- Creating Leader table
116    CREATE TABLE Leader (
117        volunteer_SSN CHAR(9),
118        Name VARCHAR(100),
119        PRIMARY KEY (volunteer_SSN, Name),
120        FOREIGN KEY (volunteer_SSN) REFERENCES Volunteer(SSN),
121        FOREIGN KEY (Name) REFERENCES Team(Name)
122    );
```

Messages

11:33:21 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.179

11) Create Serves table with month as part of the primary key:

```
124   -- Creating Serves table with month as part of the primary key
125   CREATE TABLE Serves (
126       Volunteer_SSN CHAR(9),
127       Team_Name VARCHAR(100),
128       active_inactive CHAR(1),
129       hours_worked INT,
130       month INT CHECK (month BETWEEN 1 AND 12),
131       PRIMARY KEY (Volunteer_SSN, Team_Name, month),
132       FOREIGN KEY (Volunteer_SSN) REFERENCES Volunteer(SSN),
133       FOREIGN KEY (Team_Name) REFERENCES Team(Name)
134   );
```

Messages

1:43:17 AM Started executing query at Line 10
Commands completed successfully.
Total execution time: 00:00:00.046

12) Create Cares Table:

```
138   -- Creating Cares table
139   CREATE TABLE Cares (
140       Name VARCHAR(100),
141       client_SSN CHAR(9),
142       PRIMARY KEY (Name, client_SSN),
143       FOREIGN KEY (Name) REFERENCES Team(Name),
144       FOREIGN KEY (client_SSN) REFERENCES Client(SSN)
145   );
146
147   -- Creating index on client_SSN in Cares table
148   CREATE NONCLUSTERED INDEX idx_cares_client_ssn ON Cares(client_SSN);
```

Messages

11:36:25 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.200

13) Create Employee Table:

```
150    -- Creating Employee table
151    CREATE TABLE Employee (
152        SSN CHAR(9) PRIMARY KEY,
153        marital_status VARCHAR(10),
154        hire_date DATE,
155        salary DECIMAL(10, 2),
156        FOREIGN KEY (SSN) REFERENCES Person(SSN)
157    );
158
159    -- Creating index on SSN in Employee table
160    CREATE NONCLUSTERED INDEX idx_employee_ssn ON Employee(SSN);
161
```

Messages

11:38:07 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.344

14) Create Reports Table:

```
162    -- Creating Reports table
163    CREATE TABLE Reports (
164        employee_SSN CHAR(9),
165        team_name VARCHAR(100),
166        date DATE,
167        description TEXT,
168        PRIMARY KEY (employee_SSN, team_name, date),
169        FOREIGN KEY (employee_SSN) REFERENCES Employee(SSN),
170        FOREIGN KEY (team_name) REFERENCES Team(Name)
171    );
172
173    -- Creating index on employee_SSN in Reports table
174    CREATE NONCLUSTERED INDEX idx_reports_employee_ssn ON Reports(employee_SSN);
```

Messages

11:38:55 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.242

15) Create Expense Table:

```
176    -- Creating Expense table
177    CREATE TABLE Expense (
178        employee_SSN CHAR(9),
179        date DATE,
180        amount DECIMAL(10, 2),
181        description TEXT,
182        PRIMARY KEY (employee_SSN, date),
183        FOREIGN KEY (employee_SSN) REFERENCES Employee(SSN)
184    );
185
186    -- Creating index on date in Expense table
187    CREATE NONCLUSTERED INDEX idx_expense_date ON Expense(date);
```

Messages

11:39:35 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.247

16) Create Check Table:

```
189    -- Creating Check table
190    CREATE TABLE [Check] (
191        Donor_SSN CHAR(9),
192        date DATE,
193        check_number CHAR(10),
194        campaign_name VARCHAR(100),
195        type VARCHAR(50),
196        PRIMARY KEY (Donor_SSN, check_number),
197        FOREIGN KEY (Donor_SSN) REFERENCES Donor(SSN)
198    );
199
200    -- Creating index on Donor_SSN in Check table
201    CREATE NONCLUSTERED INDEX idx_check_donor_ssn ON [Check](Donor_SSN);
```

Messages

11:40:55 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.451

17) Create Credit_Card table:

```sql
-- Creating Credit_Card table
CREATE TABLE Credit_Card (
    Donor_SSN CHAR(9),
    date DATE,
    card_number CHAR(16),
    card_type VARCHAR(50),
    Amount DECIMAL(10, 2),
    type VARCHAR(50),
    campaign_name VARCHAR(100),
    expiration_date DATE,
    PRIMARY KEY (Donor_SSN, card_number),
    FOREIGN KEY (Donor_SSN) REFERENCES Donor(SSN)
);

-- Creating index on Donor_SSN in Credit_Card table
CREATE NONCLUSTERED INDEX idx_credit_card_donor_ssn ON Credit_Card(Donor_SSN);

```

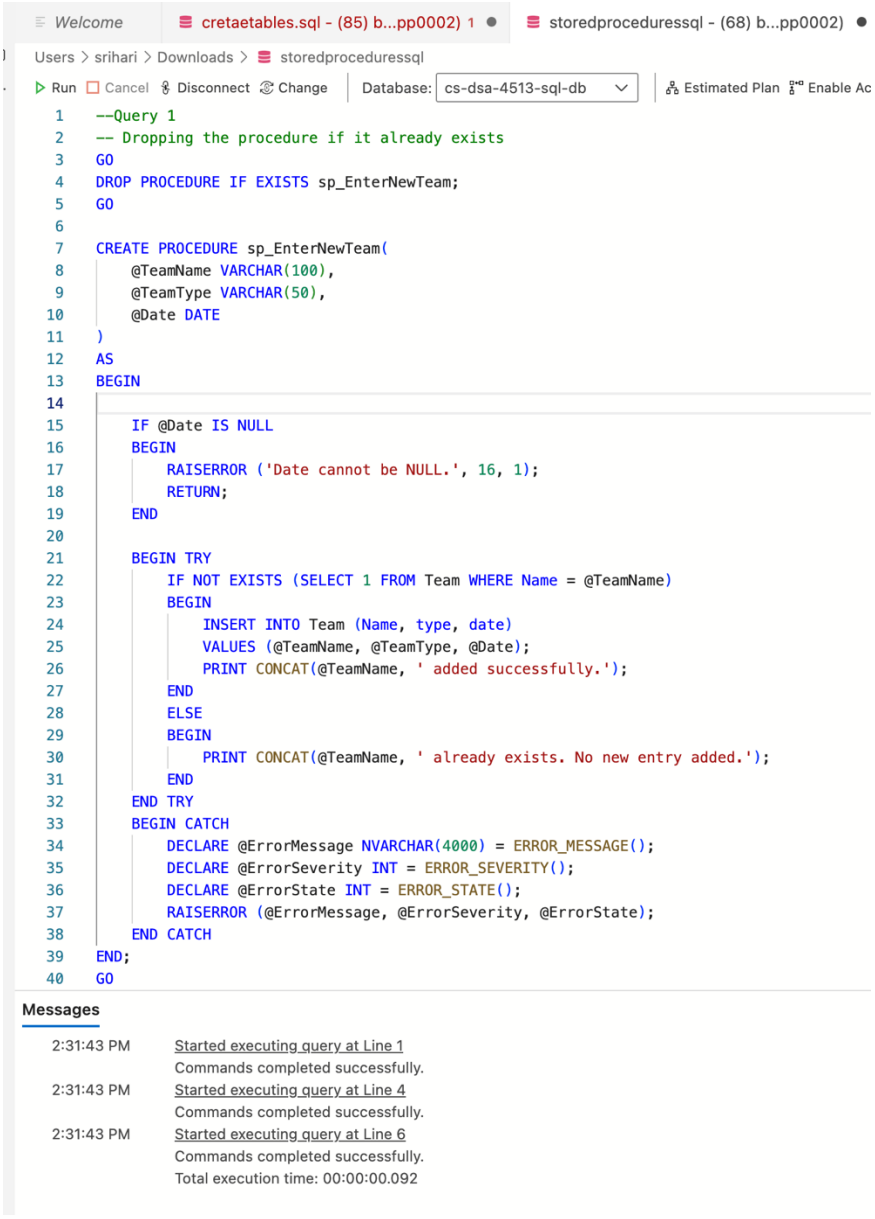
Messages

11:41:26 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.329

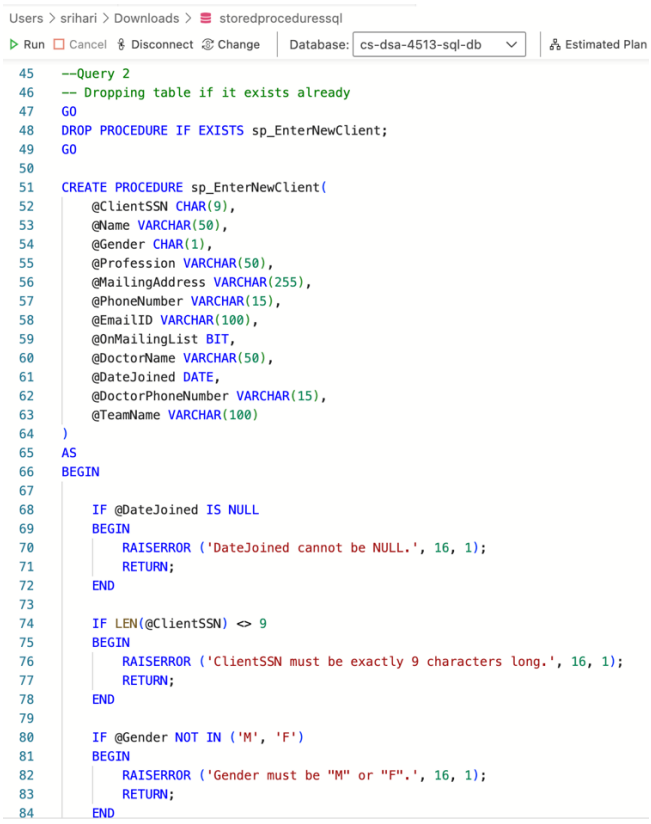
Task 5.

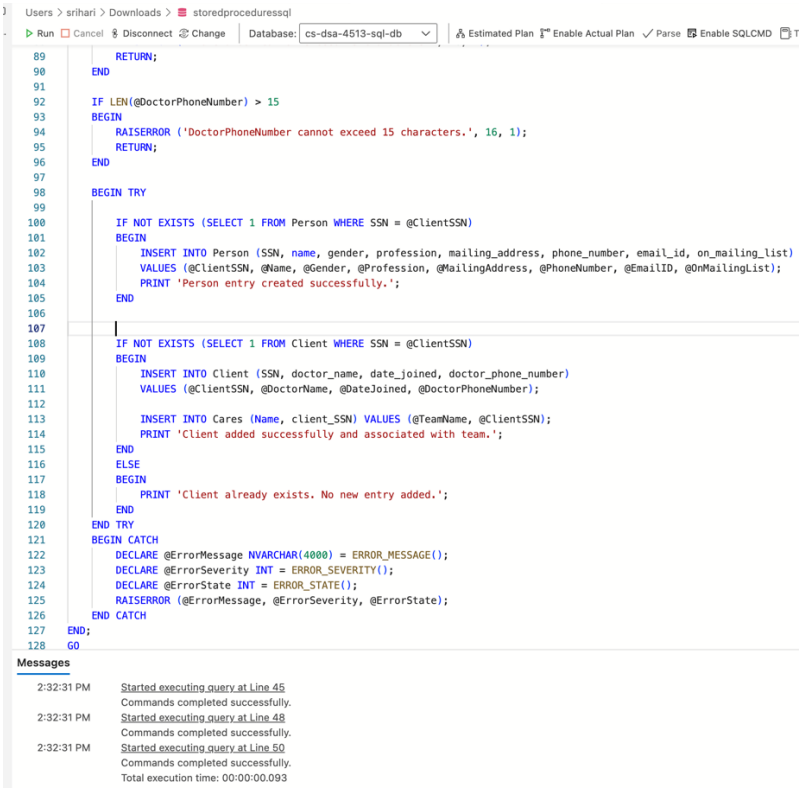
5,1 SQL statements (and Transact SQL stored procedure) Implementing all queries (1-15 and error checking)

1. Enter a new team into the database (1/month).

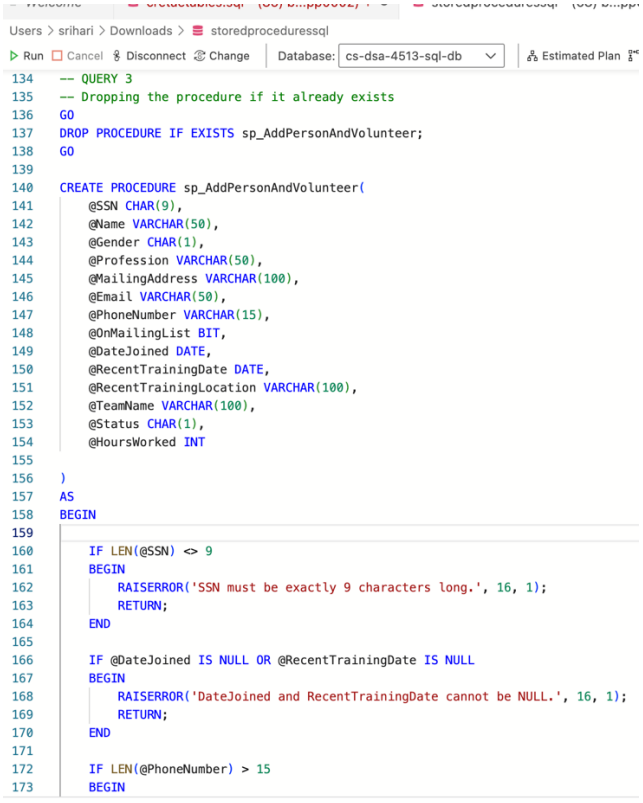


2. Enter a new client into the database and associate him or her with one or more teams (1/week).





3. Enter a new volunteer into the database and associate him or her with one or more teams (2/month).

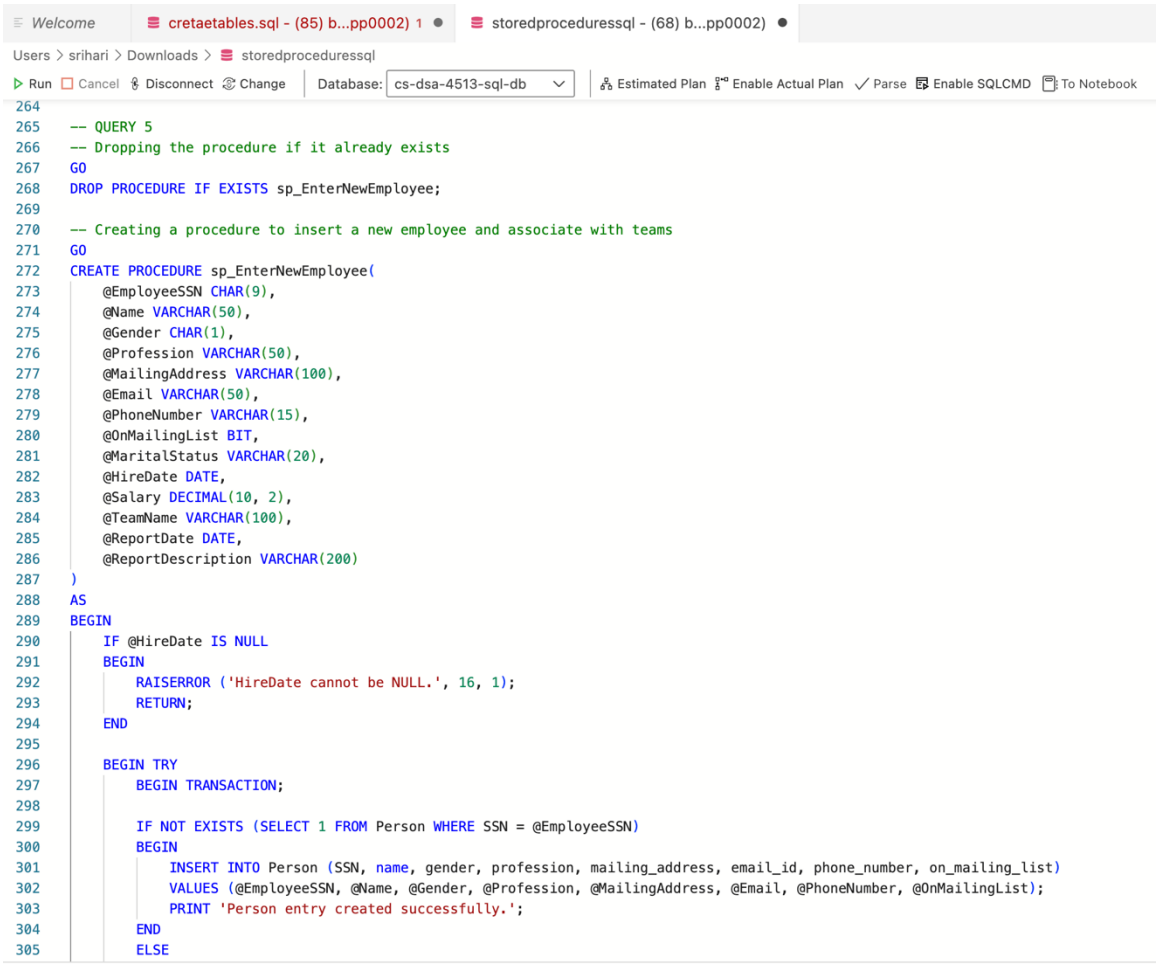


```
Users > srihari > Downloads > storedproceduressql
Run  Cancel  Disconnect  Change   Database: cs-dsa-4513-sql-db   Estimated Plan
134  -- QUERY 3
135  -- Dropping the procedure if it already exists
136  GO
137  DROP PROCEDURE IF EXISTS sp_AddPersonAndVolunteer;
138  GO
139
140  CREATE PROCEDURE sp_AddPersonAndVolunteer(
141      @SSN CHAR(9),
142      @Name VARCHAR(50),
143      @Gender CHAR(1),
144      @Profession VARCHAR(50),
145      @MailingAddress VARCHAR(100),
146      @Email VARCHAR(50),
147      @PhoneNumber VARCHAR(15),
148      @OnMailingList BIT,
149      @DateJoined DATE,
150      @RecentTrainingDate DATE,
151      @RecentTrainingLocation VARCHAR(100),
152      @TeamName VARCHAR(100),
153      @Status CHAR(1),
154      @HoursWorked INT
155
156  )
157  AS
158  BEGIN
159
160      IF LEN(@SSN) <> 9
161      BEGIN
162          RAISERROR('SSN must be exactly 9 characters long.', 16, 1);
163          RETURN;
164      END
165
166      IF @DateJoined IS NULL OR @RecentTrainingDate IS NULL
167      BEGIN
168          RAISERROR('DateJoined and RecentTrainingDate cannot be NULL.', 16, 1);
169          RETURN;
170      END
171
172      IF LEN(@PhoneNumber) > 15
173      BEGIN
```

```
Welcome  cretaetables.sql - (85) b...pp0002) 1  storedproceduressql - (68) b...pp0002)
Users > srihari > Downloads > storedproceduressql
Run  Cancel  Disconnect  Change   Database: cs-dsa-4513-sql-db   Estimated Plan  Enable Actual Plan  Parse  Enable SQLCMD
175          RETURN;
176      END
177
178      IF NOT EXISTS (SELECT 1 FROM Team WHERE Name = @TeamName)
179      BEGIN
180          RAISERROR('TeamName does not exist in the Team table.', 16, 1);
181          RETURN;
182      END
183
184      BEGIN TRY
185          BEGIN TRANSACTION;
186
187          IF NOT EXISTS (SELECT 1 FROM Person WHERE SSN = @SSN)
188          BEGIN
189              INSERT INTO Person (SSN, name, gender, profession, mailing_address, email_id, phone_number, on_mailing_list)
190              VALUES (@SSN, @Name, @Gender, @Profession, @MailingAddress, @Email, @PhoneNumber, @OnMailingList);
191              PRINT 'Person entry created successfully.';
192          END
193          ELSE
194          BEGIN
195              PRINT 'Person with this SSN already exists. Proceeding to add volunteer details.';
196          END
197
198          INSERT INTO Volunteer (SSN, date_joined, recent_training_date, recent_training_location)
199          VALUES (@SSN, @DateJoined, @RecentTrainingDate, @RecentTrainingLocation);
200
201          INSERT INTO Serves (Volunteer_SSN, Team_Name, active_inactive, hours_worked, month)
202          VALUES (@SSN, @TeamName, @Status, @HoursWorked, DATEPART(MONTH, GETDATE()));
203
204          COMMIT TRANSACTION;
205
206          PRINT 'Volunteer and person entries added successfully.';
207      END TRY
208      BEGIN CATCH
209          ROLLBACK TRANSACTION;
210          DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
211          RAISERROR (@ErrorMessage, 16, 1);
212      END CATCH
213  END;
214  GO
Messages
2:33:35 PM  Started executing query at Line 134
            Commands completed successfully.
2:33:35 PM  Started executing query at Line 137
            Commands completed successfully.
2:33:35 PM  Started executing query at Line 139
            Commands completed successfully.
            Total execution time: 00:00:00.094
```

4. Enter the number of hours a volunteer worked this month for a particular team (30/month).

5. Enter a new employee into the database and associate him or her with one or more teams (1/year).




```
264
265   -- QUERY 5
266   -- Dropping the procedure if it already exists
267   GO
268   DROP PROCEDURE IF EXISTS sp_EnterNewEmployee;
269
270   -- Creating a procedure to insert a new employee and associate with teams
271   GO
272   CREATE PROCEDURE sp_EnterNewEmployee(
273       @EmployeeSSN CHAR(9),
274       @Name VARCHAR(50),
275       @Gender CHAR(1),
276       @Profession VARCHAR(50),
277       @MailingAddress VARCHAR(100),
278       @Email VARCHAR(50),
279       @PhoneNumber VARCHAR(15),
280       @OnMailingList BIT,
281       @MaritalStatus VARCHAR(20),
282       @HireDate DATE,
283       @Salary DECIMAL(10, 2),
284       @TeamName VARCHAR(100),
285       @ReportDate DATE,
286       @ReportDescription VARCHAR(200)
287   )
288   AS
289   BEGIN
290       IF @HireDate IS NULL
291       BEGIN
292           RAISERROR ('HireDate cannot be NULL.', 16, 1);
293           RETURN;
294       END
295
296       BEGIN TRY
297           BEGIN TRANSACTION;
298
299           IF NOT EXISTS (SELECT 1 FROM Person WHERE SSN = @EmployeeSSN)
300           BEGIN
301               INSERT INTO Person (SSN, name, gender, profession, mailing_address, email_id, phone_number, on_mailing_list)
302               VALUES (@EmployeeSSN, @Name, @Gender, @Profession, @MailingAddress, @Email, @PhoneNumber, @OnMailingList);
303               PRINT 'Person entry created successfully.';
304           END
305           ELSE
```


Welcome
cretaetables.sql - (85) b...pp0002) 1
storedproceduressql - (68) b...pp0002)
Users > srihari > Downloads > storedproceduressql
Run Cancel Disconnect Change
Database: cs-dsa-4513-sql-db
Estimated Plan Enable Actual Plan Parse Enable SQLCMD
)
AS
BEGIN
IF @HireDate IS NULL
BEGIN
RAISERROR ('HireDate cannot be NULL.', 16, 1);
RETURN;
END
BEGIN TRY
BEGIN TRANSACTION;
IF NOT EXISTS (SELECT 1 FROM Person WHERE SSN = @EmployeeSSN)
BEGIN
INSERT INTO Person (SSN, name, gender, profession, mailing_address, email_id, phone_number, on_mailing_list)
VALUES (@EmployeeSSN, @Name, @Gender, @Profession, @MailingAddress, @Email, @PhoneNumber, @OnMailingList);
PRINT 'Person entry created successfully.';
END
ELSE
BEGIN
PRINT 'Person with this SSN already exists. Proceeding to add employee details.';
END
INSERT INTO Employee (SSN, marital_status, hire_date, salary)
VALUES (@EmployeeSSN, @MaritalStatus, @HireDate, @Salary);
INSERT INTO Reports (employee_SSN, team_name, date, description)
VALUES (@EmployeeSSN, @TeamName, @ReportDate, @ReportDescription);
COMMIT TRANSACTION;
PRINT 'Employee and person entries added successfully and associated with team.';
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION;
DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
DECLARE @ErrorState INT = ERROR_STATE();
RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
END CATCH
END;
GO
Messages
2:36:09 PM Started executing query at Line 265
Commands completed successfully.
2:36:09 PM Started executing query at Line 268
Commands completed successfully.
2:36:09 PM Started executing query at Line 272
Commands completed successfully.
Total execution time: 00:00:00.143
Ln 327, Col 5 Spaces: 4 UTF-8 LF SQL 0 rows MSSQL 00:00:00

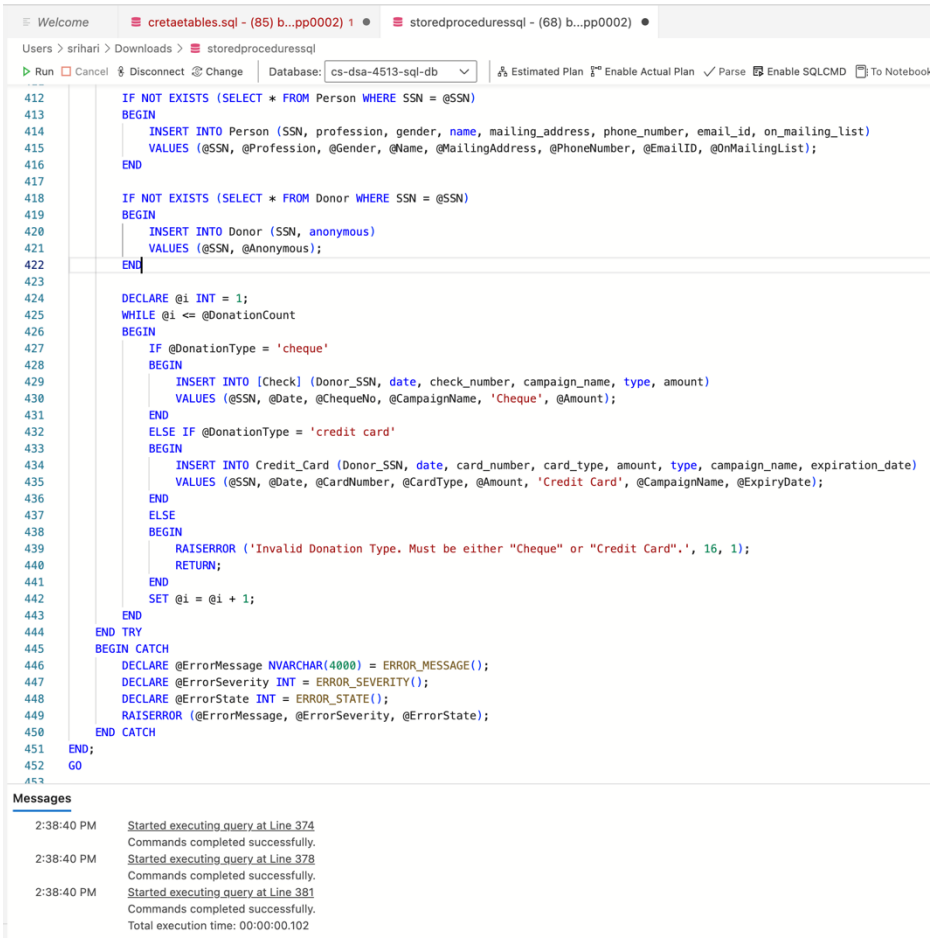
6. Enter an expense charged by an employee (1/day).

```
331  -- QUERY 6
332  -- Dropping the procedure if it already exists
333  GO
334  DROP PROCEDURE IF EXISTS sp_EnterExpense;
335
336  -- Creating a procedure to enter an expense charged by an employee
337  GO
338  CREATE PROCEDURE sp_EnterExpense(
339      @EmployeeSSN CHAR(9),
340      @ExpenseDate DATE,
341      @Amount DECIMAL(10, 2),
342      @Description VARCHAR(200)
343  )
344  AS
345  BEGIN
346      IF @ExpenseDate IS NULL OR @Amount IS NULL
347      BEGIN
348          RAISERROR ('ExpenseDate and Amount cannot be NULL.', 16, 1);
349          RETURN;
350      END
351
352      BEGIN TRY
353          IF EXISTS (SELECT 1 FROM Employee WHERE SSN = @EmployeeSSN)
354          BEGIN
355              INSERT INTO Expense (employee_SSN, date, amount, description)
356              VALUES (@EmployeeSSN, @ExpenseDate, @Amount, @Description);
357              PRINT 'Expense entry created successfully.';
358          END
359          ELSE
360          BEGIN
361              RAISERROR('Error: Employee SSN does not exist. Cannot create expense entry.', 16, 1);
362          END
363      END TRY
364      BEGIN CATCH
365          DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
366          DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
367          DECLARE @ErrorState INT = ERROR_STATE();
368          RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
369      END CATCH
370  END;
371  GO
```

Messages

```
2:37:23 PM    Started executing query at Line 331
              Commands completed successfully.
2:37:23 PM    Started executing query at Line 334
              Commands completed successfully.
2:37:23 PM    Started executing query at Line 338
              Commands completed successfully.
              Total execution time: 00:00:00.158
```

7. Enter a new donor and associate him or her with several donations (1/day).


Welcome
cretaetables.sql - (85) b...pp0002) 1
storedproceduressql - (68) b...pp0002)
Users > srihari > Downloads > storedproceduressql
Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db Estimated Plan Enable Actual Plan Parse Enable SQLCMD To Notebook
412 IF NOT EXISTS (SELECT * FROM Person WHERE SSN = @SSN)
413 BEGIN
414 INSERT INTO Person (SSN, profession, gender, name, mailing_address, phone_number, email_id, on_mailing_list)
415 VALUES (@SSN, @Profession, @Gender, @Name, @MailingAddress, @PhoneNumber, @EmailID, @OnMailingList);
416 END
417
418 IF NOT EXISTS (SELECT * FROM Donor WHERE SSN = @SSN)
419 BEGIN
420 INSERT INTO Donor (SSN, anonymous)
421 VALUES (@SSN, @Anonymous);
422 END
423
424 DECLARE @i INT = 1;
425 WHILE @i <= @DonationCount
426 BEGIN
427 IF @DonationType = 'cheque'
428 BEGIN
429 INSERT INTO [Check] (Donor_SSN, date, check_number, campaign_name, type, amount)
430 VALUES (@SSN, @Date, @ChequeNo, @CampaignName, 'Cheque', @Amount);
431 END
432 ELSE IF @DonationType = 'credit card'
433 BEGIN
434 INSERT INTO Credit_Card (Donor_SSN, date, card_number, card_type, amount, type, campaign_name, expiration_date)
435 VALUES (@SSN, @Date, @CardNumber, @CardType, @Amount, 'Credit Card', @CampaignName, @ExpiryDate);
436 END
437 ELSE
438 BEGIN
439 RAISERROR ('Invalid Donation Type. Must be either "Cheque" or "Credit Card".', 16, 1);
440 RETURN;
441 END
442 SET @i = @i + 1;
443 END
444 END TRY
445 BEGIN CATCH
446 DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
447 DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
448 DECLARE @ErrorState INT = ERROR_STATE();
449 RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
450 END CATCH
451 END;
452 GO
Messages
2:38:40 PM Started executing query at Line 374
Commands completed successfully.
2:38:40 PM Started executing query at Line 378
Commands completed successfully.
2:38:40 PM Started executing query at Line 381
Commands completed successfully.
Total execution time: 00:00:00.102

8. Retrieve the name and phone number of the doctor of a particular client (1/week).

```
330   -- QUERY 8
331   -- Dropping the procedure if it already exists
332   GO
333   DROP PROCEDURE IF EXISTS sp_RetrieveClientDoctor;
334
335   -- Creating a procedure to retrieve the doctor information of a client
336   GO
337   CREATE PROCEDURE sp_RetrieveClientDoctor(
338       @ClientSSN CHAR(9)
339   )
340   AS
341   BEGIN
342       SELECT doctor_name, doctor_phone_number
343       FROM Client
344       WHERE SSN = @ClientSSN;
345   END;
346   GO
347
```

Messages

```
2:03:03 AM    Started executing query at Line 330
              Commands completed successfully.
2:03:03 AM    Started executing query at Line 333
              Commands completed successfully.
2:03:03 AM    Started executing query at Line 337
              Commands completed successfully.
              Total execution time: 00:00:00.120
```

9. Retrieve the total amount of expenses charged by each employee for a particular period of time. The list should be sorted by the total amount of expenses (1/month).

```
348   -- QUERY 9
349   -- Dropping the procedure if it already exists
350   GO
351   DROP PROCEDURE IF EXISTS sp_TotalExpensesByEmployee;
352
353   -- Creating a procedure to retrieve total expenses charged by each employee for a period
354   GO
355   CREATE PROCEDURE sp_TotalExpensesByEmployee(
356       @StartDate DATE,
357       @EndDate DATE
358   )
359   AS
360   BEGIN
361       SELECT employee_SSN, SUM(amount) AS total_expenses
362       FROM Expense
363       WHERE date BETWEEN @StartDate AND @EndDate
364       GROUP BY employee_SSN
365       ORDER BY total_expenses DESC;
366   END;
367   GO
368
369   -- QUERY 10
370   -- Dropping the procedure if it already exists
371   GO
372   DROP PROCEDURE IF EXISTS sp_VolunteersForClientTeams;
373
374   -- Creating a procedure to list volunteers for a client's teams
375   GO
376   CREATE PROCEDURE sp_VolunteersForClientTeams(
377       @ClientSSN CHAR(9)
378   )
```

Messages

```
2:03:03 AM    Started executing query at Line 330
              Commands completed successfully.
2:03:03 AM    Started executing query at Line 333
              Commands completed successfully.
2:03:03 AM    Started executing query at Line 337
              Commands completed successfully.
              Total execution time: 00:00:00.120
```

10. Retrieve the list of volunteers that are members of teams that support a particular client (4/year).

```
369   -- QUERY 10
370   -- Dropping the procedure if it already exists
371   GO
372   DROP PROCEDURE IF EXISTS sp_VolunteersForClientTeams;
373
374   -- Creating a procedure to list volunteers for a client's teams
375   GO
376   CREATE PROCEDURE sp_VolunteersForClientTeams(
377       @ClientSSN CHAR(9)
378   )
379   AS
380   BEGIN
381       SELECT Volunteer.SSN, Volunteer.recent_training_location
382       FROM Volunteer
383       JOIN Serves ON Volunteer.SSN = Serves.Volunteer_SSN
384       JOIN Cares ON Serves.Team_Name = Cares.Name
385       WHERE Cares.client_SSN = @ClientSSN;
386   END;
387   GO
388
389
```

Messages

```
2:05:32 AM    Started executing query at Line 369
              Commands completed successfully.
2:05:32 AM    Started executing query at Line 372
              Commands completed successfully.
2:05:32 AM    Started executing query at Line 376
              Commands completed successfully.
              Total execution time: 00:00:00.133
```

11. Retrieve the names of all teams that were founded after a particular date (1/month)

```
-- QUERY 11
-- Dropping the procedure if it already exists
GO
DROP PROCEDURE IF EXISTS sp_RetrieveTeamsFoundedAfterDate;

-- Creating a procedure to retrieve team names founded after a specified date
GO
CREATE PROCEDURE sp_RetrieveTeamsFoundedAfterDate(
    @Date DATE
)
AS
BEGIN
    SELECT Name
    FROM Team
    WHERE date > @Date;
END;
GO
```

ssages

2:05:32 AM Started executing query at Line 369
Commands completed successfully.
2:05:32 AM Started executing query at Line 372
Commands completed successfully.
2:05:32 AM Started executing query at Line 376
Commands completed successfully.

12. Retrieve the names, social security numbers, contact information, and emergency contact information of all people in the database (1/week).

```
-- QUERY 12
-- Dropping the procedure if it already exists
GO
DROP PROCEDURE IF EXISTS sp_RetrieveAllPeopleInfo;

-- Creating a procedure to retrieve all personal and emergency contact information for everyone in the database
GO
CREATE PROCEDURE sp_RetrieveAllPeopleInfo
AS
BEGIN
    SELECT
        P.SSN,
        P.name AS PersonName,
        P.phone_number AS PhoneNumber,
        P.email_id AS EmailID,
        P.mailing_address AS MailingAddress,
        EC.name AS EmergencyContactName,
        EC.phone_number AS EmergencyContactPhone,
        EC.relationship AS EmergencyContactRelation
    FROM
        Person P
    LEFT JOIN
        Emergency_Contact EC ON P.SSN = EC.SSN;
END;
GO
```

essages

2:06:26 AM Started executing query at Line 408
Commands completed successfully.
2:06:26 AM Started executing query at Line 412
Commands completed successfully.
2:06:26 AM Started executing query at Line 416
Commands completed successfully.
Total execution time: 00:00:00.131

13. Retrieve the name and total amount donated by donors that are also employees. The list should be sorted by the total amount of the donations, and indicate if each donor wishes to remain anonymous (1/week)

```
435   -- QUERY 13
436   -- Dropping the procedure if it already exists
437
438   GO
439   DROP PROCEDURE IF EXISTS sp_RetrieveEmployeeDonors;
440
441   GO
442   CREATE PROCEDURE sp_RetrieveEmployeeDonors
443   AS
444   BEGIN
445       SELECT
446           D.SSN,
447           D.anonymous AS IsAnonymous,
448           ISNULL(SUM(CC.amount), 0) AS TotalCreditDonation,
449           ISNULL(SUM(CK.amount), 0) AS TotalCheckDonation,
450           ISNULL(SUM(CC.amount), 0) + ISNULL(SUM(CK.amount), 0) AS TotalDonation
451       FROM
452           Donor D
453       LEFT JOIN
454           Credit_Card CC ON D.SSN = CC.Donor_SSN
455       LEFT JOIN
456           [Check] CK ON D.SSN = CK.Donor_SSN
457       GROUP BY
458           D.SSN, D.anonymous
459       HAVING
460           ISNULL(SUM(CC.amount), 0) + ISNULL(SUM(CK.amount), 0) > 0
461       ORDER BY
462           TotalDonation DESC;
463   END;
464   GO
465
466
467
468
469
```

Messages

```
2:06:55 AM    Started executing query at Line 435
              Commands completed successfully.
2:06:55 AM    Started executing query at Line 439
              Commands completed successfully.
2:06:55 AM    Started executing query at Line 442
              Commands completed successfully.
              Total execution time: 00:00:00.118
```

14. Increase the salary by 10% of all employees to whom more than one team must report. (1/year)

```sql
-- QUERY 14
-- Dropping the procedure if it already exists
GO
DROP PROCEDURE IF EXISTS sp_IncreaseSalaryForMultiTeamEmployees;

-- Creating a procedure to increase salary by 10% for employees managing multiple teams
GO
CREATE PROCEDURE sp_IncreaseSalaryForMultiTeamEmployees
AS
BEGIN
    UPDATE Employee
    SET salary = salary * 1.10
    WHERE SSN IN (
        SELECT employee_SSN
        FROM Reports
        GROUP BY employee_SSN
        HAVING COUNT(DISTINCT team_name) > 1
    );
    PRINT 'Salaries updated successfully for eligible employees.';
END;
GO
```

Messages

```
2:07:20 AM    Started executing query at Line 471
              Commands completed successfully.
2:07:20 AM    Started executing query at Line 474
              Commands completed successfully.
2:07:20 AM    Started executing query at Line 478
              Commands completed successfully.
              Total execution time: 00:00:00.121
```

Ln 466, Col 1 Spaces: 4 UTF-8

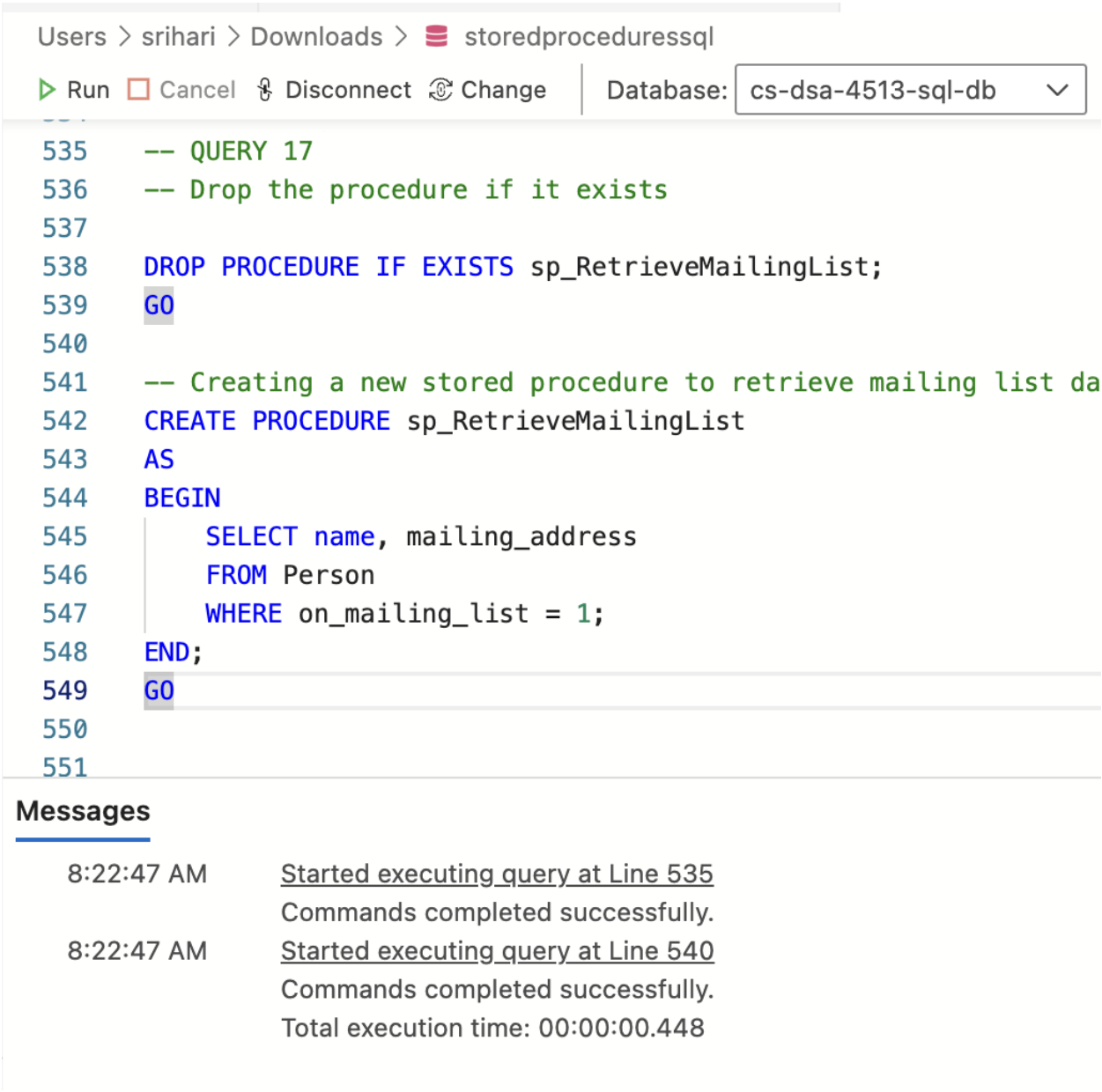
15. Delete all clients who do not have health insurance and whose value of importance for transportation is less than 5 (4/year).

```
493   --Query 15
494   GO
495   DROP PROCEDURE IF EXISTS sp_DeleteUninsuredClientsWithLowTransportValue;
496
497   GO
498   CREATE PROCEDURE sp_DeleteUninsuredClientsWithLowTransportValue
499   AS
500   BEGIN
501       DELETE FROM Cares
502       WHERE client_SSN IN (
503           SELECT C.SSN
504           FROM Client C
505           LEFT JOIN Covered_By CB ON C.SSN = CB.client_SSN
506           LEFT JOIN Needs N ON C.SSN = N.client_SSN AND N.type = 'Transportation'
507           WHERE CB.policy_id IS NULL
508             AND (N.importance IS NULL OR N.importance < 5)
509       );
510       DELETE FROM Needs
511       WHERE client_SSN IN (
512           SELECT C.SSN
513           FROM Client C
514           LEFT JOIN Covered_By CB ON C.SSN = CB.client_SSN
515           LEFT JOIN Needs N ON C.SSN = N.client_SSN AND N.type = 'Transportation'
516           WHERE CB.policy_id IS NULL
517             AND (N.importance IS NULL OR N.importance < 5)
518       );
519       DELETE FROM Client
520       WHERE SSN IN (
521           SELECT C.SSN
522           FROM Client C
523           LEFT JOIN Covered_By CB ON C.SSN = CB.client_SSN
524           LEFT JOIN Needs N ON C.SSN = N.client_SSN AND N.type = 'Transportation'
525           WHERE CB.policy_id IS NULL
526             AND (N.importance IS NULL OR N.importance < 5)
527       );
528       PRINT 'Eligible clients and their associated records in Needs and Cares were deleted successfully.';
529   END;
530   GO
531
```

essages

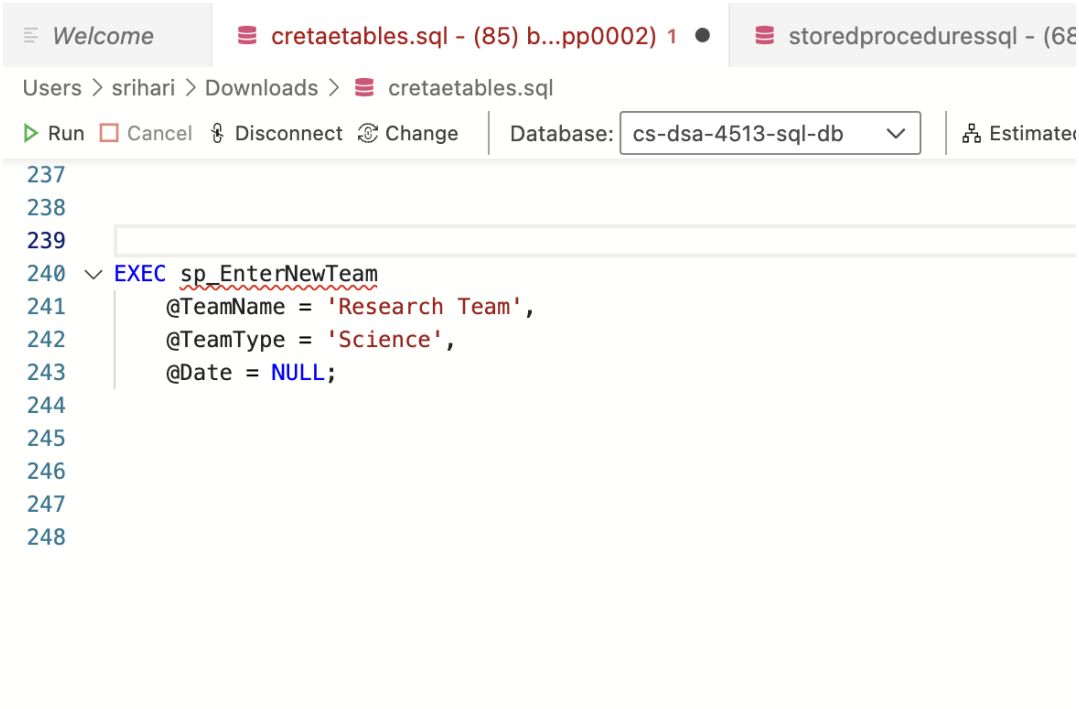
```
2:08:12 AM    Started executing query at Line 493
              Commands completed successfully.
2:08:12 AM    Started executing query at Line 495
              Commands completed successfully.
2:08:12 AM    Started executing query at Line 498
              Commands completed successfully.
              Total execution time: 00:00:00.122
```

Query 17 for retrieving mailing list –



Error checking – Stored procedures can handle errors

1- Null Date error detection for query 1



Messages

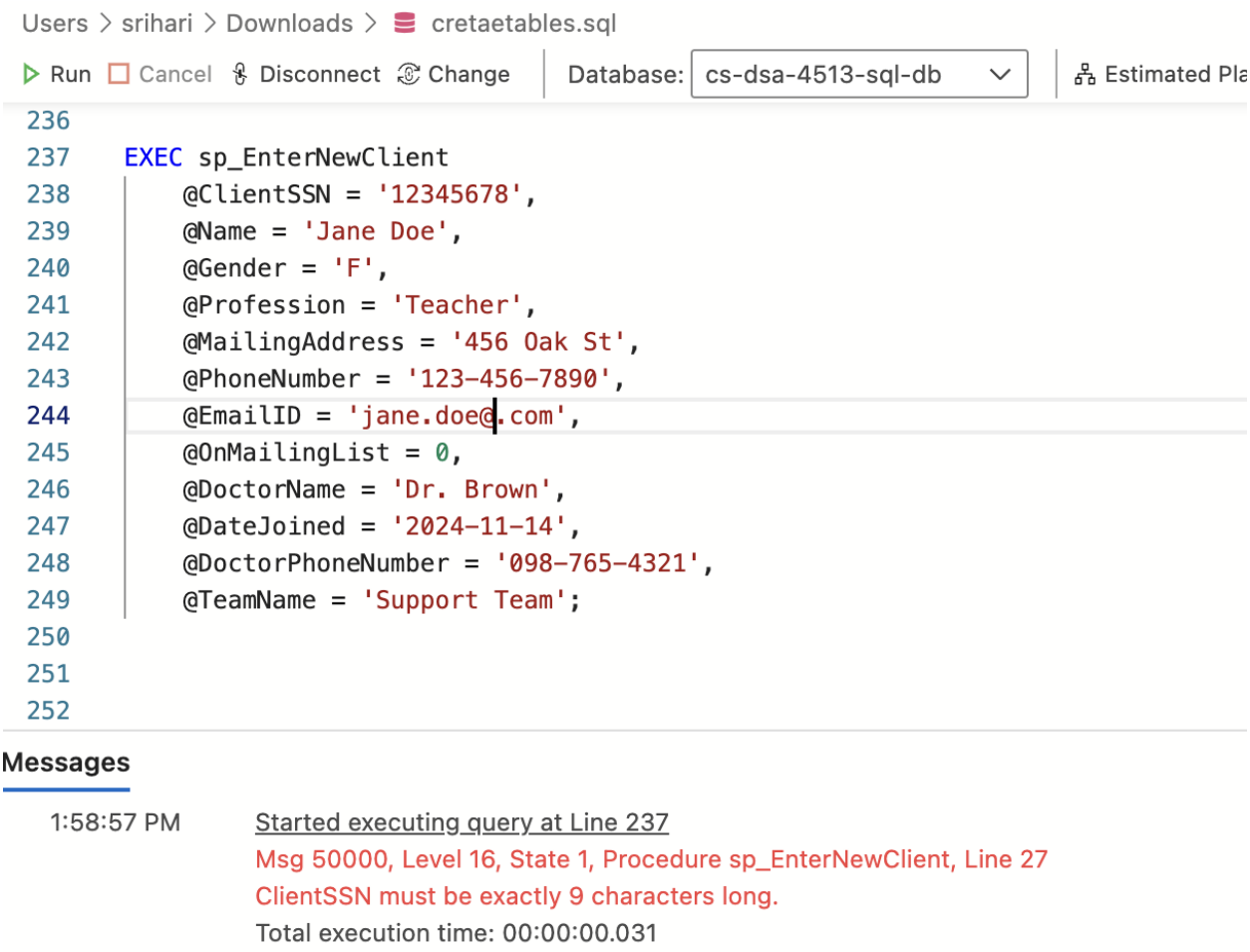
2:21:56 PM Started executing query at Line 239

Msg 50000, Level 16, State 1, Procedure sp_EnterNewTeam, Line 12

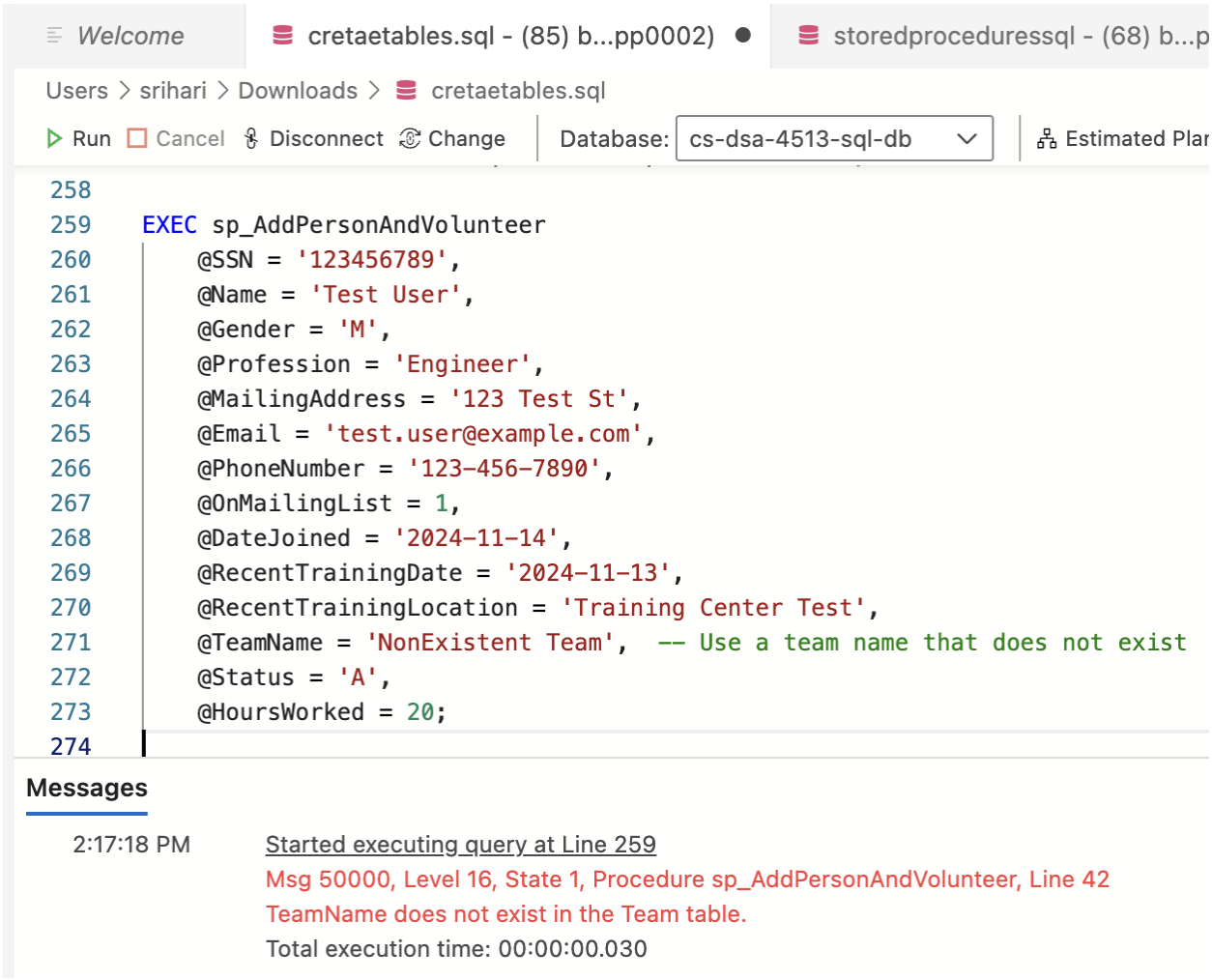
Date cannot be NULL.

Total execution time: 00:00:00.050

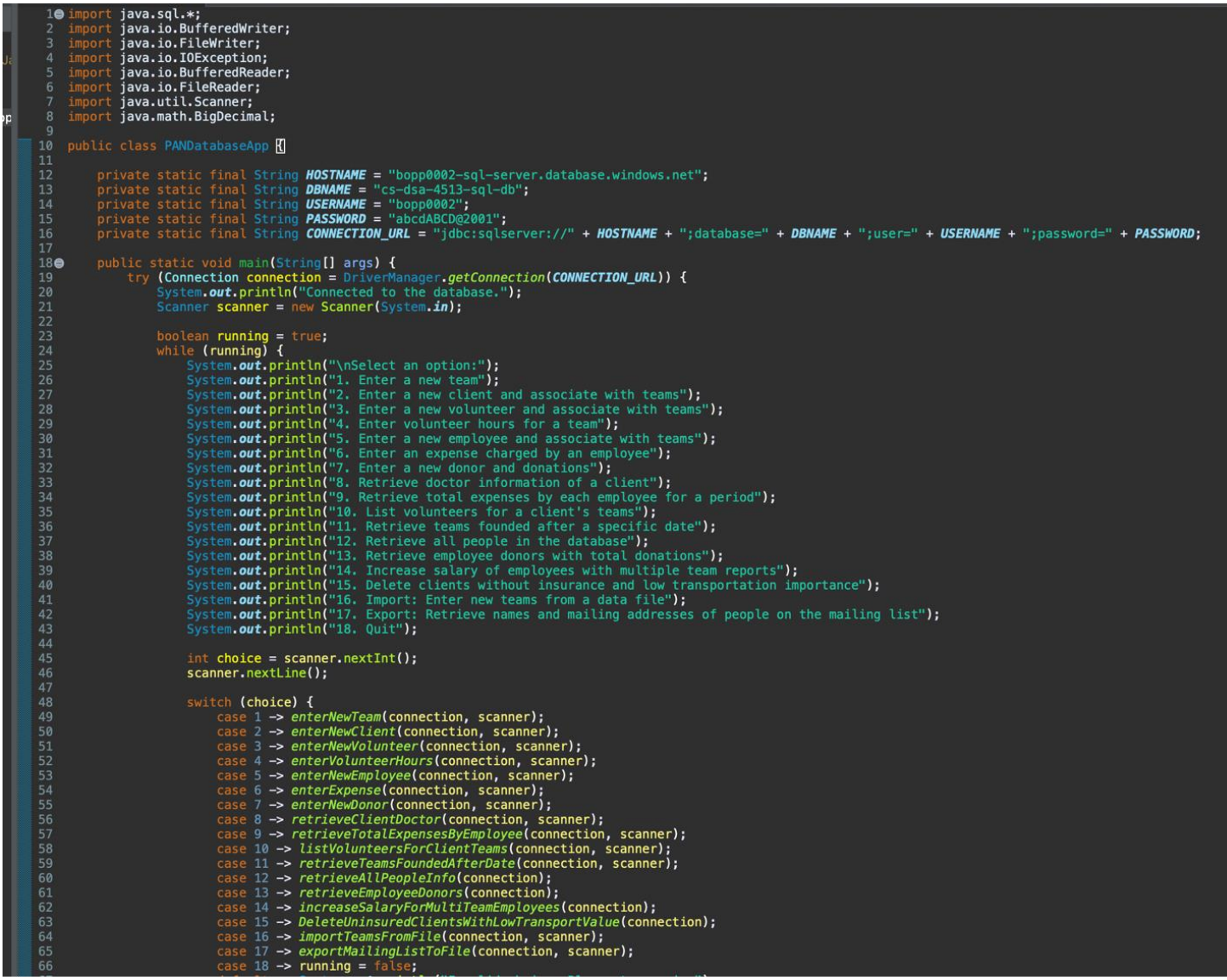
2- Error handling for Query 2 which detects the character length of ClientSSN is not exactly of same length.



3- Error Handling for query 3 for creating a non existing table “Team Name”



5,2 The Java source program and screenshots showing its successful compilation



```java
import java.sql.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Scanner;
import java.math.BigDecimal;

public class PANDatabaseApp {

    private static final String HOSTNAME = "bopp0002-sql-server.database.windows.net";
    private static final String DBNAME = "cs-dsa-4513-sql-db";
    private static final String USERNAME = "bopp0002";
    private static final String PASSWORD = "abcdABCD@2001";
    private static final String CONNECTION_URL = "jdbc:sqlserver://" + HOSTNAME + ";database=" + DBNAME + ";user=" + USERNAME + ";password=" + PASSWORD;

    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection(CONNECTION_URL)) {
            System.out.println("Connected to the database.");
            Scanner scanner = new Scanner(System.in);

            boolean running = true;
            while (running) {
                System.out.println("\nSelect an option:");
                System.out.println("1. Enter a new team");
                System.out.println("2. Enter a new client and associate with teams");
                System.out.println("3. Enter a new volunteer and associate with teams");
                System.out.println("4. Enter volunteer hours for a team");
                System.out.println("5. Enter a new employee and associate with teams");
                System.out.println("6. Enter an expense charged by an employee");
                System.out.println("7. Enter a new donor and donations");
                System.out.println("8. Retrieve doctor information of a client");
                System.out.println("9. Retrieve total expenses by each employee for a period");
                System.out.println("10. List volunteers for a client's teams");
                System.out.println("11. Retrieve teams founded after a specific date");
                System.out.println("12. Retrieve all people in the database");
                System.out.println("13. Retrieve employee donors with total donations");
                System.out.println("14. Increase salary of employees with multiple team reports");
                System.out.println("15. Delete clients without insurance and low transportation importance");
                System.out.println("16. Import: Enter new teams from a data file");
                System.out.println("17. Export: Retrieve names and mailing addresses of people on the mailing list");
                System.out.println("18. Quit");

                int choice = scanner.nextInt();
                scanner.nextLine();

                switch (choice) {
                    case 1 -> enterNewTeam(connection, scanner);
                    case 2 -> enterNewClient(connection, scanner);
                    case 3 -> enterNewVolunteer(connection, scanner);
                    case 4 -> enterVolunteerHours(connection, scanner);
                    case 5 -> enterNewEmployee(connection, scanner);
                    case 6 -> enterExpense(connection, scanner);
                    case 7 -> enterNewDonor(connection, scanner);
                    case 8 -> retrieveClientDoctor(connection, scanner);
                    case 9 -> retrieveTotalExpensesByEmployee(connection, scanner);
                    case 10 -> listVolunteersForClientTeams(connection, scanner);
                    case 11 -> retrieveTeamsFoundedAfterDate(connection, scanner);
                    case 12 -> retrieveAllPeopleInfo(connection);
                    case 13 -> retrieveEmployeeDonors(connection);
                    case 14 -> increaseSalaryForMultiTeamEmployees(connection);
                    case 15 -> DeleteUninsuredClientsWithLowTransportValue(connection);
                    case 16 -> importTeamsFromFile(connection, scanner);
                    case 17 -> exportMailingListToFile(connection, scanner);
                    case 18 -> running = false;
```

```
                    case 18 -> running = false;
                    default -> System.out.println("Invalid choice. Please try again.");
                }
            }
            System.out.println("Exiting the application.");
        } catch (SQLException e) {
            System.out.println("Error connecting to the database: " + e.getMessage());
        }
    }

    private static void enterNewTeam(Connection connection, Scanner scanner) {
        System.out.print("Enter team name: ");
        String teamName = scanner.nextLine();
        System.out.print("Enter team type: ");
        String teamType = scanner.nextLine();
        System.out.print("Enter formation date (yyyy-mm-dd): ");
        String date = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewTeam(?, ?, ?)}")) {
            stmt.setString(1, teamName);
            stmt.setString(2, teamType);
            stmt.setDate(3, Date.valueOf(date));
            stmt.execute();
            System.out.println("Team added successfully.");
        } catch (SQLException e) {
            System.out.println("Error adding team: " + e.getMessage());
        }
    }

    private static void enterNewClient(Connection connection, Scanner scanner) {
        System.out.print("Enter client SSN: ");
        String clientSSN = scanner.nextLine();
        System.out.print("Enter client name: ");
        String name = scanner.nextLine();
        System.out.print("Enter client gender (M/F): ");
        String gender = scanner.nextLine();
        System.out.print("Enter client profession: ");
        String profession = scanner.nextLine();
        System.out.print("Enter client mailing address: ");
        String mailingAddress = scanner.nextLine();
        System.out.print("Enter client phone number: ");
        String phoneNumber = scanner.nextLine();
        System.out.print("Enter client email ID: ");
        String emailID = scanner.nextLine();
        System.out.print("Is the client on the mailing list? (1 for yes, 0 for no): ");
        boolean onMailingList = scanner.nextInt() == 1;
        scanner.nextLine();
        System.out.print("Enter doctor name: ");
        String doctorName = scanner.nextLine();
        System.out.print("Enter date joined (yyyy-mm-dd): ");
        String dateJoined = scanner.nextLine();
        System.out.print("Enter doctor phone number: ");
        String doctorPhone = scanner.nextLine();
        System.out.print("Enter team name: ");
        String teamName = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewClient(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
            stmt.setString(1, clientSSN);
            stmt.setString(2, name);
            stmt.setString(3, gender);
            stmt.setString(4, profession);
            stmt.setString(5, mailingAddress);
            stmt.setString(6, phoneNumber);
            stmt.setString(7, emailID);
            stmt.setBoolean(8, onMailingList);
```

```
            stmt.setBoolean(8, onMailingList);
            stmt.setString(9, doctorName);
            stmt.setDate(10, Date.valueOf(dateJoined));
            stmt.setString(11, doctorPhone);
            stmt.setString(12, teamName);
            stmt.execute();
            System.out.println("Client added and associated with team successfully.");
        } catch (SQLException e) {
            System.out.println("Error adding client: " + e.getMessage());
        }
    }



    private static void enterNewVolunteer(Connection connection, Scanner scanner) {
        System.out.print("Enter volunteer SSN: ");
        String volunteerSSN = scanner.nextLine();
        System.out.print("Enter volunteer name: ");
        String name = scanner.nextLine();
        System.out.print("Enter gender (M/F): ");
        String gender = scanner.nextLine();
        System.out.print("Enter profession: ");
        String profession = scanner.nextLine();
        System.out.print("Enter mailing address: ");
        String address = scanner.nextLine();
        System.out.print("Enter email: ");
        String email = scanner.nextLine();
        System.out.print("Enter phone number: ");
        String phone = scanner.nextLine();
        System.out.print("Is the volunteer on the mailing list? (1 for yes, 0 for no): ");
        int onMailingList = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter date joined (yyyy-mm-dd): ");
        String dateJoined = scanner.nextLine();
        System.out.print("Enter recent training date (yyyy-mm-dd): ");
        String trainingDate = scanner.nextLine();
        System.out.print("Enter recent training location: ");
        String trainingLocation = scanner.nextLine();
        System.out.print("Enter team name: ");
        String teamName = scanner.nextLine();
        System.out.print("Enter status (A for active, I for inactive): ");
        String status = scanner.nextLine();
        System.out.print("Enter hours worked: ");
        int hoursWorked = scanner.nextInt();
        scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_AddPersonAndVolunteer(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
            stmt.setString(1, volunteerSSN);
            stmt.setString(2, name);
            stmt.setString(3, gender);
            stmt.setString(4, profession);
            stmt.setString(5, address);
            stmt.setString(6, email);
            stmt.setString(7, phone);
            stmt.setInt(8, onMailingList);
            stmt.setDate(9, Date.valueOf(dateJoined));
            stmt.setDate(10, Date.valueOf(trainingDate));
            stmt.setString(11, trainingLocation);
            stmt.setString(12, teamName);
            stmt.setString(13, status);
            stmt.setInt(14, hoursWorked);
            stmt.execute();
            System.out.println("Volunteer and person entries added successfully and associated with the team.");
        } catch (SQLException e) {
            System.out.println("Error adding volunteer: " + e.getMessage());
```

```
            System.out.println("Error adding volunteer: " + e.getMessage());
        }
    }


    private static void enterVolunteerHours(Connection connection, Scanner scanner) {
        System.out.print("Enter volunteer SSN: ");
        String volunteerSSN = scanner.nextLine();
        System.out.print("Enter team name: ");
        String teamName = scanner.nextLine();
        System.out.print("Enter additional hours worked: ");
        int hours = scanner.nextInt();
        System.out.print("Enter month: ");
        int month = scanner.nextInt();
        System.out.print("Enter status (A for active, I for inactive): ");
        String status = scanner.next();
        scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_EnterVolunteerHours(?, ?, ?, ?, ?)}")) {
            stmt.setString(1, volunteerSSN);
            stmt.setString(2, teamName);
            stmt.setInt(3, hours);
            stmt.setInt(4, month);
            stmt.setString(5, status);
            stmt.execute();
            System.out.println("Volunteer hours updated successfully.");
        } catch (SQLException e) {
            System.out.println("Error updating volunteer hours: " + e.getMessage());
        }
    }



    private static void enterNewEmployee(Connection connection, Scanner scanner) {
        System.out.print("Enter employee SSN: ");
        String employeeSSN = scanner.nextLine();
        System.out.print("Enter employee name: ");
        String name = scanner.nextLine();
        System.out.print("Enter employee gender (M/F): ");
        String gender = scanner.nextLine();
        System.out.print("Enter employee profession: ");
        String profession = scanner.nextLine();
        System.out.print("Enter mailing address: ");
        String mailingAddress = scanner.nextLine();
        System.out.print("Enter email ID: ");
        String email = scanner.nextLine();
        System.out.print("Enter phone number: ");
        String phoneNumber = scanner.nextLine();
        System.out.print("Is the employee on the mailing list? (1 for yes, 0 for no): ");
        int onMailingList = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter marital status: ");
        String maritalStatus = scanner.nextLine();
        System.out.print("Enter hire date (yyyy-mm-dd): ");
        String hireDate = scanner.nextLine();
        System.out.print("Enter salary: ");
        double salary = scanner.nextDouble();
        scanner.nextLine();
        System.out.print("Enter team name: ");
        String teamName = scanner.nextLine();
        System.out.print("Enter report date (yyyy-mm-dd): ");
        String reportDate = scanner.nextLine();
        System.out.print("Enter report description: ");
        String reportDescription = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewEmployee(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
            stmt.setString(1, employeeSSN);
```

PANDatabaseApp.java

```java
        try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewEmployee(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
            stmt.setString(1, employeeSSN);
            stmt.setString(2, name);
            stmt.setString(3, gender);
            stmt.setString(4, profession);
            stmt.setString(5, mailingAddress);
            stmt.setString(6, email);
            stmt.setString(7, phoneNumber);
            stmt.setInt(8, onMailingList);
            stmt.setString(9, maritalStatus);
            stmt.setDate(10, Date.valueOf(hireDate));
            stmt.setDouble(11, salary);
            stmt.setString(12, teamName);
            stmt.setDate(13, Date.valueOf(reportDate));
            stmt.setString(14, reportDescription);
            stmt.execute();
            System.out.println("Employee added and associated with team successfully.");
        } catch (SQLException e) {
            System.out.println("Error adding employee: " + e.getMessage());
        }
    }

    private static void enterExpense(Connection connection, Scanner scanner) {
        System.out.print("Enter employee SSN: ");
        String employeeSSN = scanner.nextLine();
        System.out.print("Enter expense date (yyyy-mm-dd): ");
        String expenseDate = scanner.nextLine();
        System.out.print("Enter amount: ");
        double amount = scanner.nextDouble();
        scanner.nextLine();
        System.out.print("Enter description: ");
        String description = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_EnterExpense(?, ?, ?, ?)}")) {
            stmt.setString(1, employeeSSN);
            stmt.setDate(2, Date.valueOf(expenseDate));
            stmt.setDouble(3, amount);
            stmt.setString(4, description);
            stmt.execute();
            System.out.println("Expense entry created successfully.");
        } catch (SQLException e) {
            System.out.println("Error adding expense: " + e.getMessage());
        }
    }

    private static void enterNewDonor(Connection connection, Scanner scanner) {
        System.out.print("Enter donor SSN: ");
        String donorSSN = scanner.nextLine();
        System.out.print("Enter donor name: ");
        String name = scanner.nextLine();
        System.out.print("Enter donor gender (M/F): ");
        String gender = scanner.nextLine();
        System.out.print("Enter donor profession: ");
        String profession = scanner.nextLine();
        System.out.print("Enter mailing address: ");
        String mailingAddress = scanner.nextLine();
        System.out.print("Enter email ID: ");
        String email = scanner.nextLine();
        System.out.print("Enter phone number: ");
        String phoneNumber = scanner.nextLine();
        System.out.print("Is the donor on the mailing list? (1 for yes, 0 for no): ");
        boolean onMailingList = scanner.nextInt() == 1;
        System.out.print("Is the donor anonymous? (1 for yes, 0 for no): ");
        boolean anonymous = scanner.nextInt() == 1;
        System.out.print("Enter the number of donations: ");
        int donationCount = scanner.nextInt();
```

PANDatabaseApp.java

```
        scanner.nextLine();

        for (int i = 0; i < donationCount; i++) {
            System.out.println("Entering details for donation " + (i + 1));
            System.out.print("Enter donation type (Cheque/Credit Card): ");
            String donationType = scanner.nextLine();
            System.out.print("Enter campaign name: ");
            String campaignName = scanner.nextLine();
            System.out.print("Enter donation date (yyyy-mm-dd): ");
            String date = scanner.nextLine();
            System.out.print("Enter amount: ");
            double amount = scanner.nextDouble();
            scanner.nextLine();

            try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewDonor(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
                stmt.setString(1, donorSSN);
                stmt.setString(2, name);
                stmt.setString(3, gender);
                stmt.setString(4, profession);
                stmt.setString(5, mailingAddress);
                stmt.setString(6, phoneNumber);
                stmt.setString(7, email);
                stmt.setBoolean(8, onMailingList);
                stmt.setBoolean(9, anonymous);
                stmt.setInt(10, 1);
                stmt.setString(11, donationType);
                stmt.setBigDecimal(12, BigDecimal.valueOf(amount));
                stmt.setDate(13, Date.valueOf(date));

                if ("Cheque".equalsIgnoreCase(donationType)) {
                    System.out.print("Enter cheque number: ");
                    stmt.setString(14, scanner.nextLine());
                    stmt.setNull(15, Types.CHAR);
                    stmt.setNull(16, Types.VARCHAR);
                    stmt.setNull(17, Types.DATE);
                } else if ("Credit Card".equalsIgnoreCase(donationType)) {
                    System.out.print("Enter card number: ");
                    stmt.setNull(14, Types.CHAR);
                    stmt.setString(15, scanner.nextLine());
                    System.out.print("Enter card type (e.g., Visa, MasterCard): ");
                    stmt.setString(16, scanner.nextLine());
                    System.out.print("Enter expiration date (yyyy-mm-dd): ");
                    stmt.setDate(17, Date.valueOf(scanner.nextLine()));
                } else {
                    System.out.println("Invalid donation type.");
                    continue;
                }

                stmt.setString(18, campaignName);
                stmt.execute();
                System.out.println("Donation added successfully.");
            } catch (SQLException e) {
                System.out.println("Error adding donation: " + e.getMessage());
            }
        }
    }




    private static void retrieveClientDoctor(Connection connection, Scanner scanner) {
        System.out.print("Enter client SSN: ");
        String clientSSN = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveClientDoctor(?)}")) {
            stmt.setString(1, clientSSN);
            ResultSet rs = stmt.executeQuery();
```

PANDatabaseApp.java

```
            stmt.setString(1, clientSSN);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("Doctor Name: " + rs.getString("doctor_name"));
                System.out.println("Doctor Phone Number: " + rs.getString("doctor_phone_number"));
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving doctor information: " + e.getMessage());
        }
    }

    private static void retrieveTotalExpensesByEmployee(Connection connection, Scanner scanner) {
        System.out.print("Enter start date (yyyy-mm-dd): ");
        String startDate = scanner.nextLine();
        System.out.print("Enter end date (yyyy-mm-dd): ");
        String endDate = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_TotalExpensesByEmployee(?, ?)}")) {
            stmt.setDate(1, Date.valueOf(startDate));
            stmt.setDate(2, Date.valueOf(endDate));
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("Employee SSN: " + rs.getString("employee_SSN"));
                System.out.println("Total Expenses: " + rs.getDouble("total_expenses"));
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving total expenses: " + e.getMessage());
        }
    }

    private static void listVolunteersForClientTeams(Connection connection, Scanner scanner) {
        System.out.print("Enter client SSN: ");
        String clientSSN = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_VolunteersForClientTeams(?)}")) {
            stmt.setString(1, clientSSN);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("Volunteer SSN: " + rs.getString("SSN"));
                System.out.println("Recent Training Location: " + rs.getString("recent_training_location"));
            }
        } catch (SQLException e) {
            System.out.println("Error listing volunteers: " + e.getMessage());
        }
    }

    private static void retrieveTeamsFoundedAfterDate(Connection connection, Scanner scanner) {
        System.out.print("Enter date (yyyy-mm-dd): ");
        String date = scanner.nextLine();

        try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveTeamsFoundedAfterDate(?)}")) {
            stmt.setDate(1, Date.valueOf(date));
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("Team Name: " + rs.getString("Name"));
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving teams: " + e.getMessage());
        }
    }

    private static void retrieveAllPeopleInfo(Connection connection) {
        try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveAllPeopleInfo()}")) {
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("Person Name: " + rs.getString("PersonName"));
```

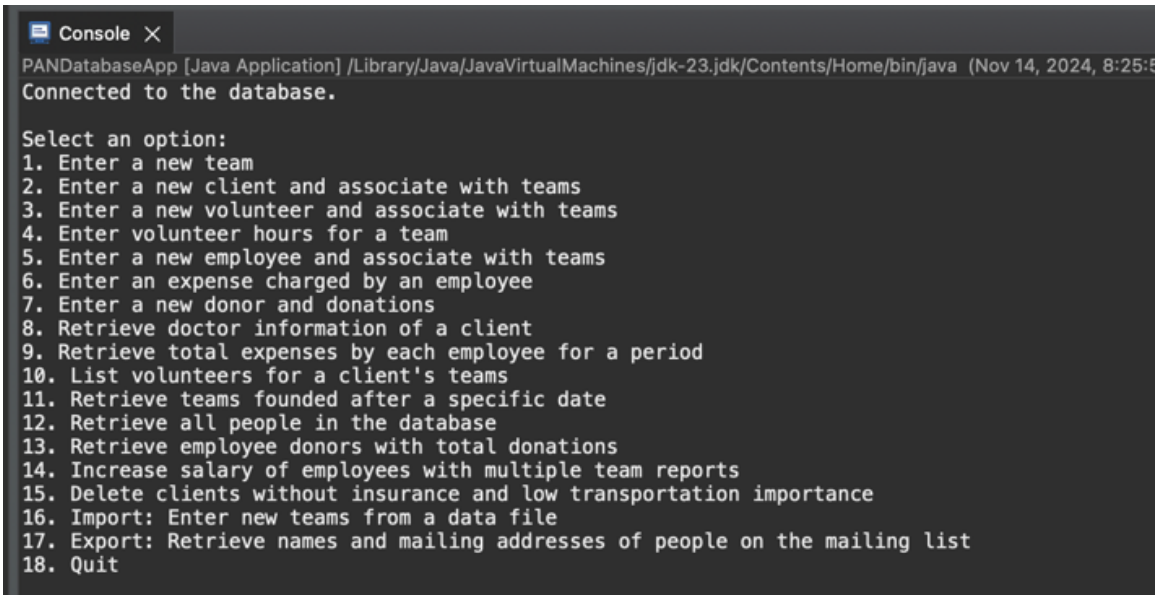
PANDatabaseApp.java

```
                System.out.println("Person Name: " + rs.getString("PersonName"));
                System.out.println("SSN: " + rs.getString("SSN"));
                System.out.println("Mailing Address: " + rs.getString("MailingAddress"));
                System.out.println("Phone Number: " + rs.getString("PhoneNumber"));
                System.out.println("Email ID: " + rs.getString("EmailID"));
                System.out.println("Emergency Contact Name: " + rs.getString("EmergencyContactName"));
                System.out.println("Emergency Contact Phone: " + rs.getString("EmergencyContactPhone"));
                System.out.println("Emergency Contact Relation: " + rs.getString("EmergencyContactRelation"));
                System.out.println("-----------------------------");
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving people: " + e.getMessage());
        }
    }



    private static void retrieveEmployeeDonors(Connection connection) {
        try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveEmployeeDonors()}")) {
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("SSN: " + rs.getString("SSN"));
                System.out.println("Is Anonymous: " + rs.getBoolean("IsAnonymous"));
                System.out.println("Total Credit Donations: $" + rs.getDouble("TotalCreditDonation"));
                System.out.println("-----------------------------");
            }
        } catch (SQLException e) {
            System.out.println("Error retrieving employee donors: " + e.getMessage());
        }
    }


    private static void increaseSalaryForMultiTeamEmployees(Connection connection) {
        try (CallableStatement stmt = connection.prepareCall("{call sp_IncreaseSalaryForMultiTeamEmployees()}")) {
            stmt.execute();
            System.out.println("Salary updated successfully for eligible employees.");
        } catch (SQLException e) {
            System.out.println("Error updating salary: " + e.getMessage());
        }
    }


    private static void DeleteUninsuredClientsWithLowTransportValue(Connection connection) {
        try (CallableStatement stmt = connection.prepareCall("{call sp_DeleteUninsuredClientsWithLowTransportValue()}")) {
            stmt.execute();
            System.out.println("Clients without insurance and low importance for transportation were deleted.");
        } catch (SQLException e) {
            System.out.println("Error deleting clients: " + e.getMessage());
        }
    }



    private static void importTeamsFromFile(Connection connection, Scanner scanner) {
        System.out.print("Enter input file name for importing teams: ");
        String fileName = scanner.nextLine();

        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] values = line.split(",");
                if (values.length == 3) {
                    String teamName = values[0].trim();
                    String teamType = values[1].trim();
                    String formationDate = values[2].trim();
```

PANDatabaseApp.java

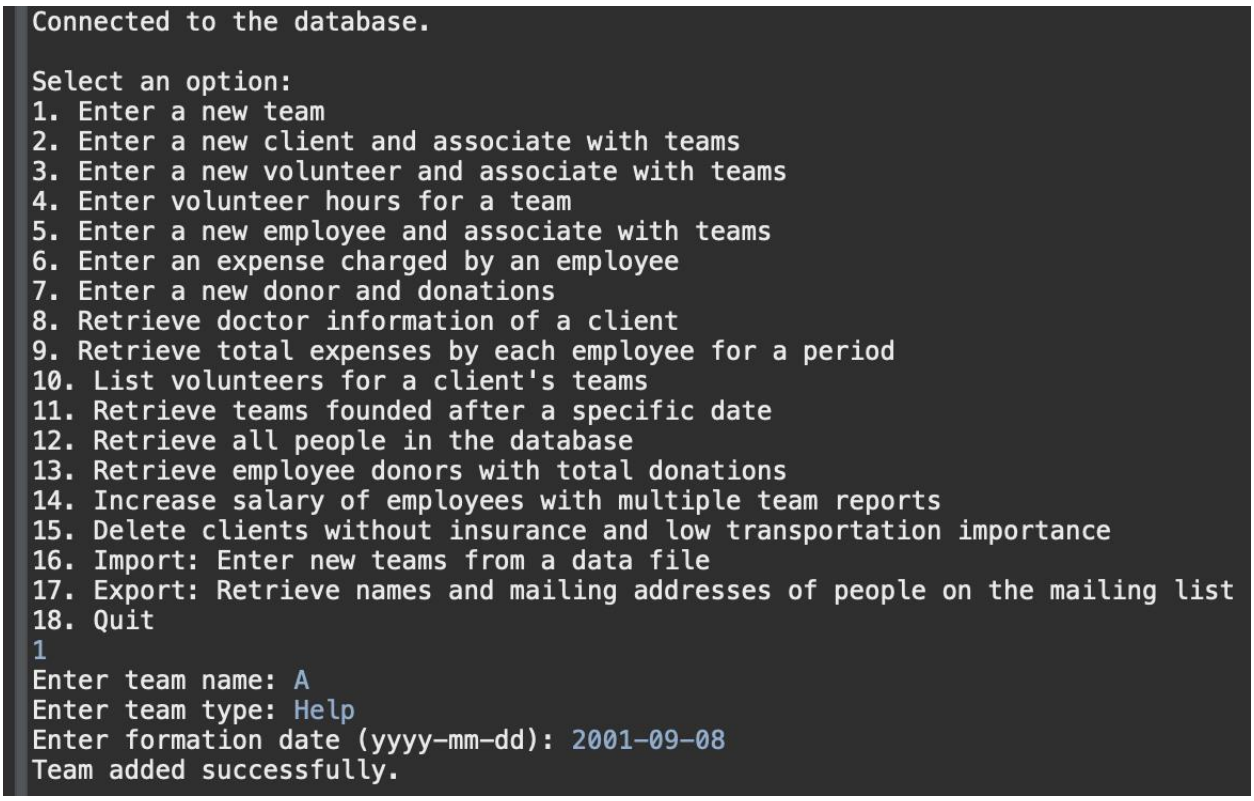
```
            stmt.execute();
            System.out.println("Clients without insurance and low importance for transportation were deleted.");
        } catch (SQLException e) {
            System.out.println("Error deleting clients: " + e.getMessage());
        }
    }



    private static void importTeamsFromFile(Connection connection, Scanner scanner) {
        System.out.print("Enter input file name for importing teams: ");
        String fileName = scanner.nextLine();

        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] values = line.split(",");
                if (values.length == 3) {
                    String teamName = values[0].trim();
                    String teamType = values[1].trim();
                    String formationDate = values[2].trim();

                    try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewTeam(?, ?, ?)}")) {
                        stmt.setString(1, teamName);
                        stmt.setString(2, teamType);
                        stmt.setDate(3, Date.valueOf(formationDate));
                        stmt.execute();
                        System.out.println("Team " + teamName + " added.");
                    } catch (SQLException e) {
                        System.out.println("Error adding team " + teamName + ": " + e.getMessage());
                    }
                } else {
                    System.out.println("Invalid line format: " + line);
                }
            }
            System.out.println("Import completed.");
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
    }

    private static void exportMailingListToFile(Connection connection, Scanner scanner) {
        System.out.print("Enter output file name for exporting mailing list: ");
        String fileName = scanner.nextLine();

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileName))) {
            String query = "SELECT name, mailing_address FROM Person WHERE on_mailing_list = 1";
            try (Statement stmt = connection.createStatement();
                 ResultSet rs = stmt.executeQuery(query)) {
                bw.write("Name,MailingAddress");
                bw.newLine();
                while (rs.next()) {
                    String name = rs.getString("name");
                    String mailingAddress = rs.getString("mailing_address");
                    bw.write(name + "," + mailingAddress);
                    bw.newLine();
                }
                System.out.println("Export completed.");
            } catch (SQLException e) {
                System.out.println("Database error: " + e.getMessage());
            }
        } catch (IOException e) {
            System.out.println("File writing error: " + e.getMessage());
        }
    }
}
```

Output –



Task 6. Java program Execution

6.1. Screenshots showing the testing of query 1



```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
1
Enter team name: B
Enter team type: Support
Enter formation date (yyyy-mm-dd): 2002-03-01
Team added successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
1
Enter team name: C
Enter team type: Care
Enter formation date (yyyy-mm-dd): 2004-01-09
Team added successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
1
Enter team name: D
Enter team type: Supply
Enter formation date (yyyy-mm-dd): 2004-06-10
Team added successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
1
Enter team name: E
Enter team type: Cultural
Enter formation date (yyyy-mm-dd): 2005-07-03
Team added successfully.
```

Output -

```
207    SELECT*FROM Team
208    SELECT*FROM Client
209    SELECT*FROM Person
210    SELECT*FROM Volunteer
211    SELECT*FROM Serves
212    SELECT*FROM Employee
213    SELECT*FROM Expense
214    SELECT*FROM Donor
215    SELECT*FROM [Check]
216    SELECT*FROM Credit_Card
```

Results Messages

	Name	type	date
1	A	Help	2001-09-08
2	B	Support	2002-03-01
3	C	Care	2004-01-09
4	D	Supply	2004-06-10
5	E	Cultural	2005-07-03

6.2. Screenshots showing the testing of query 2

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
2
Enter client SSN: 1001
Enter client name: Hari
Enter client gender (M/F): M
Enter client profession: Doctor
Enter client mailing address: 1900 N
Enter client phone number: 209876543
Enter client email ID: hari@gmail.com
Is the client on the mailing list? (1 for yes, 0 for no): 1
Enter doctor name: Sri
Enter date joined (yyyy-mm-dd): 2001-09-10
Enter doctor phone number: 398765492
Enter team name: A
Client added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
2
Enter client SSN: 2001
Enter client name: Sriya
Enter client gender (M/F): F
Enter client profession: Teacher
Enter client mailing address: 1300 S
Enter client phone number: 2864087
Enter client email ID: Sriya@gmail.com
Is the client on the mailing list? (1 for yes, 0 for no): 1
Enter doctor name: Shreya
Enter date joined (yyyy-mm-dd): 2009-03-01
Enter doctor phone number: 20019930
Enter team name: B
Client added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
2
Enter client SSN: 3001
Enter client name: Gre
Enter client gender (M/F): F
Enter client profession: Student
Enter client mailing address: 670 N
Enter client phone number: 348021
Enter client email ID: gre@gmail.com
Is the client on the mailing list? (1 for yes, 0 for no): 0
Enter doctor name: Whina
Enter date joined (yyyy-mm-dd): 2008-03-03
Enter doctor phone number: 389344
Enter team name: C
Client added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
2
Enter client SSN: 4001
Enter client name: Goku
Enter client gender (M/F): M
Enter client profession: Activist
Enter client mailing address: 400 E
Enter client phone number: 40987009
Enter client email ID: goku@gmail.com
Is the client on the mailing list? (1 for yes, 0 for no): 1
Enter doctor name: Vejeta
Enter date joined (yyyy-mm-dd): 2010-01-01
Enter doctor phone number: 2890093
Enter team name: D
Client added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
2
Enter client SSN: 5001
Enter client name: Dei
Enter client gender (M/F): M
Enter client profession: Professor
Enter client mailing address: 500 N
Enter client phone number: 387789
Enter client email ID: dei@gmail.com
Is the client on the mailing list? (1 for yes, 0 for no): 0
Enter doctor name: Geet
Enter date joined (yyyy-mm-dd): 2004-08-23
Enter doctor phone number: 88730092
Enter team name: E
Client added and associated with team successfully.
```

Output –

```
208    SELECT*FROM Client
209    SELECT*FROM Person
210    SELECT*FROM Volunteer
211    SELECT*FROM Serves
212    SELECT*FROM Employee
213    SELECT*FROM Expense
214    SELECT*FROM Donor
215    SELECT*FROM [Check]
216    SELECT*FROM Credit_Card
```

Results Messages

	SSN	doctor_name	date_joined	doctor_phone_number
1	1001	Sri	2001-09-10	398765492
2	2001	Shreya	2009-03-01	20019930
3	3001	Whina	2008-03-03	389344
4	4001	Vejeta	2010-01-01	2890093
5	5001	Geet	2004-08-23	88730092

6.3. Screenshots showing the testing of query 3

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
3
Enter volunteer SSN: 6001
Enter volunteer name: Deepika
Enter gender (M/F): F
Enter profession: lawyer
Enter mailing address: 300 Dr
Enter email: deepika@gmail.com
Enter phone number: 3098449
Is the volunteer on the mailing list? (1 for yes, 0 for no): 0
Enter date joined (yyyy-mm-dd): 2007-04-30
Enter recent training date (yyyy-mm-dd): 2020-09-10
Enter recent training location: Norman
Enter team name: A
Enter status (A for active, I for inactive): A
Enter hours worked: 32
Volunteer and person entries added successfully and associated with the team.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
3
Enter volunteer SSN: 7001
Enter volunteer name: Prya
Enter gender (M/F): F
Enter profession: designer
Enter mailing address: 390 K
Enter email: priya@gmail.com
Enter phone number: 30928874
Is the volunteer on the mailing list? (1 for yes, 0 for no): 0
Enter date joined (yyyy-mm-dd): 2009-07-30
Enter recent training date (yyyy-mm-dd): 2019-10-10
Enter recent training location: OKC
Enter team name: B
Enter status (A for active, I for inactive): A
Enter hours worked: 90
Volunteer and person entries added successfully and associated with the team.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
3
Enter volunteer SSN: 8001
Enter volunteer name: Rithu
Enter gender (M/F): F
Enter profession: Racer
Enter mailing address: 670 S
Enter email: rithu@gmail.com
Enter phone number: 30834001
Is the volunteer on the mailing list? (1 for yes, 0 for no): 1
Enter date joined (yyyy-mm-dd): 2010-01-04
Enter recent training date (yyyy-mm-dd): 2020-10-05
Enter recent training location: Kansas
Enter team name: C
Enter status (A for active, I for inactive): A
Enter hours worked: 50
Volunteer and person entries added successfully and associated with the team.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
3
Enter volunteer SSN: 9001
Enter volunteer name: Shubham
Enter gender (M/F): M
Enter profession: PHD
Enter mailing address: shubh@gmail.com
Enter email: shubham@gmail.com
Enter phone number: 2095403001
Is the volunteer on the mailing list? (1 for yes, 0 for no): 0
Enter date joined (yyyy-mm-dd): 2012-02-10
Enter recent training date (yyyy-mm-dd): 2018-04-04
Enter recent training location: St louis
Enter team name: D
Enter status (A for active, I for inactive): I
Enter hours worked: 201
Volunteer and person entries added successfully and associated with the team.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
3
Enter volunteer SSN: 6590
Enter volunteer name: Rohan
Enter gender (M/F): M
Enter profession: Engineer
Enter mailing address: 1500 Beau
Enter email: rohan@gmail.com
Enter phone number: 297001092
Is the volunteer on the mailing list? (1 for yes, 0 for no): 0
Enter date joined (yyyy-mm-dd): 2010-05-24
Enter recent training date (yyyy-mm-dd): 2020-08-30
Enter recent training location: Denver
Enter team name: E
Enter status (A for active, I for inactive): I
Enter hours worked: 40
Volunteer and person entries added successfully and associated with the team.
```

Output -

```
210    SELECT*FROM Volunteer
211    SELECT*FROM Serves
212    SELECT*FROM Employee
213    SELECT*FROM Expense
214    SELECT*FROM Donor
215    SELECT*FROM [Check]
216    SELECT*FROM Credit_Card
```

Results Messages

	SSN	date_joined	recent_training_date	recent_training_location
1	6001	2007-04-30	2020-09-10	Norman
2	6590	2010-05-24	2020-08-30	Denver
3	7001	2009-07-30	2019-10-10	OKC
4	8001	2010-01-04	2020-10-05	Kansas
5	9001	2012-02-10	2018-04-04	St louis

6.4. Screenshots showing the testing of query 4

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
4
Enter volunteer SSN: 6001
Enter team name: A
Enter additional hours worked: 20
Enter month: 3
Enter status (A for active, I for inactive): I
Volunteer hours updated successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
4
Enter volunteer SSN: 6590
Enter team name: B
Enter additional hours worked: 30
Enter month: 2
Enter status (A for active, I for inactive): A
Volunteer hours updated successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
4
Enter volunteer SSN: 7001
Enter team name: C
Enter additional hours worked: 59
Enter month: 2
Enter status (A for active, I for inactive): A
Volunteer hours updated successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
4
Enter volunteer SSN: 8001
Enter team name: D
Enter additional hours worked: 19
Enter month: 8
Enter status (A for active, I for inactive): A
Volunteer hours updated successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
4
Enter volunteer SSN: 9001
Enter team name: E
Enter additional hours worked: 72
Enter month: 6
Enter status (A for active, I for inactive): I
Volunteer hours updated successfully.
```

Output -

```
211    SELECT*FROM Serves
212    SELECT*FROM Employee
213    SELECT*FROM Expense
214    SELECT*FROM Donor
215    SELECT*FROM [Check]
216    SELECT*FROM Credit_Card
```

Results Messages

	Volunteer_SSN	Team_Name	active_inactive	hours_worked	month
1	6001	A	I	20	3
2	6590	B	A	30	2
3	7001	C	A	59	2
4	8001	D	A	19	8
5	9001	E	I	72	6

6.5. Screenshots showing the testing of query 5

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
5
Enter employee SSN: 1234
Enter employee name: Ramya
Enter employee gender (M/F): F
Enter employee profession: Architect
Enter mailing address: 398 H
Enter email ID: ramya@gmail.com
Enter phone number: 2390090
Is the employee on the mailing list? (1 for yes, 0 for no): 1
Enter marital status: Single
Enter hire date (yyyy-mm-dd): 2015-02-01
Enter salary: 20000
Enter team name: A
Enter report date (yyyy-mm-dd): 2016-01-02
Enter report description: Financial queries
Employee added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
5
Enter employee SSN: 6789
Enter employee name: Tiyona
Enter employee gender (M/F): F
Enter employee profession: Caterer
Enter mailing address: 319 H
Enter email ID: Tiyona@gmail.com
Enter phone number: 398740
Is the employee on the mailing list? (1 for yes, 0 for no): 1
Enter marital status: Married
Enter hire date (yyyy-mm-dd): 2018-02-01
Enter salary: 39000
Enter team name: C
Enter report date (yyyy-mm-dd): 2020-01-01
Enter report description: Food queries
Employee added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
5
Enter employee SSN: 7659
Enter employee name: Tara
Enter employee gender (M/F): F
Enter employee profession: Fashion
Enter mailing address: 700 South
Enter email ID: tara@gmail.com
Enter phone number: 399992990
Is the employee on the mailing list? (1 for yes, 0 for no): 0
Enter marital status: Single
Enter hire date (yyyy-mm-dd): 2017-01-20
Enter salary: 40000
Enter team name: D
Enter report date (yyyy-mm-dd): 2019-03-18
Enter report description: Clothes related queries
Employee added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
5
Enter employee SSN: 3914
Enter employee name: Hruth
Enter employee gender (M/F): M
Enter employee profession: Driver
Enter mailing address: 418 North Dr
Enter email ID: hruth@gmail.com
Enter phone number: 98700723
Is the employee on the mailing list? (1 for yes, 0 for no): 1
Enter marital status: Married
Enter hire date (yyyy-mm-dd): 2016-10-25
Enter salary: 52000
Enter team name: D
Enter report date (yyyy-mm-dd): 2020-02-01
Enter report description: Car issues
Employee added and associated with team successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
5
Enter employee SSN: 4321
Enter employee name: Harsh
Enter employee gender (M/F): M
Enter employee profession: Actor
Enter mailing address: 309 I
Enter email ID: harsh@gmail.com
Enter phone number: 508909209
Is the employee on the mailing list? (1 for yes, 0 for no): 0
Enter marital status: Married
Enter hire date (yyyy-mm-dd): 2012-03-04
Enter salary: 3000
Enter team name: B
Enter report date (yyyy-mm-dd): 2015-02-01
Enter report description: Cultural queries
Employee added and associated with team successfully.
```

Output –

```
212    SELECT*FROM Employee
213    SELECT*FROM Expense
214    SELECT*FROM Donor
215    SELECT*FROM [Check]
216    SELECT*FROM Credit_Card
```

Results Messages

	SSN	marital_status	hire_date	salary
1	1234	Single	2015-02-01	20000.00
2	3914	Married	2016-10-25	52000.00
3	4321	Married	2012-03-04	3000.00
4	6789	Married	2018-02-01	39000.00
5	7659	Single	2017-01-20	40000.00

6.6. Screenshots showing the testing of query 6

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
6
Enter employee SSN: 1234
Enter expense date (yyyy-mm-dd): 2016-12-01
Enter amount: 200
Enter description: Food
Expense entry created successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
6
Enter employee SSN: 3914
Enter expense date (yyyy-mm-dd): 2017-11-01
Enter amount: 290
Enter description: Groceries
Expense entry created successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
6
Enter employee SSN: 4321
Enter expense date (yyyy-mm-dd): 2013-02-02
Enter amount: 400
Enter description: Rent
Expense entry created successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
6
Enter employee SSN: 6789
Enter expense date (yyyy-mm-dd): 2019-05-16
Enter amount: 800
Enter description: Utilities
Expense entry created successfully.
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
6
Enter employee SSN: 7659
Enter expense date (yyyy-mm-dd): 2020-10-13
Enter amount: 500
Enter description: Fuel
Expense entry created successfully.
```

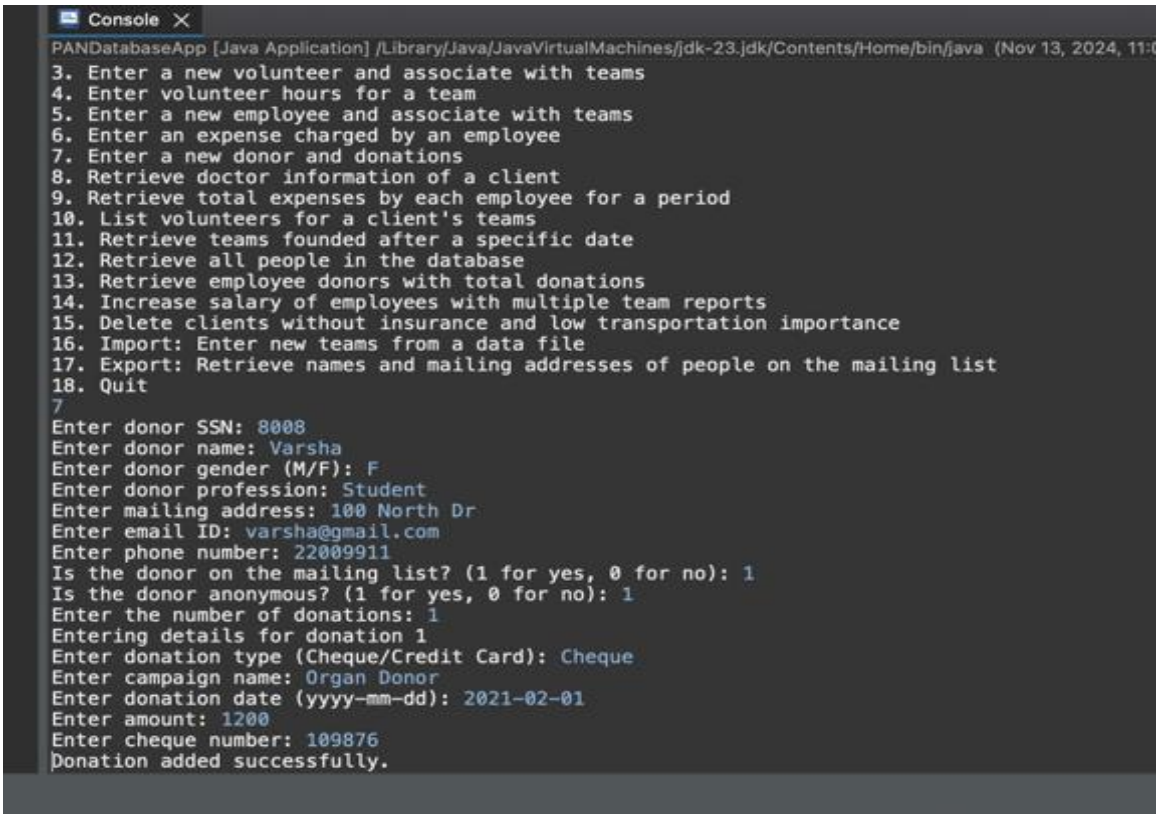
Output -

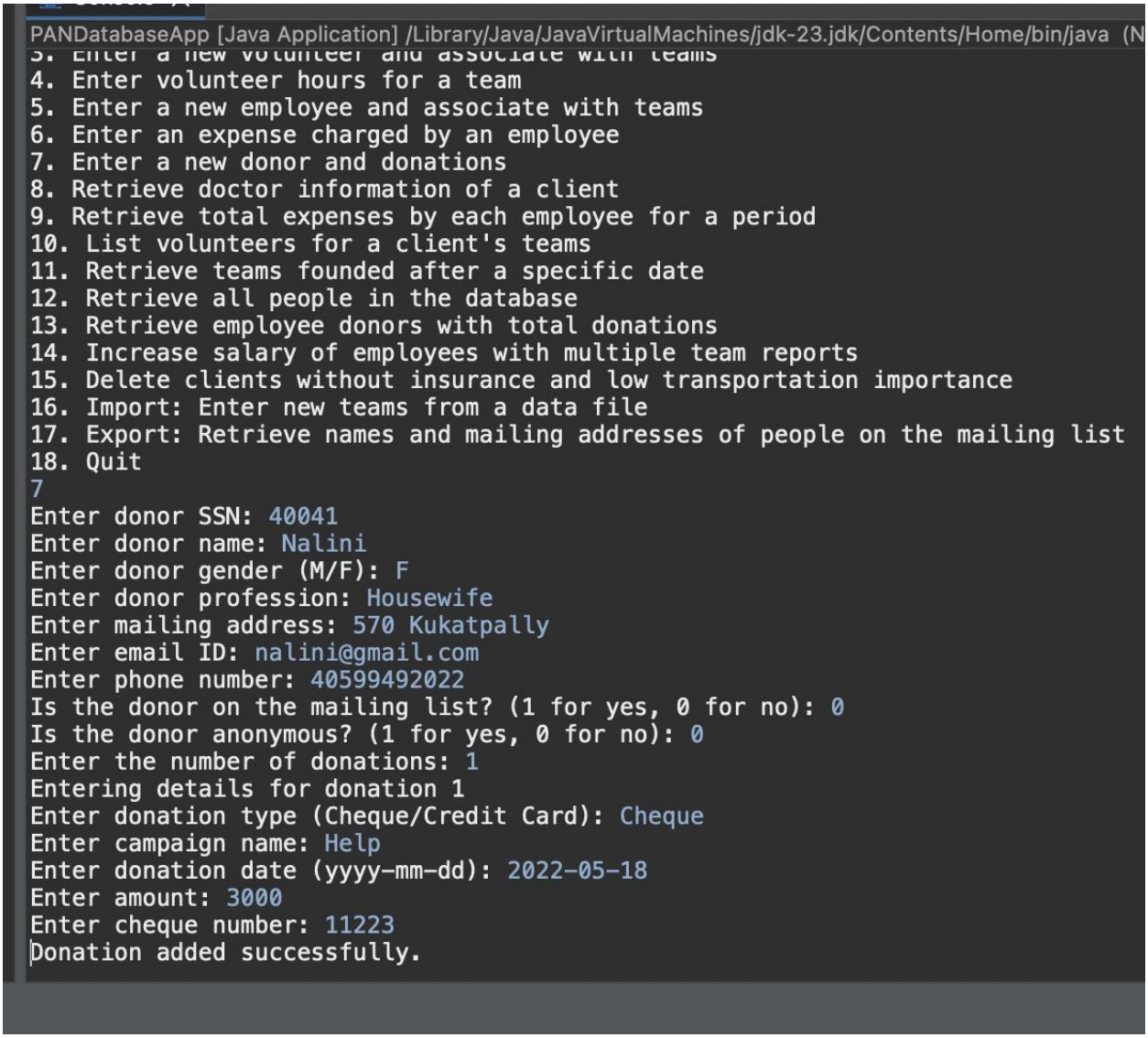
```
213    SELECT*FROM Expense
214    SELECT*FROM Donor
215    SELECT*FROM [Check]
216    SELECT*FROM Credit_Card
```

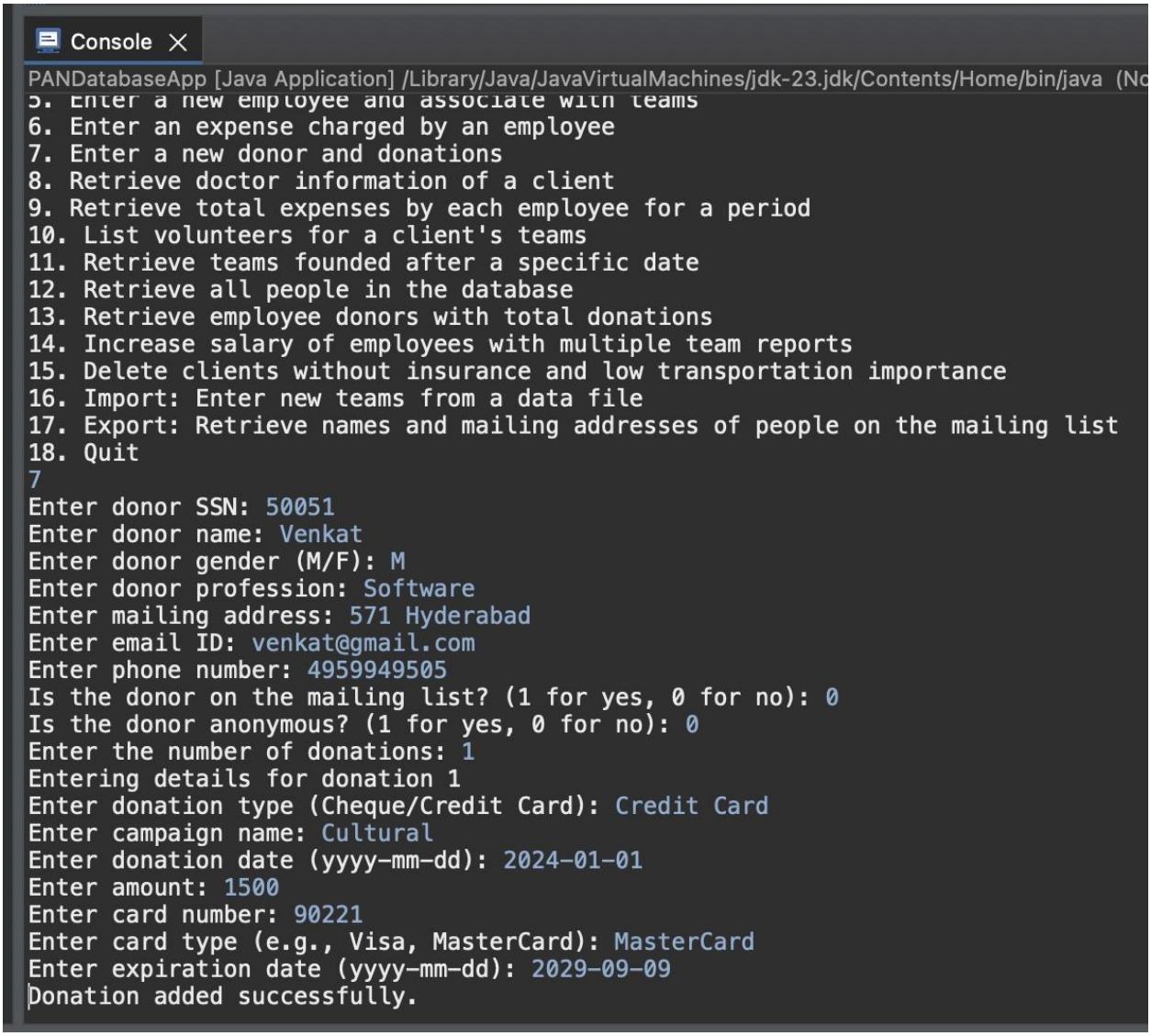
Results Messages

	employee_SSN	date	amount	description
1	1234	2016-12-01	200.00	Food
2	3914	2017-11-01	290.00	Groceries
3	4321	2013-02-02	400.00	Rent
4	6789	2019-05-16	800.00	Utilities
5	7659	2020-10-13	500.00	Fuel

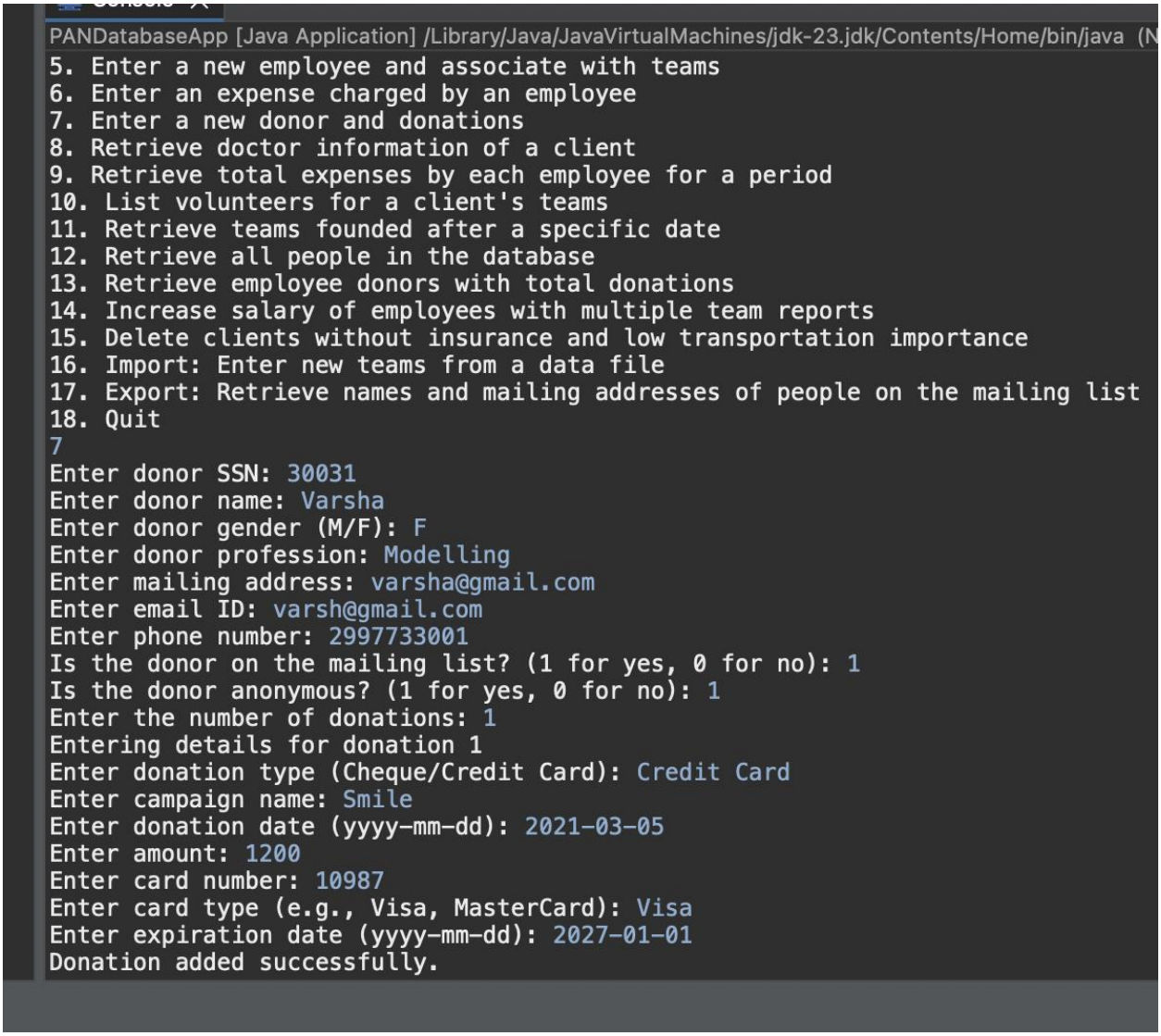
6.7. Screenshots showing the testing of query 7




PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (N
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
7
Enter donor SSN: 40041
Enter donor name: Nalini
Enter donor gender (M/F): F
Enter donor profession: Housewife
Enter mailing address: 570 Kukatpally
Enter email ID: nalini@gmail.com
Enter phone number: 40599492022
Is the donor on the mailing list? (1 for yes, 0 for no): 0
Is the donor anonymous? (1 for yes, 0 for no): 0
Enter the number of donations: 1
Entering details for donation 1
Enter donation type (Cheque/Credit Card): Cheque
Enter campaign name: Help
Enter donation date (yyyy-mm-dd): 2022-05-18
Enter amount: 3000
Enter cheque number: 11223
Donation added successfully.


Console
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (No
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
7
Enter donor SSN: 50051
Enter donor name: Venkat
Enter donor gender (M/F): M
Enter donor profession: Software
Enter mailing address: 571 Hyderabad
Enter email ID: venkat@gmail.com
Enter phone number: 4959949505
Is the donor on the mailing list? (1 for yes, 0 for no): 0
Is the donor anonymous? (1 for yes, 0 for no): 0
Enter the number of donations: 1
Entering details for donation 1
Enter donation type (Cheque/Credit Card): Credit Card
Enter campaign name: Cultural
Enter donation date (yyyy-mm-dd): 2024-01-01
Enter amount: 1500
Enter card number: 90221
Enter card type (e.g., Visa, MasterCard): MasterCard
Enter expiration date (yyyy-mm-dd): 2029-09-09
Donation added successfully.

```
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
7
Enter donor SSN: 50981
Enter donor name: Anup
Enter donor gender (M/F): M
Enter donor profession: Doctor
Enter mailing address: 1900 Kukat
Enter email ID: anup@gmail.com
Enter phone number: 98922034
Is the donor on the mailing list? (1 for yes, 0 for no): 1
Is the donor anonymous? (1 for yes, 0 for no): 0
Enter the number of donations: 1
Entering details for donation 1
Enter donation type (Cheque/Credit Card): Cheque
Enter campaign name: Healthy Heart
Enter donation date (yyyy-mm-dd): 2019-02-05
Enter amount: 2009
Enter cheque number: 398711
Donation added successfully.
```


PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (N
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
7
Enter donor SSN: 30031
Enter donor name: Varsha
Enter donor gender (M/F): F
Enter donor profession: Modelling
Enter mailing address: varsha@gmail.com
Enter email ID: varsh@gmail.com
Enter phone number: 2997733001
Is the donor on the mailing list? (1 for yes, 0 for no): 1
Is the donor anonymous? (1 for yes, 0 for no): 1
Enter the number of donations: 1
Entering details for donation 1
Enter donation type (Cheque/Credit Card): Credit Card
Enter campaign name: Smile
Enter donation date (yyyy-mm-dd): 2021-03-05
Enter amount: 1200
Enter card number: 10987
Enter card type (e.g., Visa, MasterCard): Visa
Enter expiration date (yyyy-mm-dd): 2027-01-01
Donation added successfully.

Output –

```
214  SELECT*FROM Donor
215  SELECT*FROM [Check]
216  SELECT*FROM Credit_Card
```

Results Messages

	SSN	anonymous
1	30031	1
2	40041	0
3	50051	0
4	50981	0
5	8008	1

	Donor_SSN	date	check_number	campaign_name	type	amount
1	40041	2022-05-18	11223	Help	Cheque	3000.00
2	50981	2019-02-05	398711	Healthy Heart	Cheque	2009.00
3	8008	2021-02-01	109876	Organ Donor	Cheque	1200.00

	Donor_SSN	date	card_number	card_type	Amount	type	campaign_name	expiration_date
1	30031	2021-03-05	10987	Visa	1200.00	Credit Card	Smile	2027-01-01
2	50051	2024-01-01	90221	MasterCard	1500.00	Credit Card	Cultural	2029-09-09

6.8. Screenshots showing the testing of query 8

```
Console
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (N
Is the donor on the mailing list? (1 for yes, 0 for no): 1
Is the donor anonymous? (1 for yes, 0 for no): 0
Enter the number of donations: 1
Entering details for donation 1
Enter donation type (Cheque/Credit Card): Cheque
Enter campaign name: Healthy Heart
Enter donation date (yyyy-mm-dd): 2019-02-05
Enter amount: 2009
Enter cheque number: 398711
Donation added successfully.

Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
8
Enter client SSN: 1001
Doctor Name: Sri
Doctor Phone Number: 398765492
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
8
Enter client SSN: 2001
Doctor Name: Shreya
Doctor Phone Number: 20019930
```

6.9. Screenshots showing the testing of query 9

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
9
Enter start date (yyyy-mm-dd): 2015-01-01
Enter end date (yyyy-mm-dd): 2024-01-01
Employee SSN: 6789
Total Expenses: 800.0
Employee SSN: 7659
Total Expenses: 500.0
Employee SSN: 3914
Total Expenses: 290.0
Employee SSN: 1234
Total Expenses: 200.0
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
9
Enter start date (yyyy-mm-dd): 2012-01-01
Enter end date (yyyy-mm-dd): 2020-01-01
Employee SSN: 6789
Total Expenses: 800.0
Employee SSN: 4321
Total Expenses: 400.0
Employee SSN: 3914
Total Expenses: 290.0
Employee SSN: 1234
Total Expenses: 200.0
```

6.10. Screenshots showing the testing of query 10

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit

10
Enter client SSN: 3001
Volunteer SSN: 7001
Recent Training Location: OKC
```

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
10
Enter client SSN: 5001
Volunteer SSN: 9001
Recent Training Location: St louis
```

6.11. Screenshots showing the testing of query 11

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
11
Enter date (yyyy-mm-dd): 2005-01-01
Team Name: E
```

```
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
11
Enter date (yyyy-mm-dd): 2001-01-01
Team Name: A
Team Name: B
Team Name: C
Team Name: D
Team Name: E
```

6.12. Screenshots showing the testing of query 12

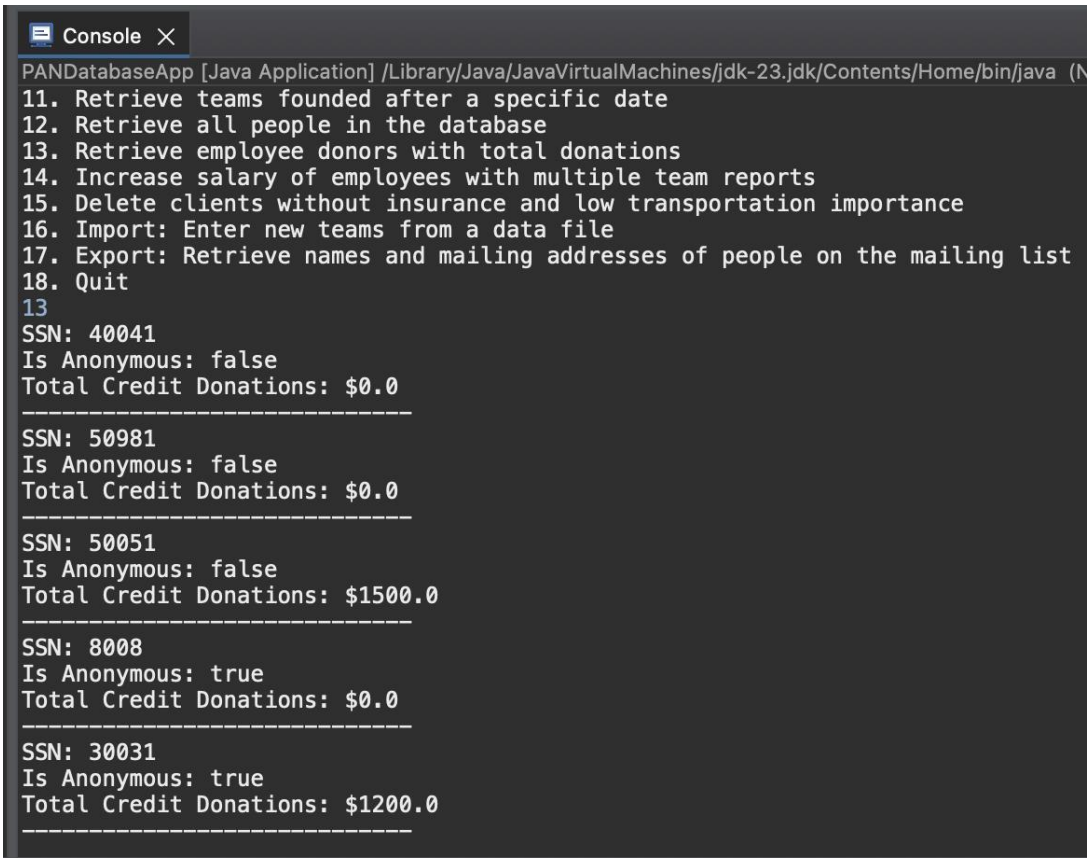
```
------------------------------
Person Name: Gre
SSN: 3001
Mailing Address: 670 N
Phone Number: 348021
Email ID: gre@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Varsha
SSN: 30031
Mailing Address: varsha@gmail.com
Phone Number: 2997733001
Email ID: varsh@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Hruth
SSN: 3914
Mailing Address: 418 North Dr
Phone Number: 98700723
Email ID: hruth@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Goku
SSN: 4001
Mailing Address: 400 E
Phone Number: 40987009
Email ID: goku@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Nalini
SSN: 40041
Mailing Address: 570 Kukatpally
Phone Number: 40599492022
Email ID: nalini@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Harsh
SSN: 4321
Mailing Address: 309 I
Phone Number: 508909209
Email ID: harsh@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
```

```
----------------------------
Person Name: Dei
SSN: 5001
Mailing Address: 500 N
Phone Number: 387789
Email ID: dei@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Venkat
SSN: 50051
Mailing Address: 571 Hyderabad
Phone Number: 4959949505
Email ID: venkat@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Anup
SSN: 50981
Mailing Address: 1900 Kukat
Phone Number: 98922034
Email ID: anup@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Nitin
SSN: 5555
Mailing Address: 1700 B
Phone Number: 3086744
Email ID: nitin@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Deepika
SSN: 6001
Mailing Address: 300 Dr
Phone Number: 3098449
Email ID: deepika@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Rohan
SSN: 6590
Mailing Address: 1500 Beau
Phone Number: 297001092
Email ID: rohan@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
```

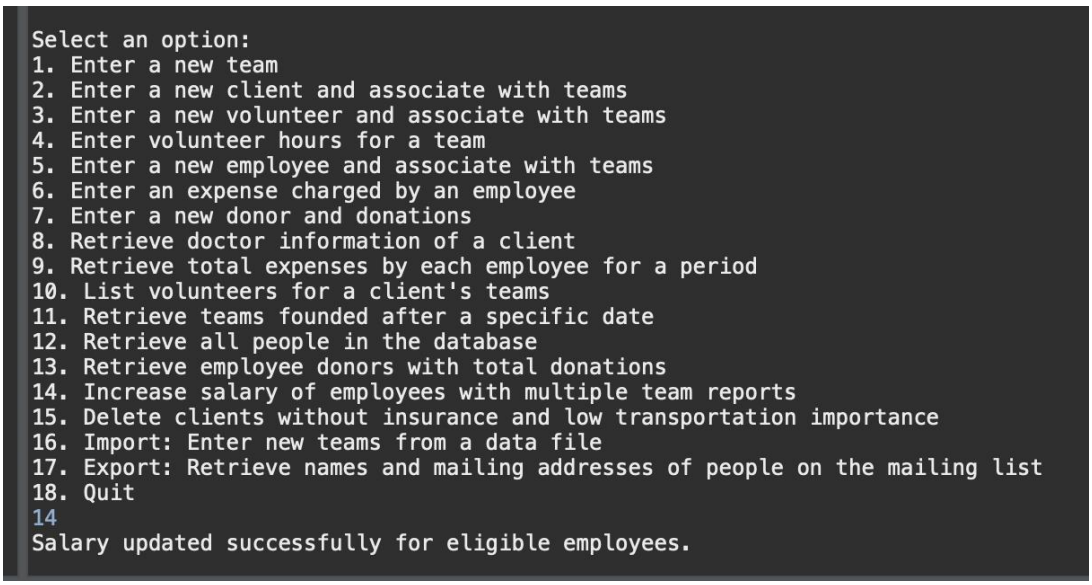
```
------------------------------
Person Name: Nitin
SSN: 6666
Mailing Address: 500 Fortworth
Phone Number: 901800234
Email ID: nitin@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Tiyona
SSN: 6789
Mailing Address: 319 H
Phone Number: 398740
Email ID: Tiyona@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Prya
SSN: 7001
Mailing Address: 390 K
Phone Number: 30928874
Email ID: priya@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Tara
SSN: 7659
Mailing Address: 700 South
Phone Number: 399992990
Email ID: tara@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Rithu
SSN: 8001
Mailing Address: 670 S
Phone Number: 30834001
Email ID: rithu@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
Person Name: Varsha
SSN: 8008
Mailing Address: 100 North Dr
Phone Number: 22009911
Email ID: varsha@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
------------------------------
```

```
----------------------------
Person Name: Rithu
SSN: 8001
Mailing Address: 670 S
Phone Number: 30834001
Email ID: rithu@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Varsha
SSN: 8008
Mailing Address: 100 North Dr
Phone Number: 22009911
Email ID: varsha@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
Person Name: Shubham
SSN: 9001
Mailing Address: shubh@gmail.com
Phone Number: 2095403001
Email ID: shubham@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
----------------------------
```

6.13. Screenshots showing the testing of query 13



6.14. Screenshots showing the testing of query 14



Output –

```
568    SELECT * FROM Employee;
569
570
```

Results Messages

	SSN	marital_status	hire_date	salary
1	1234	Single	2015-02-01	22000.00
2	3914	Married	2016-10-25	57200.00
3	4321	Married	2012-03-04	3000.00
4	6789	Married	2018-02-01	39000.00
5	7659	Single	2017-01-20	40000.00

6.15. Screenshots showing the testing of query 15

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
15
Clients without insurance and low importance for transportation were deleted.
```

Output -

```
SELECT*FROM Needs
SELECT*FROM Insurance_Policy
SELECT*FROM Client
```

Results | Messages

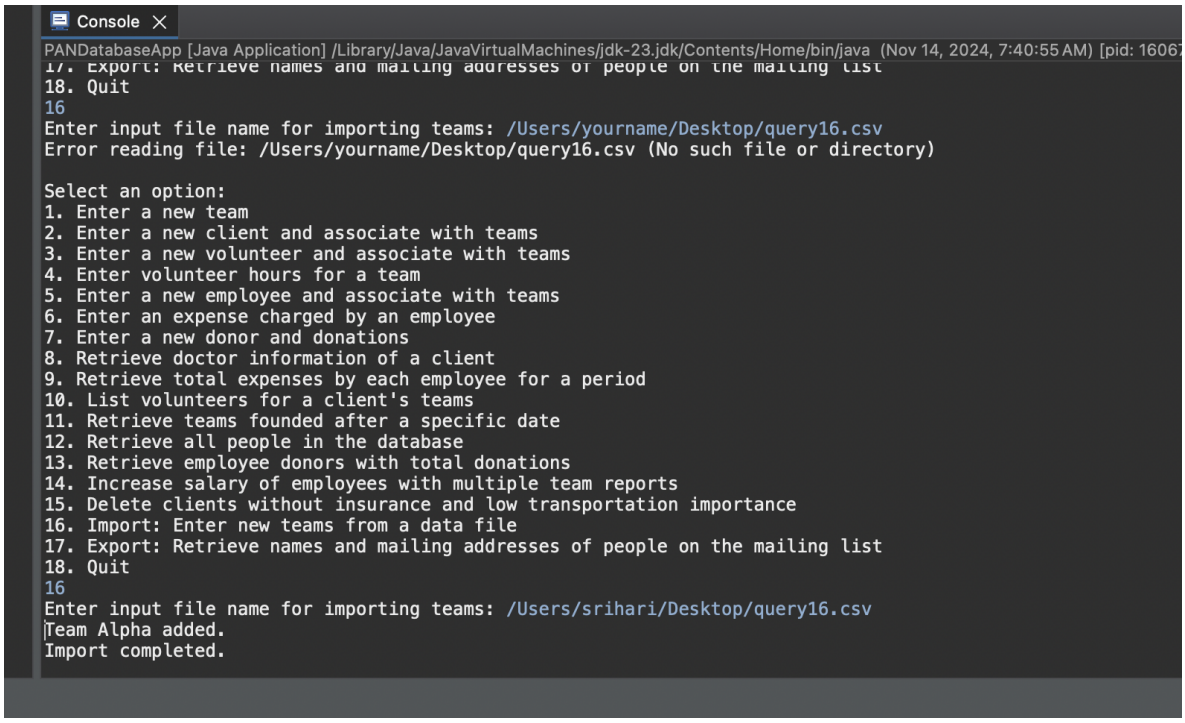
	type	importance	client_SSN
1	Medical	1	4001
2	Transportation	2	3001
3	Transportation	2	4001
4	Housing	3	4001
5	Housing	4	3001
6	Transportation	4	5001
7	Food	5	3001
8	Food	5	5001

	policy_id	provider_name	provider_address	type
1	POL1001	HealthFirst	123 Main St	Health
2	POL2001	AutoSecure	456 Oak St	Auto
3	POL3001	SafeHome	789 Pine St	Home
4	POL4001	LifeCare	321 Elm St	Life
5	POL5001	HealthPlus	654 Maple St	Health

Results grid

	SSN	doctor_name	date_joined	doctor_phone_number
1	3001	Whina	2008-03-03	389344
2	4001	Vejeta	2010-01-01	2890093
3	5001	Geet	2004-08-23	88730092

6.17. Screenshots showing the testing of query 16 and 17



Output -

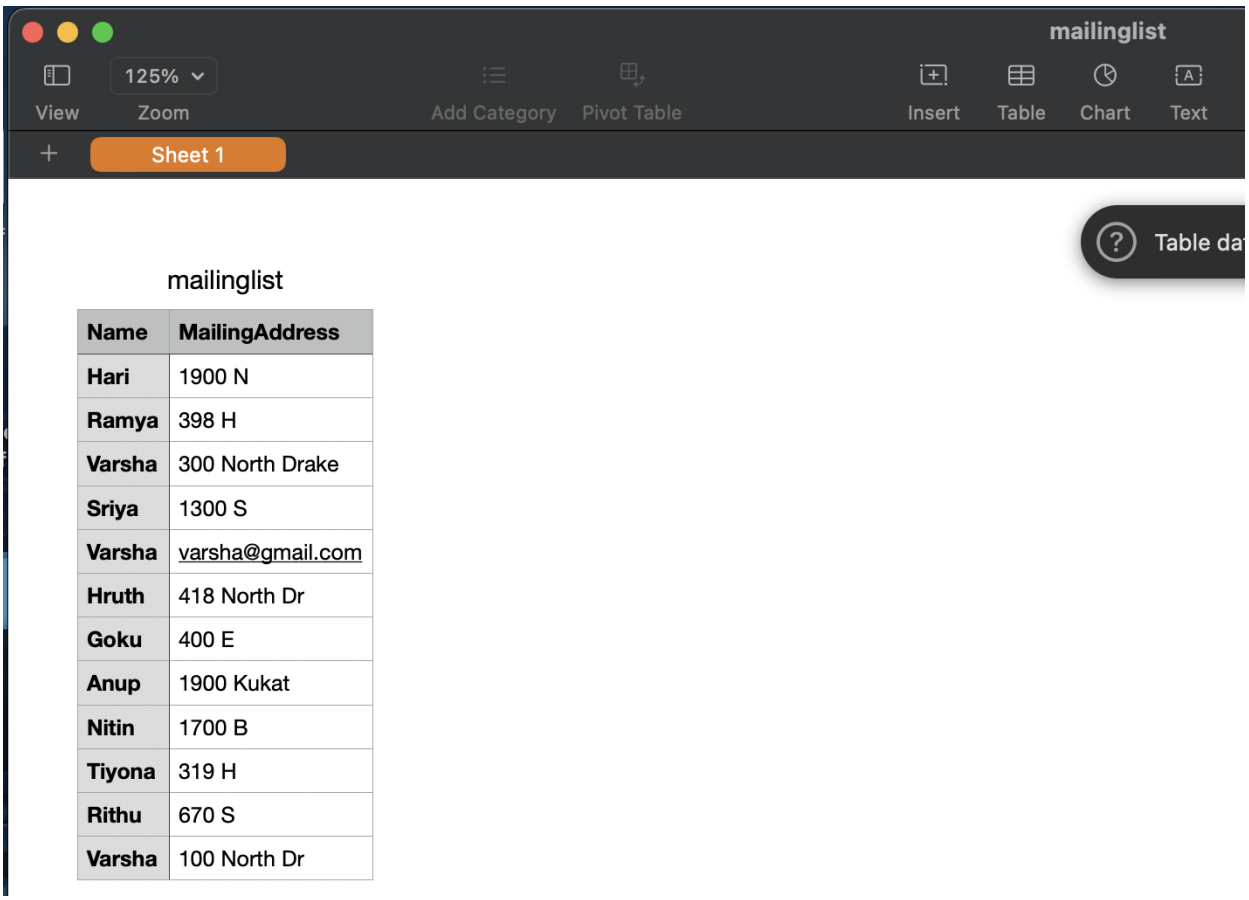
Results Messages

	Name	type	date
1	A	Help	2001-09-08
2	Alpha	Research	2022-02-01
3	B	Support	2002-03-01
4	C	Care	2004-01-09
5	D	Supply	2004-06-10
6	E	Cultural	2005-07-03
7	TeamA	Type1	2023-01-01
8	TeamB	Type2	2023-02-01

Screenshots showing the testing of query 17

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
17
Enter output file name for exporting mailing list: /Users/srihari/Desktop/mailinglist.csv
Export completed.
```

Output -

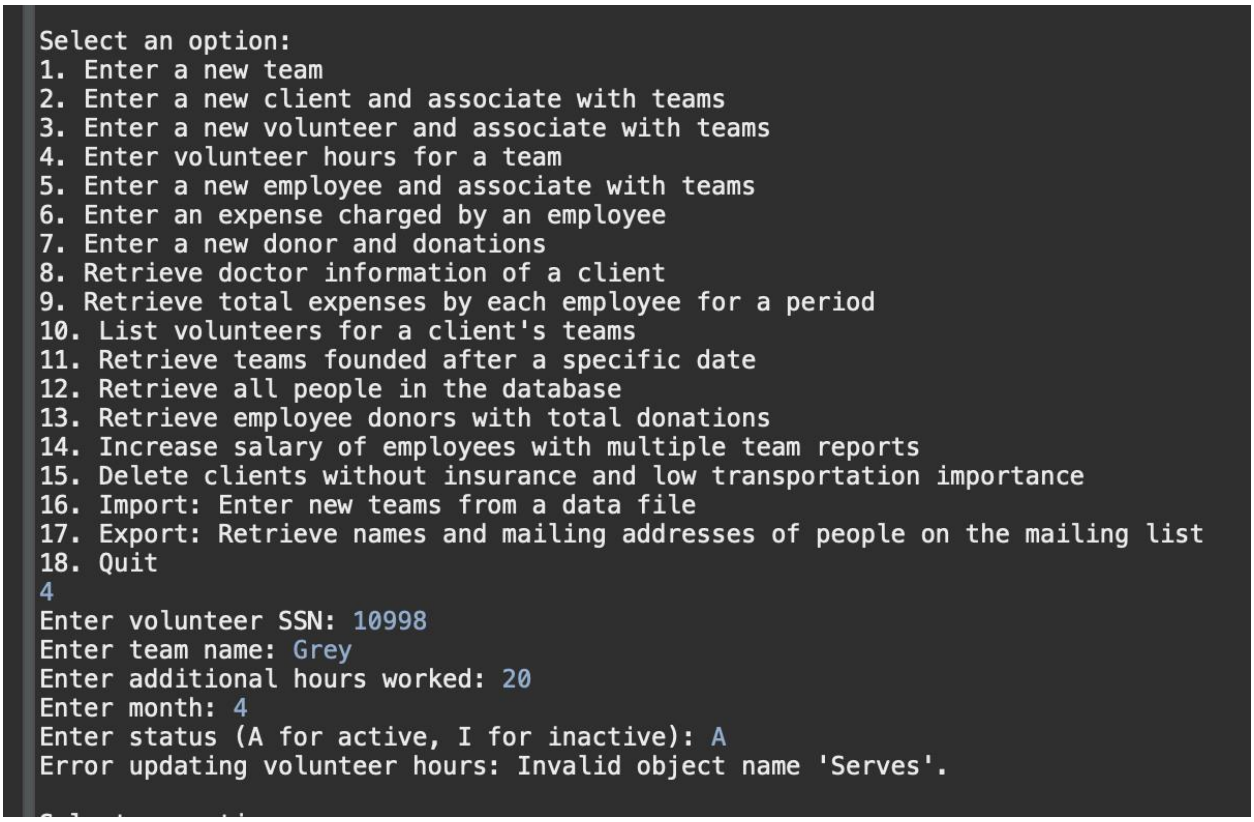


mailinglist

Name	MailingAddress
Hari	1900 N
Ramya	398 H
Varsha	300 North Drake
Sriya	1300 S
Varsha	varsha@gmail.com
Hruth	418 North Dr
Goku	400 E
Anup	1900 Kukat
Nitin	1700 B
Tiyona	319 H
Rithu	670 S
Varsha	100 North Dr

6.17 - Errors – Screenshots showing the testing of three types of errors are as follows;

1 – The error message " Invalid object name 'serves' " indicates that SQL Server cannot find a table or view named Serves in the database due to Incorrect SSN entered by user for the volunteer's SSN.



2- The error message indicates that the Java program expects a numeric input for the "Enter amount" parameter field but instead the user has entered a numeric string which is "abcd"

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
6
Enter employee SSN: 1234
Enter expense date (yyyy-mm-dd): 2001-03-04
Enter amount: abcd
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:964)
        at java.base/java.util.Scanner.next(Scanner.java:1619)
        at java.base/java.util.Scanner.nextDouble(Scanner.java:2590)
        at PANDatabaseApp.enterExpense(PANDatabaseApp.java:288)
        at PANDatabaseApp.main(PANDatabaseApp.java:54)
```

3- The error in the Java code indicates that the date format provided in the "Enter Formation date" field does not match the expected format. Here, the date is required to be of format (yyyy-mm-dd) but the user has given the incorrect fromat of (dd-mm-yyyy).

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
1
Enter team name: XYZ
Enter team type: team xyz
Enter formation date (yyyy-mm-dd): 20-09-2002
Exception in thread "main" java.lang.IllegalArgumentException
        at java.sql/java.sql.Date.valueOf(Date.java:141)
        at PANDatabaseApp.enterNewTeam(PANDatabaseApp.java:87)
        at PANDatabaseApp.main(PANDatabaseApp.java:49)
```

6.18. Screenshots showing the testing of the quit option

```
Select an option:
1. Enter a new team
2. Enter a new client and associate with teams
3. Enter a new volunteer and associate with teams
4. Enter volunteer hours for a team
5. Enter a new employee and associate with teams
6. Enter an expense charged by an employee
7. Enter a new donor and donations
8. Retrieve doctor information of a client
9. Retrieve total expenses by each employee for a period
10. List volunteers for a client's teams
11. Retrieve teams founded after a specific date
12. Retrieve all people in the database
13. Retrieve employee donors with total donations
14. Increase salary of employees with multiple team reports
15. Delete clients without insurance and low transportation importance
16. Import: Enter new teams from a data file
17. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit

18
Exiting the application.
```