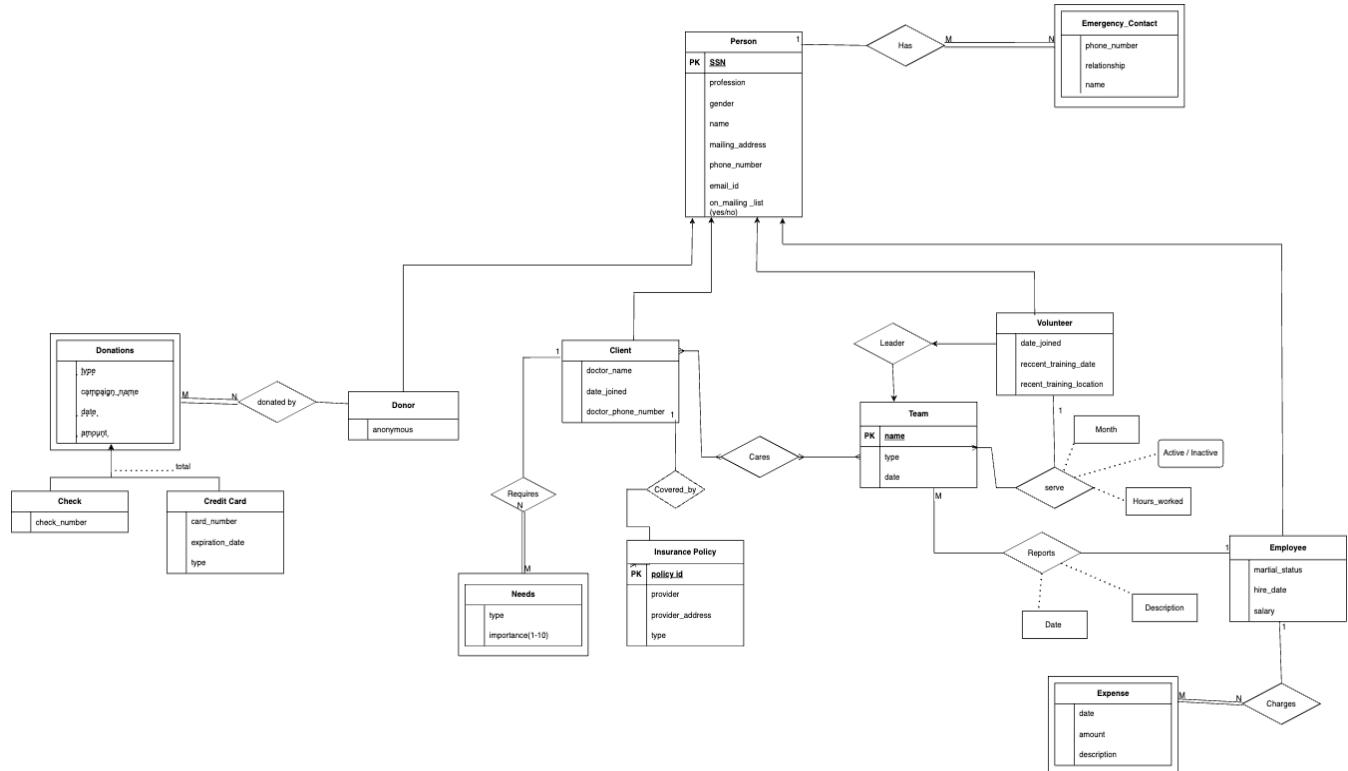


Course name : **Database Management Systems**
Course Number : **DSA 4513-001**
Section Number : **001**
Semester and Year : **Fall 2024**
Instructor's Name : **Dr. Le Gruenwald**
Student Name : **L Srihari Boppana**
Student Id : **113638264**
Email address : **l.srihari.boppana-1@ou.edu**
Title : **PAN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

Tasks Performed	Page Number
Task 1. ER Diagram	3
Task 2. Relational Database Schemas	4
Task 3.	
3.1. Discussion of storage structures for tables	5 – 7
3.2. Discussion of storage structures for tables (Azure SQL Database)	8
Task 4. SQL statements and screenshots showing the creation of tables in Azure SQL Database	9-17
Task 5.	
5.1 SQL statements (and Transact SQL stored procedures, if any) Implementing all queries (1-15 and error checking)	18 - 36
5.2 The Java source program and screenshots showing its successful compilation	37 - 46
Task 6. Java program Execution	
6.1. Screenshots showing the testing of query 1	46 – 50
6.2. Screenshots showing the testing of query 2	51 – 56
6.3. Screenshots showing the testing of query 3	57 - 62
6.4. Screenshots showing the testing of query 4	62 - 65
6.5. Screenshots showing the testing of query 5	65 - 70
6.6. Screenshots showing the testing of query 6	70 - 73
6.7. Screenshots showing the testing of query 7	73 - 78
6.8. Screenshots showing the testing of query 8	78 - 79
6.9. Screenshots showing the testing of query 9	79 - 80
6.10. Screenshots showing the testing of query 10	80 - 81
6.11. Screenshots showing the testing of query 11	81 - 82
6.12. Screenshots showing the testing of query 12	83 - 87
6.13. Screenshots showing the testing of query 13	88
6.14. Screenshots showing the testing of query 14	88 - 89
6.15. Screenshots showing the testing of query 15	89 - 90
6.16. Screenshots showing the testing of the import and export options	91
6.17. Screenshots showing the testing of three types of errors	92 - 95
6.18. Screenshots showing the testing of the quit option	96

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_SSNN, policy id)
- **Team** (Name , type, date)
- **Volunteer** (SSN , date_joined, recent_training_date, recent_training_location)
- **Leader(volunteer SSN, Name)**
- **Serves** (**Volunteer_SSNN**, **Team_Name** , active / inactive , hours _worked,month)
- **Cares**(Name,client_SSNN)
- **Employee** (SSN , marital_status, hire_date, salary)
- **Reports** (employee_SSNN, 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
```

Creating Person table

Messages

```
10:21:59 PM    Started executing query at Line 1
                Commands completed successfully.
                Total execution time: 00:00:00.296
```

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
```

);

Messages

```
10:23:00 PM    Started executing query at Line 1
                Commands completed successfully.
                Total execution time: 00:00:00.151
```

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

```
61  -- Creating Needs table
62  CREATE TABLE Needs (
63      type VARCHAR(50),
64      importance INT,
65      client_SSN CHAR(9),
66      FOREIGN KEY (client_SSN) REFERENCES Client(SSN),
67      PRIMARY KEY (client_SSN, type)
68 );
69
70  -- Creating index on importance in Needs table
71  CREATE NONCLUSTERED INDEX idx_needs_importance ON Needs(importance);
72
```

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

```
73  -- Creating Insurance_Policy table
74  CREATE TABLE Insurance_Policy (
75      policy_id CHAR(10) PRIMARY KEY,
76      provider_name VARCHAR(100),
77      provider_address VARCHAR(255),
78      type VARCHAR(50)
79 );
80
```

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);
92
```

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);
102
```

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 );  
123
```

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);
149
```

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);  
175
```

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);
188
```

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);
202
```

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:

```
203  -- Creating Credit_Card table
204  CREATE TABLE Credit_Card (
205      Donor_SSN CHAR(9),
206      date DATE,
207      card_number CHAR(16),
208      card_type VARCHAR(50),
209      Amount DECIMAL(10, 2),
210      type VARCHAR(50),
211      campaign_name VARCHAR(100),
212      expiration_date DATE,
213      PRIMARY KEY (Donor_SSN, card_number),
214      FOREIGN KEY (Donor_SSN) REFERENCES Donor(SSN)
215  );
216
217  -- Creating index on Donor_SSN in Credit_Card table
218  CREATE NONCLUSTERED INDEX idx_credit_card_donor_ssn ON Credit_Card(Donor_SSN);
219
```

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).



```
--Query 1
-- Dropping the procedure if it already exists
GO
DROP PROCEDURE IF EXISTS sp_EnterNewTeam;
GO

CREATE PROCEDURE sp_EnterNewTeam(
    @TeamName VARCHAR(100),
    @TeamType VARCHAR(50),
    @Date DATE
)
AS
BEGIN
    IF @Date IS NULL
    BEGIN
        RAISERROR ('Date cannot be NULL.', 16, 1);
        RETURN;
    END

    BEGIN TRY
        IF NOT EXISTS (SELECT 1 FROM Team WHERE Name = @TeamName)
        BEGIN
            INSERT INTO Team (Name, type, date)
            VALUES (@TeamName, @TeamType, @Date);
            PRINT CONCAT(@TeamName, ' added successfully.');
        END
        ELSE
        BEGIN
            PRINT CONCAT(@TeamName, ' already exists. No new entry added.');
        END
    END TRY
    BEGIN CATCH
        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

2:31:43 PM	Started executing query at Line 1
	Commands completed successfully.
2:31:43 PM	Started executing query at Line 4
	Commands completed successfully.
2:31:43 PM	Started executing query at Line 6
	Commands completed successfully.
	Total execution time: 00:00:00.092

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

```
Users > srihari > Downloads > storedproceduresql
Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db Estimated Plan
45 --Query 2
46 -- Dropping table if it exists already
47 GO
48 DROP PROCEDURE IF EXISTS sp_EnterNewClient;
49 GO
50
51 CREATE PROCEDURE sp_EnterNewClient(
52     @ClientSSN CHAR(9),
53     @Name VARCHAR(50),
54     @Gender CHAR(1),
55     @Profession VARCHAR(50),
56     @MailingAddress VARCHAR(255),
57     @PhoneNumber VARCHAR(15),
58     @EmailID VARCHAR(100),
59     @OnMailingList BIT,
60     @DoctorName VARCHAR(50),
61     @DateJoined DATE,
62     @DoctorPhoneNumber VARCHAR(15),
63     @TeamName VARCHAR(100)
64 )
65 AS
66 BEGIN
67
68     IF @DateJoined IS NULL
69     BEGIN
70         RAISERROR ('DateJoined cannot be NULL.', 16, 1);
71         RETURN;
72     END
73
74     IF LEN(@ClientSSN) <> 9
75     BEGIN
76         RAISERROR ('ClientSSN must be exactly 9 characters long.', 16, 1);
77         RETURN;
78     END
79
80     IF @Gender NOT IN ('M', 'F')
81     BEGIN
82         RAISERROR ('Gender must be "M" or "F".', 16, 1);
83         RETURN;
84     END
```

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

```

Users > srihari > Downloads > storedproceduresql
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 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 | storedproceduresql - (68 b..pp0002) 1
Users > srihari > Downloads > storedproceduresql
Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db Estimated Plan Enable Actual Plan Parse Enable SQLCMD
175     RETURN;
176
177 IF NOT EXISTS (SELECT 1 FROM Team WHERE Name = @TeamName)
178 BEGIN
179     RAISERROR('TeamName does not exist in the Team table.', 16, 1);
180     RETURN;
181 END
182
183 BEGIN TRY
184     BEGIN TRANSACTION;
185
186     IF NOT EXISTS (SELECT 1 FROM Person WHERE SSN = @SSN)
187     BEGIN
188         INSERT INTO Person (SSN, name, gender, profession, mailing_address, email_id, phone_number, on_mailing_list)
189         VALUES (@SSN, @Name, @Gender, @Profession, @MailingAddress, @Email, @PhoneNumber, @OnMailingList);
190         PRINT 'Person entry created successfully。';
191     END
192     ELSE
193     BEGIN
194         PRINT 'Person with this SSN already exists. Proceeding to add volunteer details。';
195     END
196
197     INSERT INTO Volunteer (SSN, date_joined, recent_training_date, recent_training_location)
198     VALUES (@SSN, @DateJoined, @RecentTrainingDate, @RecentTrainingLocation);
199
200     INSERT INTO Serves (Volunteer_SSN, Team_Name, active_inactive, hours_worked, month)
201     VALUES (@SSN, @TeamName, @Status, @HoursWorked, DATEPART(MONTH, GETDATE()));
202
203     COMMIT TRANSACTION;
204
205     PRINT 'Volunteer and person entries added successfully。';
206
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 132
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).

The screenshot shows a SQL query window in SSMS. The title bar indicates the file is 'cretaetables.sql' (line 85) and the database is 'storedproceduresql' (line 68). The code is as follows:

```

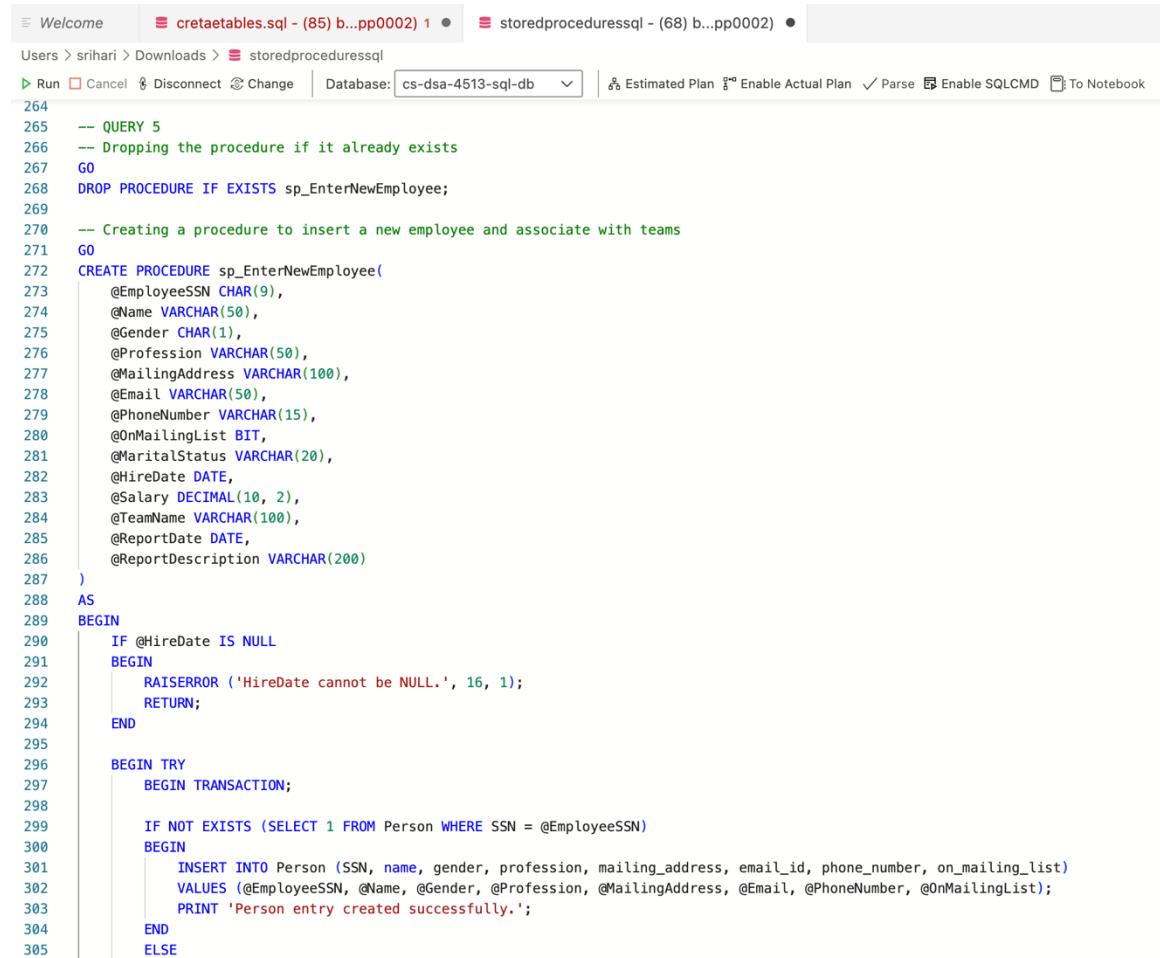
220 -- Query 4
221 -- Drop procedure if it already exists
222 GO
223 DROP PROCEDURE IF EXISTS sp_EnterVolunteerHours;
224 GO
225 CREATE PROCEDURE sp_EnterVolunteerHours(
226     @VolunteerSSN CHAR(9),
227     @TeamName VARCHAR(100),
228     @AdditionalHours INT,
229     @Month INT,
230     @Status CHAR(1)
231 )
232 AS
233 BEGIN
234     IF @Month NOT BETWEEN 1 AND 12
235     BEGIN
236         RAISERROR('Error: Month must be between 1 and 12.', 16, 1);
237         RETURN;
238     END
239     BEGIN TRY
240         IF EXISTS (SELECT 1 FROM Serves WHERE Volunteer_SSN = @VolunteerSSN AND Team_Name = @TeamName AND month = @Month)
241         BEGIN
242             UPDATE Serves
243             SET hours_worked = hours_worked + @AdditionalHours
244             WHERE Volunteer_SSN = @VolunteerSSN AND Team_Name = @TeamName AND month = @Month;
245             PRINT 'Volunteer hours updated successfully for the month.';
246         END
247         ELSE
248             BEGIN
249                 INSERT INTO Serves (Volunteer_SSN, Team_Name, hours_worked, month, active_inactive)
250                 VALUES (@VolunteerSSN, @TeamName, @AdditionalHours, @Month, @Status);
251                 PRINT 'New month entry added with volunteer hours.';
252             END
253     END TRY
254     BEGIN CATCH
255         DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
256         DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
257         DECLARE @ErrorState INT = ERROR_STATE();
258         RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
259     END CATCH
260 END;
261 GO

```

Messages

- 2:34:46 PM Started executing query at Line 220
Commands completed successfully.
- 2:34:46 PM Started executing query at Line 223
Commands completed successfully.
- 2:34:46 PM Started executing query at Line 225
Commands completed successfully.
Total execution time: 00:00:00.114

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



The screenshot shows a SQL Server Management Studio (SSMS) interface. The title bar includes tabs for 'Welcome', 'cretaetables.sql - (85) b...pp0002' (active), and 'storedproceduresql - (68) b...pp0002'. The toolbar has options like 'Run', 'Cancel', 'Disconnect', 'Change', 'Database: cs-dsa-4513-sql-db', and various execution and management tools. The main pane displays a T-SQL script for creating a stored procedure named 'sp_EnterNewEmployee'. The script includes code for dropping the procedure if it exists, defining parameters for employee details and team association, and handling NULL values for hire date. It also includes a TRY-CATCH block for inserting the employee into the 'Person' table.

```
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
306             BEGIN
```

Welcome [cretaetables.sql](#) - (85 b...pp0002) 1 [storedproceduresql](#) - (68 b...pp0002) 2

Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db Estimated Plan Enable Actual Plan Parse Enable SQLCMD Tr

```

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
306             BEGIN
307                 PRINT 'Person with this SSN already exists. Proceeding to add employee details.';
308             END
309
310         INSERT INTO Employee (SSN, marital_status, hire_date, salary)
311             VALUES (@EmployeeSSN, @MaritalStatus, @HireDate, @Salary);
312
313         INSERT INTO Reports (employee_SSN, team_name, date, description)
314             VALUES (@EmployeeSSN, @TeamName, @ReportDate, @ReportDescription);
315
316         COMMIT TRANSACTION;
317
318         PRINT 'Employee and person entries added successfully and associated with team.';
319     END TRY
320     BEGIN CATCH
321         ROLLBACK TRANSACTION;
322         DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
323         DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
324         DECLARE @ErrorState INT = ERROR_STATE();
325         RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
326     END CATCH
327 END;
328 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).

The screenshot shows a SQL Server Management Studio (SSMS) window. The title bar indicates the current file is 'creatables.sql' and the database is 'cs-dsa-4513-sql-db'. The main area contains the T-SQL code for creating a stored procedure named 'sp_EnterNewDonor'. The code includes parameters for SSN, Name, Gender, Profession, Mailing Address, Phone Number, Email ID, On Mailing List, Anonymous status, Donation Count, Donation Type, Amount, Date, Cheque No, Card Number, Card Type, Expiry Date, and Campaign Name. It includes validation logic to check if Date and Amount are not null, and if the SSN does not exist in the Person table, it inserts a new row with the provided values.

```
374 -- QUERY 7
375 -- Dropping the procedure if it already exists
376
377 GO
378 DROP PROCEDURE IF EXISTS sp_EnterNewDonor;
379
380 GO
381 CREATE PROCEDURE sp_EnterNewDonor(
382     @SSN CHAR(9),
383     @Name VARCHAR(100),
384     @Gender CHAR(1),
385     @Profession VARCHAR(50),
386     @MailingAddress VARCHAR(255),
387     @PhoneNumber VARCHAR(15),
388     @EmailID VARCHAR(100),
389     @OnMailingList BIT,
390     @Anonymous BIT,
391     @DonationCount INT,
392     @DonationType NVARCHAR(20),
393     @Amount DECIMAL(10, 2),
394     @Date DATE,
395     @ChequeNo CHAR(10) = NULL,
396     @CardNumber CHAR(16) = NULL,
397     @CardType VARCHAR(50) = NULL,
398     @ExpiryDate DATE = NULL,
399     @CampaignName VARCHAR(100)
400 )
401 AS
402 BEGIN
403     IF @Date IS NULL OR @Amount IS NULL
404     BEGIN
405         RAISERROR ('Date and Amount cannot be NULL.', 16, 1);
406         RETURN;
407     END
408
409     BEGIN TRY
410         SET @DonationType = LTRIM(RTRIM(LOWER(@DonationType)));
411
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     END TRY
418     BEGIN CATCH
419         IF @@TRANCOUNT > 0
420             ROLLBACK TRANSACTION;
421         ELSE
422             IF XACT_STATE() < 0
423                 ROLLBACK TRANSACTION;
424             ELSE
425                 IF XACT_STATE() = 1
426                     COMMIT TRANSACTION;
427     END CATCH
428 END
```

Welcome cretaetables.sql - (85 b...pp0002) 1 storedproceduresql - (68 b...pp0002)

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
453 
```

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)

```
11  -- QUERY 11
12  -- Dropping the procedure if it already exists
13  GO
14  DROP PROCEDURE IF EXISTS sp_RetrieveTeamsFoundedAfterDate;
15
16  -- Creating a procedure to retrieve team names founded after a specified date
17  GO
18  CREATE PROCEDURE sp_RetrieveTeamsFoundedAfterDate(
19      @Date DATE
20  )
21  AS
22  BEGIN
23      SELECT Name
24      FROM Team
25      WHERE date > @Date;
26  END;
27  GO
28
```

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.
```

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

```
409  -- QUERY 12
410  -- Dropping the procedure if it already exists
411  GO
412  DROP PROCEDURE IF EXISTS sp_RetrieveAllPeopleInfo;
413
414  -- Creating a procedure to retrieve all personal and emergency contact information for everyone in the database
415  GO
416  CREATE PROCEDURE sp_RetrieveAllPeopleInfo
417  AS
418  BEGIN
419      SELECT
420          P.SSN,
421          P.name AS PersonName,
422          P.phone_number AS PhoneNumber,
423          P.email_id AS EmailID,
424          P.mailing_address AS MailingAddress,
425          EC.name AS EmergencyContactName,
426          EC.phone_number AS EmergencyContactPhone,
427          EC.relationship AS EmergencyContactRelation
428      FROM
429          Person P
430          LEFT JOIN
431              Emergency_Contact EC ON P.SSN = EC.SSN;
432  END;
433  GO
```

messages

```
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)

```
+ / -  
471  -- QUERY 14  
472  -- Dropping the procedure if it already exists  
473  GO  
474  DROP PROCEDURE IF EXISTS sp_IncreaseSalaryForMultiTeamEmployees;  
475  
...  
476  -- Creating a procedure to increase salary by 10% for employees managing multiple teams  
477  GO  
478  CREATE PROCEDURE sp_IncreaseSalaryForMultiTeamEmployees  
479  AS  
480  BEGIN  
481      UPDATE Employee  
482      SET salary = salary * 1.10  
483      WHERE SSN IN (  
484          SELECT employee_SSN  
485          FROM Reports  
486          GROUP BY employee_SSN  
487          HAVING COUNT(DISTINCT team_name) > 1  
488      );  
489      PRINT 'Salaries updated successfully for eligible employees.';  
490  END;  
491  GO  
492
```

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 L

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
```

messages

```
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 –

```
Users > srihari > Downloads > storedproceduresql
▶ Run □ Cancel ⚙ Disconnect ☰ Change Database: cs-dsa-4513-sql-db ✓
535  -- QUERY 17
536  -- Drop the procedure if it exists
537
538  DROP PROCEDURE IF EXISTS sp_RetrieveMailingList;
539  GO
540
541  -- Creating a new stored procedure to retrieve mailing list da
542  CREATE PROCEDURE sp_RetrieveMailingList
543  AS
544  BEGIN
545      SELECT name, mailing_address
546      FROM Person
547      WHERE on_mailing_list = 1;
548  END;
549  GO
550
551
```

Messages

8:22:47 AM	<u>Started executing query at Line 535</u> Commands completed successfully.
8:22:47 AM	<u>Started executing query at Line 540</u> Commands completed successfully. Total execution time: 00:00:00.448

Error checking – Stored procedures can handle errors

1- Null Date error detection for query 1

The screenshot shows the SQL Server Management Studio interface. In the top navigation bar, there are tabs for 'Welcome', 'cretaetables.sql - (85 b...pp0002) 1' (highlighted in red), and 'storedproceduresql - (68)'. Below the tabs, the path 'Users > srihari > Downloads > cretaetables.sql' is displayed. On the left, there are buttons for 'Run', 'Cancel', 'Disconnect', and 'Change'. The 'Database' dropdown is set to 'cs-dsa-4513-sql-db'. To the right, there are buttons for 'Estimated' and 'Actual' execution plans.

Query Editor Content:

```
237  
238  
239  
240 EXEC sp_EnterNewTeam  
241     @TeamName = 'Research Team',  
242     @TeamType = 'Science',  
243     @Date = NULL;  
244  
245  
246  
247  
248
```

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.

```
Users > srihari > Downloads > 📁 cretaetables.sql
▶ Run ⚡ Cancel ⚡ Disconnect ⚡ Change Database: cs-dsa-4513-sql-db ▾ ⚡ Estimated Pla
236
237 EXEC sp_EnterNewClient
238     @ClientSSN = '12345678',
239     @Name = 'Jane Doe',
240     @Gender = 'F',
241     @Profession = 'Teacher',
242     @MailingAddress = '456 Oak St',
243     @PhoneNumber = '123-456-7890',
244     @EmailID = 'jane.doe@.com',
245     @OnMailingList = 0,
246     @DoctorName = 'Dr. Brown',
247     @DateJoined = '2024-11-14',
248     @DoctorPhoneNumber = '098-765-4321',
249     @TeamName = 'Support Team';
250
251
252
```

Messages

```
1:58:57 PM      Started executing query at Line 237
Msg 50000, Level 16, State 1, Procedure sp_EnterNewClient, Line 27
ClientSSN must be exactly 9 characters long.
Total execution time: 00:00:00.031
```

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

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Welcome', 'cretaetables.sql - (85) b...pp0002' (which is the current tab), and 'storedproceduresql - (68) b...p'. Below the tabs, the file path 'Users > srihari > Downloads > cretaetables.sql' is displayed. There are buttons for 'Run', 'Cancel', 'Disconnect', 'Change', 'Database: cs-dsa-4513-sql-db', and 'Estimated Plan'. The main area contains the following SQL code:

```
258
259 EXEC sp_AddPersonAndVolunteer
260     @SSN = '123456789',
261     @Name = 'Test User',
262     @Gender = 'M',
263     @Profession = 'Engineer',
264     @MailingAddress = '123 Test St',
265     @Email = 'test.user@example.com',
266     @PhoneNumber = '123-456-7890',
267     @OnMailingList = 1,
268     @DateJoined = '2024-11-14',
269     @RecentTrainingDate = '2024-11-13',
270     @RecentTrainingLocation = 'Training Center Test',
271     @TeamName = 'NonExistent Team', -- Use a team name that does not exist
272     @Status = 'A',
273     @HoursWorked = 20;
274
```

Below the code, under the 'Messages' section, the following log entries are shown:

- 2:17:18 PM Started executing query at Line 259
- Msg 50000, Level 16, State 1, Procedure sp_AddPersonAndVolunteer, Line 42
TeamName does not exist in the Team table.
- Total execution time: 00:00:00.030

5.2 The Java source program and screenshots showing its successful compilation

```
1@ import java.sql.*;
2 import java.io.BufferedWriter;
3 import java.io.FileWriter;
4 import java.io.IOException;
5 import java.io.BufferedReader;
6 import java.io.FileReader;
7 import java.util.Scanner;
8 import java.math.BigDecimal;
9
10 public class PANDatabaseApp {
11
12     private static final String HOSTNAME = "bopp0002-sql-server.database.windows.net";
13     private static final String DBNAME = "cs-dsa-4513-sql-db";
14     private static final String USERNAME = "bopp0002";
15     private static final String PASSWORD = "abcdABCD02001";
16     private static final String CONNECTION_URL = "jdbc:sqlserver://" + HOSTNAME + ";database=" + DBNAME + ";user=" + USERNAME + ";password=" + PASSWORD;
17
18@     public static void main(String[] args) {
19         try (Connection connection = DriverManager.getConnection(CONNECTION_URL)) {
20             System.out.println("Connected to the database.");
21             Scanner scanner = new Scanner(System.in);
22
23             boolean running = true;
24             while (running) {
25                 System.out.println("\nSelect an option:");
26                 System.out.println("1. Enter a new team");
27                 System.out.println("2. Enter a new client and associate with teams");
28                 System.out.println("3. Enter a new volunteer and associate with teams");
29                 System.out.println("4. Enter volunteer hours for a team");
30                 System.out.println("5. Enter a new employee and associate with teams");
31                 System.out.println("6. Enter an expense charged by an employee");
32                 System.out.println("7. Enter a new donor and donations");
33                 System.out.println("8. Retrieve doctor information of a client");
34                 System.out.println("9. Retrieve total expenses by each employee for a period");
35                 System.out.println("10. List volunteers for a client's teams");
36                 System.out.println("11. Retrieve teams founded after a specific date");
37                 System.out.println("12. Retrieve all people in the database");
38                 System.out.println("13. Retrieve employee donors with total donations");
39                 System.out.println("14. Increase salary of employees with multiple team reports");
40                 System.out.println("15. Delete clients without insurance and low transportation importance");
41                 System.out.println("16. Import: Enter new teams from a data file");
42                 System.out.println("17. Export: Retrieve names and mailing addresses of people on the mailing list");
43                 System.out.println("18. Quit");
44
45                 int choice = scanner.nextInt();
46                 scanner.nextLine();
47
48                 switch (choice) {
49                     case 1 -> enterNewTeam(connection, scanner);
50                     case 2 -> enterNewClient(connection, scanner);
51                     case 3 -> enterNewVolunteer(connection, scanner);
52                     case 4 -> enterVolunteerHours(connection, scanner);
53                     case 5 -> enterNewEmployee(connection, scanner);
54                     case 6 -> enterExpense(connection, scanner);
55                     case 7 -> enterNewDonor(connection, scanner);
56                     case 8 -> retrieveClientDoctor(connection, scanner);
57                     case 9 -> retrieveTotalExpensesByEmployee(connection, scanner);
58                     case 10 -> listVolunteersForClientTeams(connection, scanner);
59                     case 11 -> retrieveTeamsFoundedAfterDate(connection, scanner);
60                     case 12 -> retrieveAllPeopleInfo(connection);
61                     case 13 -> retrieveEmployeeDonors(connection);
62                     case 14 -> increaseSalaryForMultiTeamEmployees(connection);
63                     case 15 -> deleteUninsuredClientsWithLowTransportValue(connection);
64                     case 16 -> importTeamsFromFile(connection, scanner);
65                     case 17 -> exportMailingListToFile(connection, scanner);
66                     case 18 -> running = false;
67                 }
68             }
69         } catch (SQLException e) {
70             e.printStackTrace();
71         }
72     }
73 }
```

```

6             case 18 -> running = false;
7         }
8     }
9     System.out.println("Exiting the application.");
10 } catch (SQLException e) {
11     System.out.println("Error connecting to the database: " + e.getMessage());
12 }
13 }
14
15 private static void enterNewTeam(Connection connection, Scanner scanner) {
16     System.out.print("Enter team name: ");
17     String teamName = scanner.nextLine();
18     System.out.print("Enter team type: ");
19     String teamType = scanner.nextLine();
20     System.out.print("Enter formation date (yyyy-mm-dd): ");
21     String date = scanner.nextLine();
22
23     try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewTeam(?, ?, ?)}")) {
24         stmt.setString(1, teamName);
25         stmt.setString(2, teamType);
26         stmt.setDate(3, Date.valueOf(date));
27         stmt.execute();
28         System.out.println("Team added successfully.");
29     } catch (SQLException e) {
30         System.out.println("Error adding team: " + e.getMessage());
31     }
32 }
33
34 private static void enterNewClient(Connection connection, Scanner scanner) {
35     System.out.print("Enter client SSN: ");
36     String clientSSN = scanner.nextLine();
37     System.out.print("Enter client name: ");
38     String name = scanner.nextLine();
39     System.out.print("Enter client gender (M/F): ");
40     String gender = scanner.nextLine();
41     System.out.print("Enter client profession: ");
42     String profession = scanner.nextLine();
43     System.out.print("Enter client mailing address: ");
44     String mailingAddress = scanner.nextLine();
45     System.out.print("Enter client phone number: ");
46     String phoneNumber = scanner.nextLine();
47     System.out.print("Enter client email ID: ");
48     String emailID = scanner.nextLine();
49     System.out.print("Is the client on the mailing list? (1 for yes, 0 for no): ");
50     boolean onMailingList = scanner.nextInt() == 1;
51     scanner.nextLine();
52     System.out.print("Enter doctor name: ");
53     String doctorName = scanner.nextLine();
54     System.out.print("Enter date joined (yyyy-mm-dd): ");
55     String dateJoined = scanner.nextLine();
56     System.out.print("Enter doctor phone number: ");
57     String doctorPhone = scanner.nextLine();
58     System.out.print("Enter team name: ");
59     String teamName = scanner.nextLine();
60
61     try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewClient(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
62         stmt.setString(1, clientSSN);
63         stmt.setString(2, name);
64         stmt.setString(3, gender);
65         stmt.setString(4, profession);
66         stmt.setString(5, mailingAddress);
67         stmt.setString(6, phoneNumber);
68         stmt.setString(7, emailID);
69         stmt.setBoolean(8, onMailingList);
70     }
71 }

```

```

130         stmt.setBoolean(8, onMailingList);
131         stmt.setString(9, doctorName);
132         stmt.setDate(10, Date.valueOf(dateJoined));
133         stmt.setString(11, doctorPhone);
134         stmt.setString(12, teamName);
135         stmt.execute();
136         System.out.println("Client added and associated with team successfully.");
137     } catch (SQLException e) {
138         System.out.println("Error adding client: " + e.getMessage());
139     }
140 }
141
142
143
144 private static void enterNewVolunteer(Connection connection, Scanner scanner) {
145     System.out.print("Enter volunteer SSN: ");
146     String volunteerSSN = scanner.nextLine();
147     System.out.print("Enter volunteer name: ");
148     String name = scanner.nextLine();
149     System.out.print("Enter gender (M/F): ");
150     String gender = scanner.nextLine();
151     System.out.print("Enter profession: ");
152     String profession = scanner.nextLine();
153     System.out.print("Enter mailing address: ");
154     String address = scanner.nextLine();
155     System.out.print("Enter email: ");
156     String email = scanner.nextLine();
157     System.out.print("Enter phone number: ");
158     String phone = scanner.nextLine();
159     System.out.print("Is the volunteer on the mailing list? (1 for yes, 0 for no): ");
160     int onMailingList = scanner.nextInt();
161     scanner.nextLine();
162
163     System.out.print("Enter date joined (yyyy-mm-dd): ");
164     String dateJoined = scanner.nextLine();
165     System.out.print("Enter recent training date (yyyy-mm-dd): ");
166     String trainingDate = scanner.nextLine();
167     System.out.print("Enter recent training location: ");
168     String trainingLocation = scanner.nextLine();
169     System.out.print("Enter team name: ");
170     String teamName = scanner.nextLine();
171     System.out.print("Enter status (A for active, I for inactive): ");
172     String status = scanner.nextLine();
173     System.out.print("Enter hours worked: ");
174     int hoursWorked = scanner.nextInt();
175     scanner.nextLine();
176
177     try (CallableStatement stmt = connection.prepareCall("{call sp_AddPersonAndVolunteer(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
178         stmt.setString(1, volunteerSSN);
179         stmt.setString(2, name);
180         stmt.setString(3, gender);
181         stmt.setString(4, profession);
182         stmt.setString(5, address);
183         stmt.setString(6, email);
184         stmt.setString(7, phone);
185         stmt.setInt(8, onMailingList);
186         stmt.setDate(9, Date.valueOf(dateJoined));
187         stmt.setDate(10, Date.valueOf(trainingDate));
188         stmt.setString(11, trainingLocation);
189         stmt.setString(12, teamName);
190         stmt.setString(13, status);
191         stmt.setInt(14, hoursWorked);
192         stmt.execute();
193         System.out.println("Volunteer and person entries added successfully and associated with the team.");
194     } catch (SQLException e) {
195         System.out.println("Error adding volunteer: " + e.getMessage());

```

```

195     System.out.println("Error adding volunteer: " + e.getMessage());
196 }
197
198
199
200 private static void enterVolunteerHours(Connection connection, Scanner scanner) {
201     System.out.print("Enter volunteer SSN: ");
202     String volunteerSSN = scanner.nextLine();
203     System.out.print("Enter team name: ");
204     String teamName = scanner.nextLine();
205     System.out.print("Enter additional hours worked: ");
206     int hours = scanner.nextInt();
207     System.out.print("Enter month: ");
208     int month = scanner.nextInt();
209     System.out.print("Enter status (A for active, I for inactive): ");
210     String status = scanner.next();
211     scanner.nextLine();
212
213     try (CallableStatement stmt = connection.prepareCall("{call sp_EnterVolunteerHours(?, ?, ?, ?, ?, ?)}")) {
214         stmt.setString(1, volunteerSSN);
215         stmt.setString(2, teamName);
216         stmt.setInt(3, hours);
217         stmt.setInt(4, month);
218         stmt.setString(5, status);
219         stmt.execute();
220         System.out.println("Volunteer hours updated successfully.");
221     } catch (SQLException e) {
222         System.out.println("Error updating volunteer hours: " + e.getMessage());
223     }
224 }
225
226
227
228 private static void enterNewEmployee(Connection connection, Scanner scanner) {
229     System.out.print("Enter employee SSN: ");
230     String employeeSSN = scanner.nextLine();
231     System.out.print("Enter employee name: ");
232     String name = scanner.nextLine();
233     System.out.print("Enter employee gender (M/F): ");
234     String gender = scanner.nextLine();
235     System.out.print("Enter employee profession: ");
236     String profession = scanner.nextLine();
237     System.out.print("Enter mailing address: ");
238     String mailingAddress = scanner.nextLine();
239     System.out.print("Enter email ID: ");
240     String email = scanner.nextLine();
241     System.out.print("Enter phone number: ");
242     String phoneNumber = scanner.nextLine();
243     System.out.print("Is the employee on the mailing list? (1 for yes, 0 for no): ");
244     int onMailingList = scanner.nextInt();
245     scanner.nextLine();
246     System.out.print("Enter marital status: ");
247     String maritalStatus = scanner.nextLine();
248     System.out.print("Enter hire date (yyyy-mm-dd): ");
249     String hireDate = scanner.nextLine();
250     System.out.print("Enter salary: ");
251     double salary = scanner.nextDouble();
252     scanner.nextLine();
253     System.out.print("Enter team name: ");
254     String teamName = scanner.nextLine();
255     System.out.print("Enter report date (yyyy-mm-dd): ");
256     String reportDate = scanner.nextLine();
257     System.out.print("Enter report description: ");
258     String reportDescription = scanner.nextLine();
259
260     try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewEmployee(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
261         stmt.setString(1, employeeSSN);

```

```

PANDatabaseApp.java X
260     try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewEmployee(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}")) {
261         stmt.setString(1, employeeSSN);
262         stmt.setString(2, name);
263         stmt.setString(3, gender);
264         stmt.setString(4, profession);
265         stmt.setString(5, mailingAddress);
266         stmt.setString(6, email);
267         stmt.setString(7, phoneNumber);
268         stmt.setInt(8, onMailingList);
269         stmt.setString(9, maritalStatus);
270         stmt.setDate(10, Date.valueOf(hireDate));
271         stmt.setDouble(11, salary);
272         stmt.setString(12, teamName);
273         stmt.setDate(13, Date.valueOf(reportDate));
274         stmt.setString(14, reportDescription);
275         stmt.execute();
276         System.out.println("Employee added and associated with team successfully.");
277     } catch (SQLException e) {
278         System.out.println("Error adding employee: " + e.getMessage());
279     }
280 }
281
282 private static void enterExpense(Connection connection, Scanner scanner) {
283     System.out.print("Enter employee SSN: ");
284     String employeeSSN = scanner.nextLine();
285     System.out.print("Enter expense date (yyyy-mm-dd): ");
286     String expenseDate = scanner.nextLine();
287     System.out.print("Enter amount: ");
288     double amount = scanner.nextDouble();
289     scanner.nextLine();
290     System.out.print("Enter description: ");
291     String description = scanner.nextLine();
292
293     try (CallableStatement stmt = connection.prepareCall("{call sp_EnterExpense(?, ?, ?)}")) {
294         stmt.setString(1, employeeSSN);
295         stmt.setDate(2, Date.valueOf(expenseDate));
296         stmt.setDouble(3, amount);
297         stmt.setString(4, description);
298         stmt.execute();
299         System.out.println("Expense entry created successfully.");
300     } catch (SQLException e) {
301         System.out.println("Error adding expense: " + e.getMessage());
302     }
303 }
304
305 private static void enterNewDonor(Connection connection, Scanner scanner) {
306     System.out.print("Enter donor SSN: ");
307     String donorSSN = scanner.nextLine();
308     System.out.print("Enter donor name: ");
309     String name = scanner.nextLine();
310     System.out.print("Enter donor gender (M/F): ");
311     String gender = scanner.nextLine();
312     System.out.print("Enter donor profession: ");
313     String profession = scanner.nextLine();
314     System.out.print("Enter mailing address: ");
315     String mailingAddress = scanner.nextLine();
316     System.out.print("Enter email ID: ");
317     String email = scanner.nextLine();
318     System.out.print("Enter phone number: ");
319     String phoneNumber = scanner.nextLine();
320     System.out.print("Is the donor on the mailing list? (1 for yes, 0 for no): ");
321     boolean onMailingList = scanner.nextInt() == 1;
322     System.out.print("Is the donor anonymous? (1 for yes, 0 for no): ");
323     boolean anonymous = scanner.nextInt() == 1;
324     System.out.print("Enter the number of donations: ");
325     int donationCount = scanner.nextInt();

```



```

J PANDatabaseApp.java X
391         stmt.setString(1, clientSSN);
392         ResultSet rs = stmt.executeQuery();
393         while (rs.next()) {
394             System.out.println("Doctor Name: " + rs.getString("doctor_name"));
395             System.out.println("Doctor Phone Number: " + rs.getString("doctor_phone_number"));
396         }
397     } catch (SQLException e) {
398         System.out.println("Error retrieving doctor information: " + e.getMessage());
399     }
400 }
401
402 private static void retrieveTotalExpensesByEmployee(Connection connection, Scanner scanner) {
403     System.out.print("Enter start date (yyyy-mm-dd): ");
404     String startDate = scanner.nextLine();
405     System.out.print("Enter end date (yyyy-mm-dd): ");
406     String endDate = scanner.nextLine();
407
408     try (CallableStatement stmt = connection.prepareCall("{call sp_TotalExpensesByEmployee(?, ?)}")) {
409         stmt.setDate(1, Date.valueOf(startDate));
410         stmt.setDate(2, Date.valueOf(endDate));
411         ResultSet rs = stmt.executeQuery();
412         while (rs.next()) {
413             System.out.println("Employee SSN: " + rs.getString("employee_SSN"));
414             System.out.println("Total Expenses: " + rs.getDouble("total_expenses"));
415         }
416     } catch (SQLException e) {
417         System.out.println("Error retrieving total expenses: " + e.getMessage());
418     }
419 }
420
421 private static void listVolunteersForClientTeams(Connection connection, Scanner scanner) {
422     System.out.print("Enter client SSN: ");
423     String clientSSN = scanner.nextLine();
424
425     try (CallableStatement stmt = connection.prepareCall("{call sp_VolunteersForClientTeams(?)}")) {
426         stmt.setString(1, clientSSN);
427         ResultSet rs = stmt.executeQuery();
428         while (rs.next()) {
429             System.out.println("Volunteer SSN: " + rs.getString("SSN"));
430             System.out.println("Recent Training Location: " + rs.getString("recent_training_location"));
431         }
432     } catch (SQLException e) {
433         System.out.println("Error listing volunteers: " + e.getMessage());
434     }
435 }
436
437 private static void retrieveTeamsFoundedAfterDate(Connection connection, Scanner scanner) {
438     System.out.print("Enter date (yyyy-mm-dd): ");
439     String date = scanner.nextLine();
440
441     try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveTeamsFoundedAfterDate(?)}")) {
442         stmt.setDate(1, Date.valueOf(date));
443         ResultSet rs = stmt.executeQuery();
444         while (rs.next()) {
445             System.out.println("Team Name: " + rs.getString("Name"));
446         }
447     } catch (SQLException e) {
448         System.out.println("Error retrieving teams: " + e.getMessage());
449     }
450 }
451
452 private static void retrieveAllPeopleInfo(Connection connection) {
453     try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveAllPeopleInfo()}")) {
454         ResultSet rs = stmt.executeQuery();
455         while (rs.next()) {
456             System.out.println("Person Name: " + rs.getString("PersonName"));

```

```

1 PANDatabaseApp.java X
455     System.out.println("Person Name: " + rs.getString("PersonName"));
456     System.out.println("SSN: " + rs.getString("SSN"));
457     System.out.println("Mailing Address: " + rs.getString("MailingAddress"));
458     System.out.println("Phone Number: " + rs.getString("PhoneNumber"));
459     System.out.println("Email ID: " + rs.getString("EmailID"));
460     System.out.println("Emergency Contact Name: " + rs.getString("EmergencyContactName"));
461     System.out.println("Emergency Contact Phone: " + rs.getString("EmergencyContactPhone"));
462     System.out.println("Emergency Contact Relation: " + rs.getString("EmergencyContactRelation"));
463     System.out.println("-----");
464 }
465 }
466 } catch (SQLException e) {
467     System.out.println("Error retrieving people: " + e.getMessage());
468 }
469 }
470
471
472
473 private static void retrieveEmployeeDonors(Connection connection) {
474     try (CallableStatement stmt = connection.prepareCall("{call sp_RetrieveEmployeeDonors()}")) {
475         ResultSet rs = stmt.executeQuery();
476         while (rs.next()) {
477             System.out.println("SSN: " + rs.getString("SSN"));
478             System.out.println("Is Anonymous: " + rs.getBoolean("IsAnonymous"));
479             System.out.println("Total Credit Donations: $" + rs.getDouble("TotalCreditDonation"));
480             System.out.println("-----");
481         }
482     } catch (SQLException e) {
483         System.out.println("Error retrieving employee donors: " + e.getMessage());
484     }
485 }
486
487
488 private static void increaseSalaryForMultiTeamEmployees(Connection connection) {
489     try (CallableStatement stmt = connection.prepareCall("{call sp_IncreaseSalaryForMultiTeamEmployees()}")) {
490         stmt.execute();
491         System.out.println("Salary updated successfully for eligible employees.");
492     } catch (SQLException e) {
493         System.out.println("Error updating salary: " + e.getMessage());
494     }
495 }
496
497
498 private static void DeleteUninsuredClientsWithLowTransportValue(Connection connection) {
499     try (CallableStatement stmt = connection.prepareCall("{call sp_DeleteUninsuredClientsWithLowTransportValue()}")) {
500         stmt.execute();
501         System.out.println("Clients without insurance and low importance for transportation were deleted.");
502     } catch (SQLException e) {
503         System.out.println("Error deleting clients: " + e.getMessage());
504     }
505 }
506
507
508
509 private static void importTeamsFromFile(Connection connection, Scanner scanner) {
510     System.out.print("Enter input file name for importing teams: ");
511     String fileName = scanner.nextLine();
512
513     try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
514         String line;
515         while ((line = br.readLine()) != null) {
516             String[] values = line.split(",");
517             if (values.length == 3) {
518                 String teamName = values[0].trim();
519                 String teamType = values[1].trim();
520                 String formationDate = values[2].trim();
521             }
522         }
523     }
524 }

```

```

J PANDatabaseApp.java X
500         stmt.execute();
501         System.out.println("Clients without insurance and low importance for transportation were deleted.");
502     } catch (SQLException e) {
503         System.out.println("Error deleting clients: " + e.getMessage());
504     }
505 }
506
507
508
509 private static void importTeamsFromFile(Connection connection, Scanner scanner) {
510     System.out.print("Enter input file name for importing teams: ");
511     String fileName = scanner.nextLine();
512
513     try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
514         String line;
515         while ((line = br.readLine()) != null) {
516             String[] values = line.split(",");
517             if (values.length == 3) {
518                 String teamName = values[0].trim();
519                 String teamType = values[1].trim();
520                 String formationDate = values[2].trim();
521
522                 try (CallableStatement stmt = connection.prepareCall("{call sp_EnterNewTeam(?, ?, ?)}")) {
523                     stmt.setString(1, teamName);
524                     stmt.setString(2, teamType);
525                     stmt.setDate(3, Date.valueOf(formationDate));
526                     stmt.execute();
527                     System.out.println("Team " + teamName + " added.");
528                 } catch (SQLException e) {
529                     System.out.println("Error adding team " + teamName + ": " + e.getMessage());
530                 }
531             } else {
532                 System.out.println("Invalid line format: " + line);
533             }
534         }
535         System.out.println("Import completed.");
536     } catch (IOException e) {
537         System.out.println("Error reading file: " + e.getMessage());
538     }
539 }
540
541 private static void exportMailingListToFile(Connection connection, Scanner scanner) {
542     System.out.print("Enter output file name for exporting mailing list: ");
543     String fileName = scanner.nextLine();
544
545     try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileName))) {
546         String query = "SELECT name, mailing_address FROM Person WHERE on_mailing_list = 1";
547         try (Statement stmt = connection.createStatement();
548              ResultSet rs = stmt.executeQuery(query)) {
549             bw.write("Name,MailingAddress");
550             bw.newLine();
551             while (rs.next()) {
552                 String name = rs.getString("name");
553                 String mailingAddress = rs.getString("mailing_address");
554                 bw.write(name + "," + mailingAddress);
555                 bw.newLine();
556             }
557             System.out.println("Export completed.");
558         } catch (SQLException e) {
559             System.out.println("Database error: " + e.getMessage());
560         }
561     } catch (IOException e) {
562         System.out.println("File writing error: " + e.getMessage());
563     }
564 }
565 }

```

Output –

```
Console X
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (Nov 14, 2024, 8:25:5
Connected to the database.

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
```

Task 6. Java program Execution

6.1. Screenshots showing the testing of query 1

```
Connected to the database.

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: A
Enter team type: Help
Enter formation date (yyyy-mm-dd): 2001-09-08
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: 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: Vegeta  
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.
```

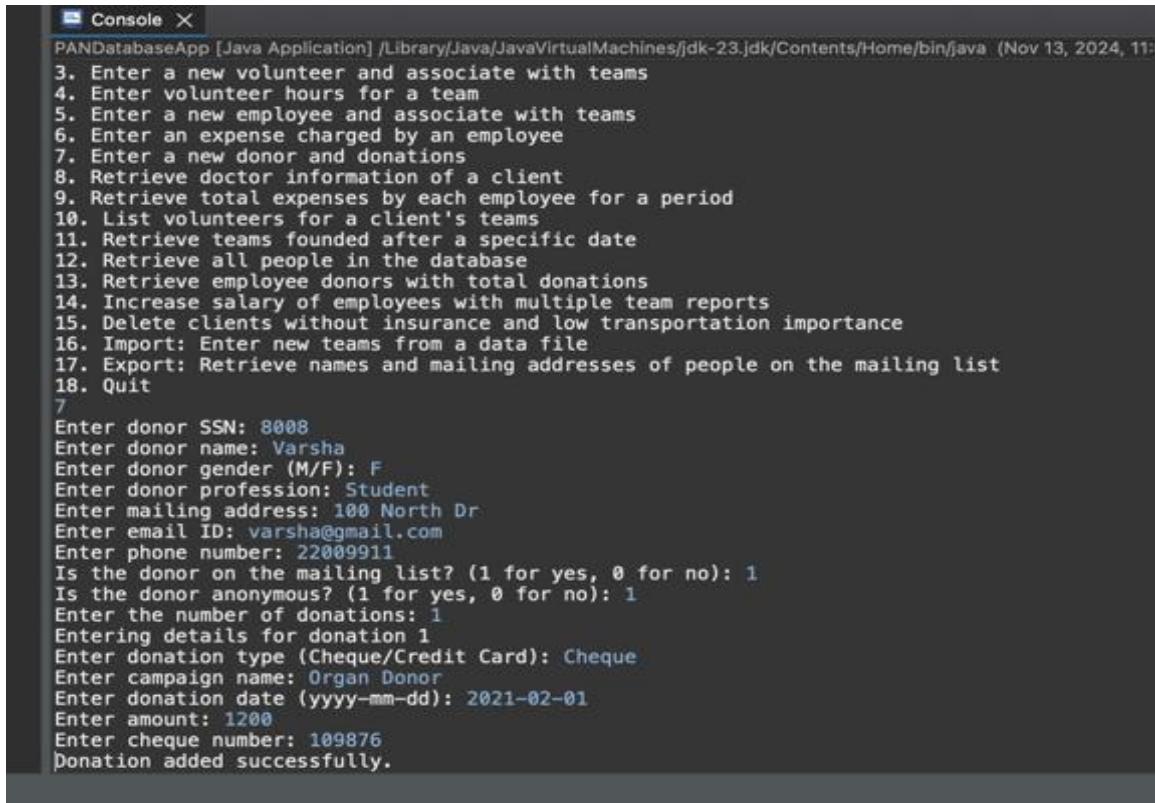
Output -

```
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



The screenshot shows a Java application console window titled "Console". The title bar also displays the path "PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java" and the date and time "(Nov 13, 2024, 11:00:00 AM)". The console output is as follows:

```
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: 8008
Enter donor name: Varsha
Enter donor gender (M/F): F
Enter donor profession: Student
Enter mailing address: 100 North Dr
Enter email ID: varsha@gmail.com
Enter phone number: 22009911
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): Cheque
Enter campaign name: Organ Donor
Enter donation date (yyyy-mm-dd): 2021-02-01
Enter amount: 1200
Enter cheque number: 109876
Donation added successfully.
```

```
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (N
5. 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 X
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
217 |
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 X
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (No
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

```
es FANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java -m
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
12
Person Name: Hari
SSN: 1001
Mailing Address: 1900 N
Phone Number: 209876543
Email ID: hari@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
-----
Person Name: Ramya
SSN: 1234
Mailing Address: 398 H
Phone Number: 2390090
Email ID: ramya@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
-----
Person Name: Varsha
SSN: 1777
Mailing Address: 300 North Drake
Phone Number: 38891002
Email ID: varhsa@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
-----
Person Name: Sriya
SSN: 2001
Mailing Address: 1300 S
Phone Number: 2864087
Email ID: Sriya@gmail.com
Emergency Contact Name: null
Emergency Contact Phone: null
Emergency Contact Relation: null
```

```
S:\FANDatabase\app [Java Application]\Library\Java\javavm\code
```

```
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

```
Console X
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (N
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
13
SSN: 40041
Is Anonymous: false
Total Credit Donations: $0.0
-----
SSN: 50981
Is Anonymous: false
Total Credit Donations: $0.0
-----
SSN: 50051
Is Anonymous: false
Total Credit Donations: $1500.0
-----
SSN: 8008
Is Anonymous: true
Total Credit Donations: $0.0
-----
SSN: 30031
Is Anonymous: true
Total Credit Donations: $1200.0
-----
```

6.14. Screenshots showing the testing of query 14

```
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
14
Salary updated successfully for eligible employees.
```

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 -

```
217  SELECT*FROM Needs
218  SELECT*FROM Insurance_Policy
219  SELECT*FROM Client
220
221
222
223
224
225
226
227
228
229
230
231
232
```

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

```
Console X
PANDatabaseApp [Java Application] /Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java (Nov 14, 2024, 7:40:55 AM) [pid: 1606]
1/. Export: Retrieve names and mailing addresses of people on the mailing list
18. Quit
16
Enter input file name for importing teams: /Users/yourname/Desktop/query16.csv
Error reading file: /Users/yourname/Desktop/query16.csv (No such file or directory)

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
16
Enter input file name for importing teams: /Users/srihari/Desktop/query16.csv
Team Alpha added.
Import completed.
```

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 -

The screenshot shows a spreadsheet application window titled "mailinglist". The ribbon menu includes "View", "Zoom" (set to 125%), "Add Category", "Pivot Table", "Insert", "Table", "Chart", and "Text". The active sheet is "Sheet 1". A tooltip "Table da" is visible near the top right. The data is displayed in a table:

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.

```
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: 10998  
Enter team name: Grey  
Enter additional hours worked: 20  
Enter month: 4  
Enter status (A for active, I for inactive): A  
Error updating volunteer hours: Invalid object name 'Serves'.
```

- 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.
```