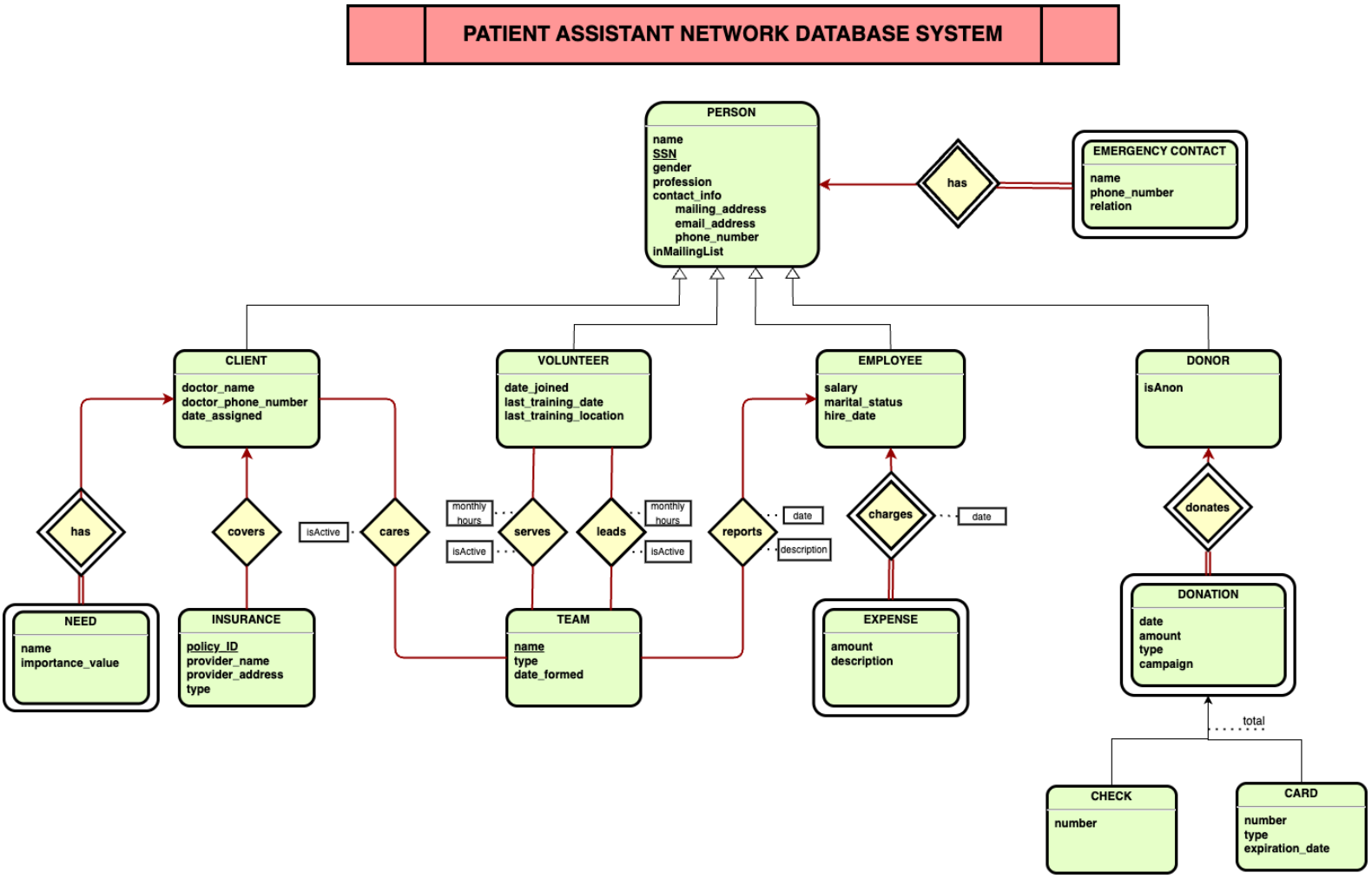
COURSE: CS-4513 – DATABASE MANAGEMENT SECTION: 001
SEMESTER: FALL 2024
INSTRUCTOR: DR. LE GRUENWALD
AUTHOR: Alicia Maranville, 113548144, Alicia.D.Maranville-1@ou.edu
TITLE: A PATIENT ASSISTANT NETWORK DATABASE SYSTEM

Tasks Performed **Page Number**

Task 1. ER Diagram



Task 2. Relational Database Schemas

emergency_contact(SSN, contact_name, phone_number, relation)

client(client_name, SSN, gender, profession, mailing_address, email_address, phone_number, inMailingList, doctor_name, doctor_phone_number, date_assigned)

need(SSN, need_name, importance_value)

insurance(policy_ID, provider_name, provider_address, type, SSN)

volunteer(name, SSN, gender, profession, mailing_address, email_address, phone_number, inMailingList, date_joined, last_training_date, last_training_location)

employee(name, SSN, gender, profession, mailing_address, email_address, phone_number, inMailingList, salary, marital_status, hire_date)

expense(SSN, amount, description, date)

team(name, type, date_formed)

cares(client_SSN, team_name, isActive)

leads(volunteer_SSN, team_name, hours, isActive)

serves(volunteer_SSN, team_name, hours, isActive)

reports(employee_SSN, team_name, date, description)

donor(name, SSN, gender, profession, mailing_address, email_address, phone_number, inMailingList, isAnon)

check_donation(SSN, date, amount, type, campaign, check_number)

card_donation(SSN, date, amount, type, campaign, card_number, card_type, card_expiration_date)

Task 3.

3.1. Discussion of storage structures for tables

Table Name	Query # and Type	Search Key	Query Frequency	Selected File Organization	Justifications
emergency_contact	12 (random search)	SSN	1/week	Heap File	Since we are only performing a random search by SSN once a week, having any extra file organization is unnecessary.
client	2 (insertion) 8 (random search) 10 (random search) 12 (random search) 15 (deletion)	n/a SSN SSN SSN n/a	1/week 1/week 4/year 1/week 4/year	B+-Tree: Index on SSN	Efficient insertion, random search, and deletion by criteria. Index on SSN because all search keys are SSN.
need	15 (range search)	SSN, name, importance_value	4/year	Indexed Sequential File: Secondary Indices on Name and Importance Value	Efficient because we don't do insertion. Can quickly search for SSN who have transportation with value < 5.
insurance	15 (range search)	SSN, type	4/year	Indexed Sequential File: Secondary Index on Type	Will be able to quickly filter by insurance type to find clients without health insurance.
volunteer	3 (insertion) 10 (random search) 12 (random search)	n/a SSN SSN	2/month 4/year 1/week	B+- Tree: Index on SSN	Efficient insertion and random search. Index on SSN because all search keys are SSN.
employee	5 (insertion) 9 (range search) 12 (random search) 13 (random search) 14 (random search)	n/a SSN SSN SSN SSN	1/year 1/month 1/week 1/week 1/year	B+- Tree: Index on SSN	Efficient insertion and random search. Index on SSN because all search keys are SSN.
expense	6 (insertion) 9 (range search)	n/a SSN, date	1/day 1/month	B+- Tree: Index on SSN and Date	Efficient insertion and range search. Index on composite key because SSN and Date make up the primary key of expense.

team	1 (insertion) 11 (range search)	n/a date_formed	1/month 1/month	Indexed Sequential File: Secondary Index on Date_Formed	Infrequent insertion and efficient range search. Index on Date_Formed because that is the search key.
cares	2 (insertion) 10 (random search)	n/a client_SSN, isActive	1/week 4/year	B+- Tree: Index on Client_SSN and Team_Name	Frequently performing insertion and infrequently performing random search. B+- tree works well for both with the primary key as index.
leads	3 (insertion) 4 (insertion) 10 (random search)	n/a n/a team_name, isActive	2/month 30/month 4/year	B+- Tree: Index on Volunteer_SSN and Team_Name	Very frequently performing insertion and infrequently performing random search. B+- tree works well for both with the primary key as index.
serves	3 (insertion) 4 (insertion) 10 (random search)	n/a n/a team_name, isActive	2/month 30/month 4/year	B+- Tree: Index on Volunteer_SSN and Team_Name	Very frequently performing insertion and infrequently performing random search. B+- tree works well for both with the primary key as index.
reports	5 (insertion) 14 (range search)	n/a employee_SSN	1/year 1/year	Indexed Sequential File: Primary Indices Employee_SSN and Team_Name	Only inserting once a year, so indexed sequential is fine. Performing range search on SSN so use the primary key as index.
donor	7 (insertion) 12 (random search) 13 (random search)	n/a SSN SSN	1/day 1/week 1/week	B+- Tree: Index on SSN	Efficient insertions and searches. Index on SSN because all search keys are SSN.
check_donation	7 (insertion) 13 (random search)	n/a SSN	1/day 1/week	B+- Tree: Index on SSN and Date	Efficient insertions and searches. Index on SSN and Date because all search keys are SSN, and SSN/Date make up the primary key.

card_donation	7 (insertion) 13 (random search)	n/a SSN	1/day 1/week	B+- Tree: Index on SSN and Date	Efficient insertions and searches. Index on SSN because all search keys are SSN, and SSN/Date make up the primary key.

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

B tree structures are used in Azure while Indexed-Sequential Files are not used, so I will change the Indexed-Sequential Files to B tree structures with the same indices as before. In Azure, data is either stored in unordered tables or clustered tables. Unordered tables store data as they are inserted (heap files) while clustered tables have ordering within the table's rows based on the primary key. Non-clustered indexes have a separate B tree that references the rows. Thus, in my database, I will use clustered indexes for some tables and non-clustered indexes for some tables depending on whether or not the index is on the primary key or not.

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

```
-- TASK 4: Creating tables and indices

-- DROP EXISTING TABLES

DROP TABLE IF EXISTS card_donation;
DROP TABLE IF EXISTS check_donation;
DROP TABLE IF EXISTS donor;
DROP TABLE IF EXISTS reports;
DROP TABLE IF EXISTS serves;
DROP TABLE IF EXISTS leads;
DROP TABLE IF EXISTS cares;
DROP TABLE IF EXISTS team;
DROP TABLE IF EXISTS expense;
DROP TABLE IF EXISTS employee;
DROP TABLE IF EXISTS volunteer;
DROP TABLE IF EXISTS insurance;
DROP TABLE IF EXISTS need;
DROP TABLE IF EXISTS client;
DROP TABLE IF EXISTS emergency_contact;
DROP TABLE IF EXISTS person;


-- CREATE TABLES / INDICES

CREATE TABLE person(
   person_name VARCHAR(100),
   SSN CHAR(11),
   gender VARCHAR(6) CHECK (gender = 'Male' OR gender = 'Female'),
   profession VARCHAR(100),
   mailing_address VARCHAR(250),
   email_address VARCHAR(100),
   phone_number CHAR(10),
   inMailingList BIT, -- 1 is in, 0 is not in
   PRIMARY KEY (SSN)
);


CREATE TABLE emergency_contact(
   SSN CHAR(11),
   contact_name VARCHAR(100),
```

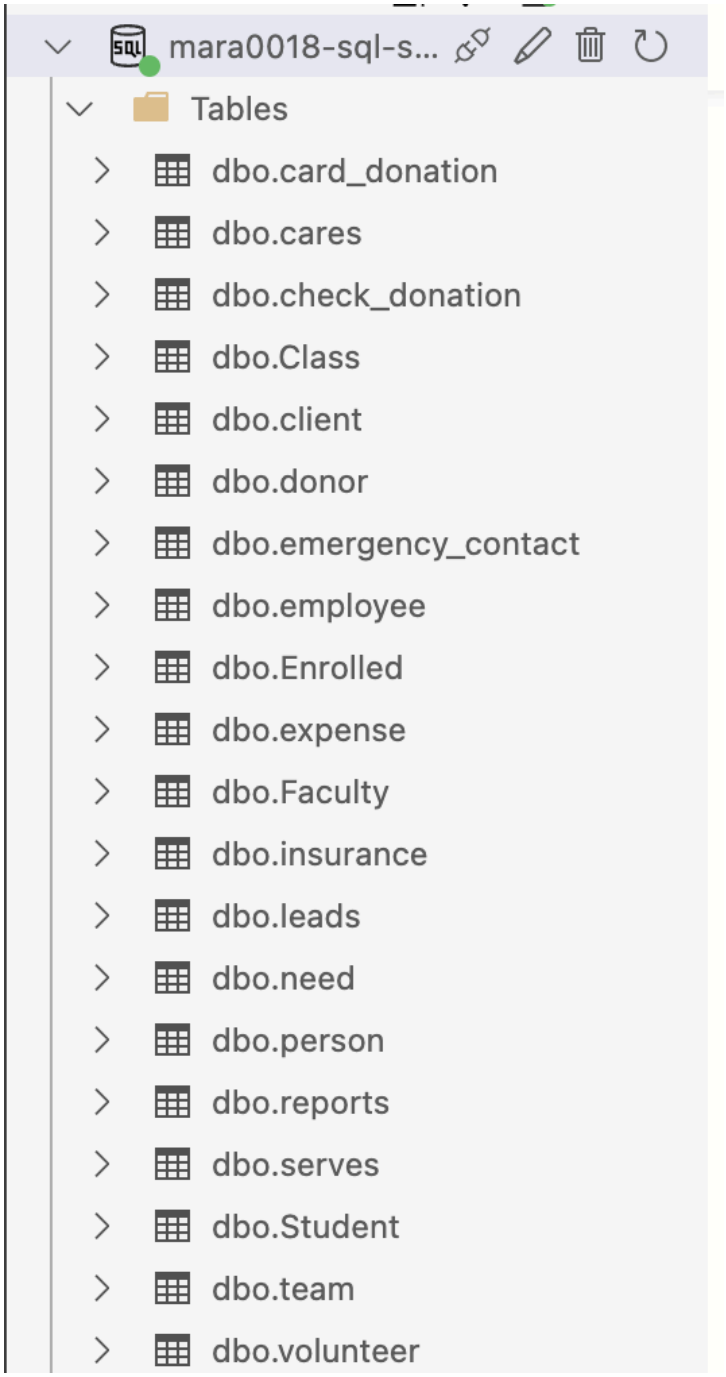
```
    phone_number CHAR(10),
    relation VARCHAR(50),
    PRIMARY KEY (SSN, contact_name),
    FOREIGN KEY (SSN) REFERENCES person ON DELETE CASCADE
);


CREATE TABLE client(
    SSN CHAR(11) PRIMARY KEY,
    doctor_name VARCHAR(100),
    doctor_phone_number CHAR(10),
    date_assigned DATE -- yyyy-MM-dd
    FOREIGN KEY (SSN) REFERENCES person ON DELETE CASCADE
);
CREATE INDEX idx_client_SSN ON client(SSN);


CREATE TABLE need(
    SSN CHAR(11),
    need_name VARCHAR(50),
    importance_value INT CHECK (importance_value > -1 AND importance_value < 11),
    PRIMARY KEY (SSN, need_name),
    FOREIGN KEY (SSN) REFERENCES client ON DELETE CASCADE
);
CREATE INDEX idx_need_name ON need(need_name);
CREATE INDEX idx_importance_value ON need(importance_value);


CREATE TABLE insurance(
    policy_ID VARCHAR(20) PRIMARY KEY,
    provider_name VARCHAR(50),
    provider_address VARCHAR(250),
    insurance_type VARCHAR(6) CHECK (insurance_type = 'Life' OR insurance_type =
'Health' OR insurance_type = 'Home' OR insurance_type = 'Auto'),
    SSN CHAR(11),
    FOREIGN KEY (SSN) REFERENCES client ON DELETE CASCADE
);
CREATE INDEX idx_insurance_type on insurance(insurance_type);


CREATE TABLE volunteer(
    SSN CHAR(11) PRIMARY KEY,
```

```
    date_joined DATE,
    last_training_date DATE,
    last_training_location VARCHAR(250),
    FOREIGN KEY (SSN) REFERENCES person ON DELETE CASCADE
);
CREATE INDEX idx_volunteer_SSN ON volunteer(SSN);


CREATE TABLE employee(
    SSN CHAR(11) PRIMARY KEY,
    salary DECIMAL(10, 2),
    marital_status VARCHAR(10),
    hire_date DATE,
    FOREIGN KEY (SSN) REFERENCES person ON DELETE CASCADE
);
CREATE INDEX idx_employee_SSN on employee(SSN);


CREATE TABLE expense(
    SSN CHAR(11),
    amount DECIMAL(10,2),
    expense_description VARCHAR(250),
    expense_date DATE,
    PRIMARY KEY (SSN, expense_date),
    FOREIGN KEY (SSN) REFERENCES employee ON DELETE CASCADE
);
CREATE INDEX idx_expense_date ON expense(expense_date);


CREATE TABLE team(
    team_name VARCHAR(100) PRIMARY KEY,
    team_type VARCHAR(50),
    date_formed DATE
);
CREATE INDEX idx_team_date_formed ON team(date_formed);


CREATE TABLE cares(
    client_SSN CHAR(11),
    team_name VARCHAR(100),
    isActive BIT, -- 1 is active, 0 is inactive
    PRIMARY KEY (client_SSN, team_name),
```

```
    FOREIGN KEY (client_SSN) REFERENCES client(SSN) ON DELETE CASCADE,
    FOREIGN KEY (team_name) REFERENCES team(team_name) ON DELETE CASCADE
);
CREATE INDEX idx_cares_client_team ON cares(client_SSN, team_name);


CREATE TABLE leads (
    volunteer_SSN CHAR(11),
    team_name VARCHAR(100),
    leads_hours INT,
    isActive BIT, -- 1 is active, 0 is inactive
    PRIMARY KEY (volunteer_SSN, team_name),
    FOREIGN KEY (volunteer_SSN) REFERENCES volunteer(SSN) ON DELETE CASCADE,
    FOREIGN KEY (team_name) REFERENCES team(team_name) ON DELETE CASCADE
);
CREATE INDEX idx_leads_volunteer_team ON leads (volunteer_SSN, team_name);


CREATE TABLE serves (
    volunteer_SSN CHAR(11),
    team_name VARCHAR(100),
    serves_hours INT,
    isActive BIT, -- 1 is active, 0 is inactive
    PRIMARY KEY (volunteer_SSN, team_name),
    FOREIGN KEY (volunteer_SSN) REFERENCES volunteer(SSN) ON DELETE CASCADE,
    FOREIGN KEY (team_name) REFERENCES team(team_name) ON DELETE CASCADE
);
CREATE INDEX idx_serves_volunteer_team ON serves (volunteer_SSN, team_name);


CREATE TABLE reports (
    employee_SSN CHAR(11),
    team_name VARCHAR(100),
    report_date DATE,
    report_description VARCHAR(250),
    PRIMARY KEY (employee_SSN, team_name),
    FOREIGN KEY (employee_SSN) REFERENCES employee(SSN) ON DELETE CASCADE,
    FOREIGN KEY (team_name) REFERENCES team(team_name) ON DELETE CASCADE
);
CREATE INDEX idx_reports_employee_team ON reports(employee_SSN, team_name);
```

```sql
CREATE TABLE donor (
   SSN CHAR(11) PRIMARY KEY,
   isAnon BIT, -- 1 for anon, 0 for not
   FOREIGN KEY (SSN) REFERENCES person(SSN) ON DELETE CASCADE
);
CREATE INDEX idx_donor_SSN ON donor (SSN);


CREATE TABLE check_donation (
   SSN CHAR(11),
   donation_date DATE,
   amount DECIMAL(10, 2),
   donation_type VARCHAR(50),
   campaign VARCHAR(100),
   check_number VARCHAR(50),
   PRIMARY KEY (SSN, donation_date),
   FOREIGN KEY (SSN) REFERENCES donor ON DELETE CASCADE
);
CREATE INDEX idx_check_donation_date ON check_donation(SSN, donation_date);


CREATE TABLE card_donation (
   SSN CHAR(11),
   donation_date DATE,
   amount DECIMAL(10, 2),
   donation_type VARCHAR(50),
   campaign VARCHAR(100),
   card_number CHAR(16),
   card_type VARCHAR(20),
   card_expiration_date DATE,
   PRIMARY KEY (SSN, donation_date),
   FOREIGN KEY (SSN) REFERENCES donor(SSN) ON DELETE CASCADE
);
CREATE INDEX idx_card_donation_date ON card_donation(SSN, donation_date);
```

Database after table creation:



Task 5.

5.1. SQL Statements Implementing All Queries

```
-- TASK 5A: Statements for all queries

IF OBJECT_ID('EnterTeam', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE EnterTeam; -- Drop the procedure
END;
GO

IF OBJECT_ID('EnterClient', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE EnterClient; -- Drop the procedure
END;
GO

IF OBJECT_ID('EnterVolunteer', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE EnterVolunteer; -- Drop the procedure
END;
GO

IF OBJECT_ID('EnterVolunteerHours', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE EnterVolunteerHours; -- Drop the procedure
END;
GO

IF OBJECT_ID('EnterEmployee', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE EnterEmployee; -- Drop the procedure
END;
GO

IF OBJECT_ID('EnterEmployeeExpense', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE EnterEmployeeExpense; -- Drop the procedure
END;
GO

IF OBJECT_ID('EnterDonorWithDonations', 'P') IS NOT NULL
```

```
BEGIN
   DROP PROCEDURE EnterDonorWithDonations; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetDoctorInfoByClient', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetDoctorInfoByClient; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetExpensesByTime', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetExpensesByTime; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetVolunteersByClient', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetVolunteersByClient; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetTeamsByDate', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetTeamsByDate; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetPeople', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetPeople; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetDonorEmployees', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetDonorEmployees; -- Drop the procedure
END;
GO

IF OBJECT_ID('IncreaseSalary', 'P') IS NOT NULL
```

```
BEGIN
   DROP PROCEDURE IncreaseSalary; -- Drop the procedure
END;
GO

IF OBJECT_ID('DeleteClients', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE DeleteClients; -- Drop the procedure
END;
GO

IF OBJECT_ID('GetMailingList', 'P') IS NOT NULL
BEGIN
   DROP PROCEDURE GetMailingList; -- Drop the procedure
END;
GO

-- QUERY 1: Enter a new team into the database
CREATE PROCEDURE EnterTeam
@team_name VARCHAR(100),
@team_type VARCHAR(50),
@date_formed DATE
AS
BEGIN
   INSERT INTO team (team_name, team_type, date_formed)
   VALUES (@team_name, @team_type, @date_formed);
END;
GO


-- QUERY 2: Enter a new client into the database and associate him or her with one or
more teamsEnter a new client into the database and associate him or her with one or
more teams
CREATE PROCEDURE EnterClient
@person_name VARCHAR(100),
@SSN CHAR(11),
@gender VARCHAR(6),
@profession VARCHAR(100),
@mailing_address VARCHAR(250),
@email_address VARCHAR(100),
@phone_number CHAR(10),
@inMailingList BIT,
```

```
@doctor_name VARCHAR(100),
@doctor_phone_number CHAR(10),
@date_assigned DATE,
@needs NVARCHAR(MAX), -- List of clients need in form need:value,...
@insurances NVARCHAR(MAX), -- List of insurance in form policyID,name,address,type|...
@teams NVARCHAR(MAX) -- List of team names to associate client with
AS
BEGIN
   -- Insert into person table
   IF NOT EXISTS (SELECT 1 FROM person WHERE SSN = @SSN)
   BEGIN
       INSERT INTO person (person_name, SSN, gender, profession, mailing_address,
email_address, phone_number, inMailingList)
       VALUES (@person_name, @SSN, @gender, @profession, @mailing_address,
@email_address, @phone_number, @inMailingList);
   END

   -- Insert into client table
   IF NOT EXISTS (SELECT 1 FROM client WHERE SSN = @SSN)
   BEGIN
       INSERT INTO client (SSN, doctor_name, doctor_phone_number, date_assigned)
       VALUES (@SSN, @doctor_name, @doctor_phone_number, @date_assigned);

       -- Insert clients needs to need table
       DECLARE @need NVARCHAR(200);
       DECLARE @need_name NVARCHAR(100);
       DECLARE @importance_value NVARCHAR(20);
       WHILE CHARINDEX(',', @needs) > 0 -- Iterate over list of needs need:value
       BEGIN
           SET @need = LEFT(@needs, CHARINDEX(',', @needs) - 1);
           SET @needs = SUBSTRING(@needs, CHARINDEX(',', @needs) + 1, LEN(@needs));
           SET @need_name = LEFT(@need, CHARINDEX(':', @need) - 1);
           SET @importance_value = SUBSTRING(@need, CHARINDEX(':', @need) + 1,
LEN(@need) - CHARINDEX(':', @need));

           -- Insert into need table
           INSERT INTO need (SSN, need_name, importance_value)
           VALUES (@SSN, @need_name, @importance_value);
       END;

       IF LEN(@needs) > 0 -- May be one entry left without comma at end
       BEGIN
```

```
            SET @need = @needs;
            SET @need_name = LEFT(@need, CHARINDEX(':', @need) - 1);
            SET @importance_value = SUBSTRING(@need, CHARINDEX(':', @need) + 1,
LEN(@need) - CHARINDEX(':', @need));

            -- Insert into need table
            INSERT INTO need (SSN, need_name, importance_value)
            VALUES (@SSN, @need_name, @importance_value);
        END;

        -- Insert clients insurance to insurance table
        DECLARE @insurance NVARCHAR(500);
        DECLARE @pos INT;
        DECLARE @delim CHAR(1) = '|';

        WHILE LEN(@insurances) > 0
        BEGIN
            SET @pos = CHARINDEX(@delim, @insurances);
            IF @pos > 0
            BEGIN
                SET @insurance = LEFT(@insurances, @pos - 1);
                SET @insurances = SUBSTRING(@insurances, @pos + 1, LEN(@insurances));
            END
            ELSE
            BEGIN
                SET @insurance = @insurances;
                SET @insurances = '';
            END
            DECLARE @policy_ID VARCHAR(20),
                    @provider_name VARCHAR(50),
                    @provider_address VARCHAR(250),
                    @insurance_type VARCHAR(6)
            SET @policy_ID = SUBSTRING(@insurance, 1, CHARINDEX(',', @insurance) - 1);
            SET @insurance = SUBSTRING(@insurance, CHARINDEX(',', @insurance) + 1,
LEN(@insurance));

            SET @provider_name = SUBSTRING(@insurance, 1, CHARINDEX(',', @insurance) -
1);
            SET @insurance = SUBSTRING(@insurance, CHARINDEX(',', @insurance) + 1,
LEN(@insurance));
```

```
            SET @provider_address = SUBSTRING(@insurance, 1, CHARINDEX(',', @insurance)
- 1);
            SET @insurance = SUBSTRING(@insurance, CHARINDEX(',', @insurance) + 1,
LEN(@insurance));

            SET @insurance_type = @insurance;

            INSERT INTO insurance (policy_ID, provider_name, provider_address,
insurance_type, SSN)
            VALUES (@policy_ID, @provider_name, @provider_address, @insurance_type,
@SSN)
        END

        -- Associate client with teams through 'cares' table
        DECLARE @team_name NVARCHAR(100);
        WHILE CHARINDEX(',', @teams) > 0 -- Iterate over list of teams
        BEGIN
            SET @team_name = LEFT(@teams, CHARINDEX(',', @teams) - 1); -- Extracts
first team name
            SET @teams = SUBSTRING(@teams, CHARINDEX(',', @teams) + 1, LEN(@teams)); --
Removes processed team from list

            INSERT INTO cares (client_SSN, team_name, isActive)
            VALUES (@SSN, @team_name, 1);
        END;

        -- Associate with last team
        IF LEN(@teams) > 0 -- May be one left without comma at end
        BEGIN
            INSERT INTO cares (client_SSN, team_name, isActive)
            VALUES (@SSN, @teams, 1);
        END;
    END
    ELSE
    BEGIN
        PRINT 'Client already exists'
    END
END;
GO
```

```
-- QUERY 3: Enter a new volunteer into the database and associate him or her with one
or more teams
CREATE PROCEDURE EnterVolunteer
@person_name VARCHAR(100),
@SSN CHAR(11),
@gender VARCHAR(6),
@profession VARCHAR(100),
@mailing_address VARCHAR(250),
@email_address VARCHAR(100),
@phone_number CHAR(10),
@inMailingList BIT,
@date_joined DATE,
@last_training_date DATE,
@last_training_location VARCHAR(250),
@teams_roles NVARCHAR(MAX) -- List of team names and roles to associate volunteer with
(format name1:role1,name2:role2,...)
AS
BEGIN
   -- Insert into person table
   IF NOT EXISTS (SELECT 1 FROM person WHERE SSN = @SSN)
   BEGIN
       INSERT INTO person (person_name, SSN, gender, profession, mailing_address,
email_address, phone_number, inMailingList)
       VALUES (@person_name, @SSN, @gender, @profession, @mailing_address,
@email_address, @phone_number, @inMailingList);
   END

   -- Insert into volunteer table
   IF NOT EXISTS (SELECT 1 FROM volunteer WHERE SSN = @SSN)
   BEGIN
       INSERT INTO volunteer (SSN, date_joined, last_training_date,
last_training_location)
       VALUES (@SSN, @date_joined, @last_training_date, @last_training_location);

       -- Associate volunteer with teams through either 'serves' or 'leads' tables
       DECLARE @team_role NVARCHAR(200);
       DECLARE @team_name NVARCHAR(100);
       DECLARE @role NVARCHAR(20);
       WHILE CHARINDEX(',', @teams_roles) > 0 -- Iterate over list of teams:roles
       BEGIN
           SET @team_role = LEFT(@teams_roles, CHARINDEX(',', @teams_roles) - 1); --
Extracts first pair team:role
```

```
            SET @teams_roles = SUBSTRING(@teams_roles, CHARINDEX(',', @teams_roles) +
1, LEN(@teams_roles));
            SET @team_name = LEFT(@team_role, CHARINDEX(':', @team_role) - 1); --
Extracts team name from team:role
            SET @role = SUBSTRING(@team_role, CHARINDEX(':', @team_role) + 1,
LEN(@team_role) - CHARINDEX(':', @team_role)); -- Extracts role from team:role

            -- Insert into table based on role
            IF @role = 'leads'
            BEGIN
                INSERT INTO leads (volunteer_SSN, team_name, leads_hours, isActive)
                VALUES (@SSN, @team_name, 0, 1); -- Inital 0 hours
            END
            ELSE IF @role = 'serves'
            BEGIN
                INSERT INTO serves (volunteer_SSN, team_name, serves_hours, isActive)
                VALUES (@SSN, @team_name, 0, 1); -- Initial 0 hours
            END
        END;

        -- Associate with last team
        IF LEN(@teams_roles) > 0 -- May be one entry left without comma at end
        BEGIN
            SET @team_role = @teams_roles;
            SET @team_name = LEFT(@team_role, CHARINDEX(':', @team_role) - 1);
            SET @role = SUBSTRING(@team_role, CHARINDEX(':', @team_role) + 1,
LEN(@team_role) - CHARINDEX(':', @team_role));

            -- Insert into table based on role
            IF @role = 'leads'
            BEGIN
                INSERT INTO leads (volunteer_SSN, team_name, leads_hours, isActive)
                VALUES (@SSN, @team_name, 0, 1); -- Initial 0 hours
            END
            ELSE IF @role = 'serves'
            BEGIN
                INSERT INTO serves (volunteer_SSN, team_name, serves_hours, isActive)
                VALUES (@SSN, @team_name, 0, 1); -- Initial 0 hours
            END
        END;
    END;
END;
```

```
GO


-- QUERY 4: Enter the number of hours a volunteer worked this month for a particular
team
CREATE PROCEDURE EnterVolunteerHours
@volunteer_SSN CHAR(11),
@team_name VARCHAR(100),
@hours INT
AS
BEGIN
   -- Try to insert into the 'leads' table first
   IF EXISTS (SELECT 1 FROM leads WHERE volunteer_SSN = @volunteer_SSN AND team_name =
@team_name)
   BEGIN
       -- Insert into the 'leads' table (volunteer is leading on this team)
       UPDATE leads
       SET leads_hours = leads_hours + @hours -- Add to existing hours if applicable
       WHERE volunteer_SSN = @volunteer_SSN AND team_name = @team_name;
   END
   ELSE IF EXISTS (SELECT 1 FROM serves WHERE volunteer_SSN = @volunteer_SSN AND
team_name = @team_name)
   BEGIN
       -- Insert into the 'serves' table (volunteer is serving on this team)
       UPDATE serves
       SET serves_hours = serves_hours + @hours -- Add to existing hours if applicable
       WHERE volunteer_SSN = @volunteer_SSN AND team_name = @team_name;
   END
   ELSE
   BEGIN
       PRINT 'Volunteer is not currently assigned to the given team. Please ensure
they are properly assigned first.';
   END
END;
GO


-- QUERY 5: Enter a new employee into the database and associate him or her with one
or more teams
CREATE PROCEDURE EnterEmployee
@person_name VARCHAR(100),
@SSN CHAR(11),
```

```
@gender VARCHAR(6),
@profession VARCHAR(100),
@mailing_address VARCHAR(250),
@email_address VARCHAR(100),
@phone_number CHAR(10),
@inMailingList BIT,
@salary DECIMAL(10, 2),
@marital_status VARCHAR(10),
@hire_date DATE,
@teams NVARCHAR(MAX) -- List of team names to associate employee with
AS
BEGIN
   -- Insert into person table
   IF NOT EXISTS (SELECT 1 FROM person WHERE SSN = @SSN)
   BEGIN
       INSERT INTO person (person_name, SSN, gender, profession, mailing_address,
email_address, phone_number, inMailingList)
       VALUES (@person_name, @SSN, @gender, @profession, @mailing_address,
@email_address, @phone_number, @inMailingList);
   END

   -- Insert into employee table
   IF NOT EXISTS (SELECT 1 FROM employee WHERE SSN = @SSN)
   BEGIN
       INSERT INTO employee (SSN, salary, marital_status, hire_date)
       VALUES (@SSN, @salary, @marital_status, @hire_date);

   -- Associate employee with teams through 'reports' table
   DECLARE @team_name NVARCHAR(100);
   WHILE CHARINDEX(',', @teams) > 0 -- Iterate over list of teams
   BEGIN
       SET @team_name = LEFT(@teams, CHARINDEX(',', @teams) - 1); -- Extracts first
team name
       SET @teams = SUBSTRING(@teams, CHARINDEX(',', @teams) + 1, LEN(@teams)); --
Removes processed team from list

       INSERT INTO reports (employee_SSN, team_name)
       VALUES (@SSN, @team_name);
   END;

   -- Associate with last team
   IF LEN(@teams) > 0 -- May be one left without comma at end
```

```
    BEGIN
        INSERT INTO reports (employee_SSN, team_name)
        VALUES (@SSN, @teams);
    END;
    END;
END;
GO


-- QUERY 6: Enter an expense charged by an employee
CREATE PROCEDURE EnterEmployeeExpense
@SSN CHAR(11),
@amount DECIMAL(10, 2),
@expense_description VARCHAR(250),
@expense_date DATE
AS
BEGIN
    INSERT INTO expense (SSN, amount, expense_description, expense_date)
    VALUES (@SSN, @amount, @expense_description, @expense_date);
END;
GO


-- QUERY 7: Enter a new donor and associate with donations
CREATE PROCEDURE EnterDonorWithDonations
@person_name NVARCHAR(100),
@SSN CHAR(11),
@gender NVARCHAR(6),
@profession NVARCHAR(100),
@mailing_address NVARCHAR(250),
@email_address NVARCHAR(100),
@phone_number CHAR(10),
@inMailingList BIT,
@isAnon BIT,
@donations NVARCHAR(MAX)
AS
BEGIN
    -- Insert into Person table if doesn't exist
    IF NOT EXISTS (SELECT 1 FROM person WHERE SSN = @SSN)
    BEGIN
        INSERT INTO person (SSN, person_name, gender, profession, mailing_address,
email_address, phone_number, inMailingList)
```

```
        VALUES (@SSN, @person_name, @gender, @profession, @mailing_address,
@email_address, @phone_number, @inMailingList);
    END

    -- Insert into Donor table
    IF NOT EXISTS (SELECT 1 FROM donor WHERE SSN = @SSN)
    BEGIN
        INSERT INTO donor (SSN, isAnon)
        VALUES (@SSN, @isAnon);

    -- Associate donor with donations (check or card)
    DECLARE @donation NVARCHAR(500);
    DECLARE @pos INT;
    DECLARE @delim CHAR(1) = '|';

    WHILE LEN(@donations) > 0
    BEGIN
        SET @pos = CHARINDEX(@delim, @donations);
        IF @pos > 0
        BEGIN
            SET @donation = LEFT(@donations, @pos - 1);
            SET @donations = SUBSTRING(@donations, @pos + 1, LEN(@donations));
        END
        ELSE
        BEGIN
            SET @donation = @donations;
            SET @donations = '';
        END

        DECLARE @donation_date DATE,
                @amount DECIMAL(10, 2),
                @donation_type NVARCHAR(50),
                @campaign NVARCHAR(100),
                @check_number NVARCHAR(50) = NULL,
                @card_number CHAR(16) = NULL,
                @card_type NVARCHAR(20) = NULL,
                @card_expiration_date DATE = NULL;
        SET @donation_date = CONVERT(DATE, SUBSTRING(@donation, 1, CHARINDEX(',',
@donation) - 1));  -- First part: donation_date
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove donation_date part
```

```
        SET @amount = CONVERT(DECIMAL(10, 2), SUBSTRING(@donation, 1, CHARINDEX(',',
@donation) - 1)); -- Second part: amount
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove amount part

        SET @donation_type = SUBSTRING(@donation, 1, CHARINDEX(',', @donation) - 1); --
Third part: donation_type
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove donation_type part

        SET @campaign = SUBSTRING(@donation, 1, CHARINDEX(',', @donation) - 1); --
Fourth part: campaign
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove campaign part

        SET @check_number = SUBSTRING(@donation, 1, CHARINDEX(',', @donation) - 1); --
Fifth part: check_number (could be NULL)
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove check_number part

        SET @card_number = SUBSTRING(@donation, 1, CHARINDEX(',', @donation) - 1); --
Sixth part: card_number (could be NULL)
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove card_number part

        SET @card_type = SUBSTRING(@donation, 1, CHARINDEX(',', @donation) - 1); --
Seventh part: card_type (could be NULL)
        SET @donation = SUBSTRING(@donation, CHARINDEX(',', @donation) + 1,
LEN(@donation)); -- Remove card_type part

        SET @card_expiration_date = CONVERT(DATE, @donation); -- Eighth part:
card_expiration_date

        IF @donation_type = 'Check'
        BEGIN
            INSERT INTO check_donation (SSN, donation_date, amount, donation_type,
campaign, check_number)
            VALUES (@SSN, @donation_date, @amount, @donation_type, @campaign,
@check_number);
        END
        ELSE
        BEGIN
```

```
            INSERT INTO card_donation (SSN, donation_date, amount, donation_type,
campaign, card_number, card_type, card_expiration_date)
            VALUES (@SSN, @donation_date, @amount, @donation_type, @campaign,
@card_number, @card_type, @card_expiration_date);
        END
    END;
    END;
END;
GO


-- QUERY 8: Retrieve the name and phone number of the doctor of a particular client
CREATE PROCEDURE GetDoctorInfoByClient
@SSN CHAR(11)
AS
BEGIN
    -- Retrieve the doctor's name and phone number for the specified client
    SELECT doctor_name, doctor_phone_number
    FROM client
    WHERE SSN = @SSN;
END;
GO


-- QUERY 9: Retrieve the total amount of expenses charged by each employee for a
particular period of time
CREATE PROCEDURE GetExpensesByTime
@start_date DATE,
@end_date DATE
AS
BEGIN
    SELECT e.SSN, SUM(amount) AS total_expenses
    FROM expense e
    WHERE expense_date BETWEEN @start_date AND @end_date
    GROUP BY e.SSN
    ORDER BY total_expenses DESC;
END;
GO

-- QUERY 10: Retrieve the list of volunteers that are members of teams that support a
particular client
CREATE PROCEDURE GetVolunteersByClient
```

```
@SSN CHAR(11)
AS
BEGIN
   SELECT v.SSN, v.date_joined
   FROM serves s
   JOIN cares c ON s.team_name = c.team_name
   JOIN volunteer v ON s.volunteer_SSN = v.SSN
   WHERE c.client_SSN = @SSN

   UNION

   -- Select volunteers from 'leads' table
   SELECT v.SSN, v.date_joined
   FROM leads l
   JOIN cares c ON l.team_name = c.team_name
   JOIN volunteer v ON l.volunteer_SSN = v.SSN
   WHERE c.client_SSN = @SSN;
END;
GO

-- QUERY 11: Retrieve the names of all teams that were founded after a particular date
CREATE PROCEDURE GetTeamsByDate
@date DATE
AS
BEGIN
   SELECT team_name
   FROM team
   WHERE date_formed > @date;
END;
GO


-- QUERY 12: Retrieve the names, social security numbers, contact information, and
emergency contact information of all people in the database
CREATE PROCEDURE GetPeople
AS
BEGIN
   SELECT p.person_name, p.SSN, p.email_address, p.phone_number
   FROM person p;
END;
GO
```

```sql
-- QUERY 13: Retrieve the name and total amount donated by donors that are also employees
CREATE PROCEDURE GetDonorEmployees
AS
BEGIN
   SELECT p.person_name,
       d.SSN,
       SUM(ISNULL(cd.amount, 0) + ISNULL(chd.amount, 0)) AS total_donations,
       d.isAnon
   FROM donor d
   JOIN person p ON d.SSN = p.SSN
   LEFT JOIN card_donation cd ON d.SSN = cd.SSN
   LEFT JOIN check_donation chd ON d.SSN = chd.SSN
   WHERE d.SSN IN (SELECT SSN FROM employee)
   GROUP BY p.person_name, d.SSN, d.isAnon
   ORDER BY total_donations DESC;
END;
GO



-- QUERY 14: Increase the salary by 10% of all employees to whom more than one team must report
CREATE PROCEDURE IncreaseSalary
AS
BEGIN
   UPDATE employee
   SET salary = salary * 1.10
   WHERE SSN IN (
       SELECT employee_SSN
       FROM reports
       GROUP BY employee_SSN
       HAVING COUNT(DISTINCT team_name) > 1
   );
END;
GO


-- QUERY 15: Delete all clients who do not have health insurance and whose value of importance for transportation is less than 5
CREATE PROCEDURE DeleteClients
```

```
AS
BEGIN
   DELETE FROM client
   WHERE SSN NOT IN (
            SELECT SSN
            FROM insurance
            WHERE insurance_type = 'Health'
        )
   AND SSN IN (
        SELECT SSN
        FROM need
        WHERE need_name = 'transportation' AND importance_value < 5
   );
END;
GO

-- QUERY 17: Retrieve names and mailing addresses of people in mailing list
CREATE PROCEDURE GetMailingList
AS
BEGIN
   SELECT person_name, mailing_address
   FROM person
   WHERE inMailingList = 1;
END;
GO
```

5.2. Java Source Program and Screenshots Showing Successful Compilation

```java
package org.dbms.group54;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.CallableStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;
import java.util.Scanner;
import java.util.logging.Logger;
import java.io.*;

public class IP_Task5b {
    private static final Logger log; // Logger for the application

    static {
        System.setProperty("java.util.logging.SimpleFormatter.format", "[%4$-7s] %5$s %n"); // Set log format
        log = Logger.getLogger(IP_Task5b.class.getName()); // Initialize the logger
    }

    public static void main(String[] args) throws Exception {
        log.info("Starting the application"); // Log application start
        log.info("Loading application properties"); // Log loading properties

        Properties properties = new Properties(); // Load database properties
        properties.load(IP_Task5b.class.getClassLoader().getResourceAsStream("application.properties")); // Load properties file

        log.info("Connecting to the database"); // Log database connection attempt
        Connection connection = DriverManager.getConnection(properties.getProperty("url"), properties); // Establish connection
        log.info("Database connection test: " + connection.getCatalog()); // Test the connection

        Scanner keyboard = new Scanner(System.in); // Initialize scanner for user input
        int query = 0; // Initialize query input variable

        System.out.println("WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM\n\nChoose an option:\n"); // Prompt user
        System.out.println("(1) Enter a new team into the database\n(2) Enter a new client into the database and associate him or her
        System.out.println("(4) Enter the number of hours a volunteer worked this month for a particular team\n(5) Enter a new employe
        System.out.println("(7) Enter a new donor and associate him or her with several donations\n(8) Retrieve the name and phone nur
                + "time\n(10) Retrieve the list of volunteers that are members of teams that support a particular client\n(11) Retriev
                + "information of all people in the database \n(13) Retrieve the name and total amount donated by donors that are alsc
                + "transportation is less than 5\n(16) Import: enter new teams from a data file until the file is empty\n(17) Export:
                + "output them to a data file \n(18) Quit.\n");

        while (query != 18) { // Loop until user selects option 18
            System.out.println("Choose an option (1-18): ");
            query = keyboard.nextInt(); // Get user input

            switch (query) { // Handle different query options
                case 1:
                    executeQuery1(connection, keyboard); // Execute query 1
                    break;
                case 2:
                    executeQuery2(connection, keyboard); // Execute query 2
                    break;
                case 3:
                    executeQuery3(connection, keyboard); // Execute query 3
                    break;
                case 4:
                    executeQuery4(connection, keyboard); // Execute query 4
                    break;
                case 5:
                    executeQuery5(connection, keyboard); // Execute query 5
                    break;
                case 6:
                    executeQuery6(connection, keyboard); // Execute query 6
                    break;
                case 7:
                    executeQuery7(connection, keyboard); // Execute query 7
                    break;
                case 8:
                    executeQuery8(connection, keyboard); // Execute query 8
                    break;
                case 9:
                    executeQuery9(connection, keyboard); // Execute query 9
                    break;
                case 10:
                    executeQuery10(connection, keyboard); // Execute query 10
                    break;
                case 11:
                    executeQuery11(connection, keyboard); // Execute query 11
                    break;
                case 12:
                    executeQuery12(connection); // Execute query 12
                    break;
                case 13:
                    executeQuery13(connection); // Execute query 13
                    break;
                case 14:
```

```
                case 14:
                    executeQuery14(connection); // Execute query 14
                    break;
                case 15:
                    executeQuery15(connection); // Execute query 15
                    break;
                case 16:
                    executeImportTeams(connection, keyboard); // Execute query 15
                    break;
                case 17:
                    executeExportMailing(connection, keyboard); // Execute query 15
                    break;
                case 18:
                    System.out.println("Terminating program."); // Exit program
                    break;
                default:
                    System.out.println("Invalid option. Please select a valid option (1-18)."); // Handle invalid input
            }
        }

        connection.close(); // Close database connection
    }

    private static void executeQuery1(Connection connection, Scanner keyboard) {
        keyboard.nextLine();
        System.out.println("Enter team name: "); // Getting team name
        String teamName = keyboard.nextLine();

        System.out.println("Enter team type: "); // Getting team type
        String teamType = keyboard.next();

        System.out.println("Enter date formed in format yyyy-MM-dd: "); // Getting date formed
        String dateFormed = keyboard.next();


        String sql = "{call EnterTeam(?, ?, ?)}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            // Setting parameters
            stmt.setString(1, teamName);
            stmt.setString(2, teamType);
            stmt.setString(3, dateFormed);

            stmt.execute(); // Execute the stored procedure
```

```
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }

    private static void executeQuery2(Connection connection, Scanner keyboard) {

        keyboard.nextLine();
        System.out.println("Enter client's name: "); // Getting name
        String personName = keyboard.nextLine();

        System.out.println("Enter client's SSN: "); // Getting SSN
        String SSN = keyboard.next();

        System.out.println("Enter client's gender: "); // Getting gender
        String gender = keyboard.next();

        keyboard.nextLine();
        System.out.println("Enter client's profession: "); // Getting profession
        String profession = keyboard.nextLine();

        keyboard.nextLine();
        System.out.println("Enter client's mailing address: "); // Getting mail address
        String mailingAddress = keyboard.nextLine();

        System.out.println("Enter client's email address: "); // Getting email address
        String emailAddress = keyboard.next();

        System.out.println("Enter client's phone number: "); // Getting phone number
        String phoneNumber = keyboard.next();

        System.out.println("Is client in mailing list? 1 for yes, 0 for no: "); // Getting mailing list
        int inMailingList = keyboard.nextInt();

        keyboard.nextLine();
        System.out.println("Enter client's doctor's name: "); // Getting doctor name
        String doctorName = keyboard.nextLine();

        System.out.println("Enter client's doctor's phone number: "); // Getting doctor phone
        String doctorPhone = keyboard.next();
```

```
        System.out.println("Enter client's date assign to doctor in yyyy-MM-dd format: "); // Getting date assigned
        String dateAssigned = keyboard.next();

        System.out.println("Enter client's needs in need1:value1,need2:value2,... format: "); // Getting needs
        String needs = keyboard.next();

        keyboard.nextLine();
        System.out.println("Enter client's insurances in policyID1,name1,address1,type1|policyID2,name2,address2,type2|.. format "); 
        String insurances = keyboard.nextLine();

        System.out.println("Enter client's teams in team1,team2,... format "); // Getting teams
        String teams = keyboard.nextLine();

        String sql = "{call EnterClient(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            // Setting parameters
            stmt.setString(1, personName);
            stmt.setString(2, SSN);
            stmt.setString(3, gender);
            stmt.setString(4, profession);
            stmt.setString(5, mailingAddress);
            stmt.setString(6, emailAddress);
            stmt.setString(7, phoneNumber);
            stmt.setInt(8, inMailingList);
            stmt.setString(9, doctorName);
            stmt.setString(10, doctorPhone);
            stmt.setString(11, dateAssigned);
            stmt.setString(12, needs);
            stmt.setString(13, insurances);
            stmt.setString(14, teams);

            stmt.execute(); // Execute the stored procedure
            System.out.println("Client inserted successfully."); // Confirm success
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }

    private static void executeQuery3(Connection connection, Scanner keyboard) {
        keyboard.nextLine();
        System.out.println("Enter volunteer's name: "); // Getting name
        String personName = keyboard.nextLine();

        System.out.println("Enter volunteer's SSN: "); // Getting SSN
        String SSN = keyboard.next();

        System.out.println("Enter volunteer's gender: "); // Getting gender
        String gender = keyboard.next();

        keyboard.nextLine();
        System.out.println("Enter volunteer's profession: "); // Getting profession
        String profession = keyboard.nextLine();

        System.out.println("Enter volunteer's mailing address: "); // Getting mail address
        String mailingAddress = keyboard.nextLine();

        System.out.println("Enter volunteer's email address: "); // Getting email address
        String emailAddress = keyboard.next();

        System.out.println("Enter volunteer's phone number: "); // Getting phone number
        String phoneNumber = keyboard.next();

        System.out.println("Is volunteer in mailing list? 1 for yes, 0 for no: "); // Getting mailing list
        int inMailingList = keyboard.nextInt();

        keyboard.nextLine();
        System.out.println("Enter volunteer's date joined in yyyy-MM-dd format: "); // Getting date joined
        String dateJoined = keyboard.next();

        System.out.println("Enter volunteer's last training date in yyyy-MM-dd format: "); // Getting training date
        String lastTrainingDate = keyboard.next();

        System.out.println("Enter volunteer's last training location:  "); // Getting training location
        String lastTrainingLocation = keyboard.next();

        keyboard.nextLine();
        System.out.println("Enter volunteers's teams/roles in team1:role1,team2:role2,... format: "); // Getting teams/roles
        String teams = keyboard.nextLine();


        String sql = "{call EnterVolunteer(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
```

```
        String sql = "{call EnterVolunteer(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            // Setting parameters
            stmt.setString(1, personName);
            stmt.setString(2, SSN);
            stmt.setString(3, gender);
            stmt.setString(4, profession);
            stmt.setString(5, mailingAddress);
            stmt.setString(6, emailAddress);
            stmt.setString(7, phoneNumber);
            stmt.setInt(8, inMailingList);
            stmt.setString(9, dateJoined);
            stmt.setString(10, lastTrainingDate);
            stmt.setString(11, lastTrainingLocation);
            stmt.setString(12, teams);

            stmt.execute(); // Execute the stored procedure
            System.out.println("Volunteer inserted successfully."); // Confirm success
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }

    private static void executeQuery4(Connection connection, Scanner keyboard) {
        System.out.println("Enter volunteer's SSN: "); // Getting team name
        String SSN = keyboard.next();

        keyboard.nextLine();
        System.out.println("Enter team name: "); // Getting team type
        String teamName = keyboard.nextLine();

        System.out.println("Enter number of hours to add: "); // Getting date formed
        int hours = keyboard.nextInt();

        String sql = "{call EnterVolunteerHours(?, ?, ?)}"; // Prepare the SQL call for the procedure
        try(CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            // Setting parameters
```

```
            // Setting parameters
            stmt.setString(1, SSN);
            stmt.setString(2, teamName);
            stmt.setInt(3, hours);

            stmt.execute(); // Execute the stored procedure
            System.out.println("Volunteer hours inserted successfully."); // Confirm success
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }

    private static void executeQuery5(Connection connection, Scanner keyboard) {
        keyboard.nextLine();
        System.out.println("Enter employee's name: "); // Getting name
        String personName = keyboard.nextLine();

        System.out.println("Enter employee's SSN: "); // Getting SSN
        String SSN = keyboard.next();

        System.out.println("Enter employee's gender: "); // Getting gender
        String gender = keyboard.next();

        keyboard.nextLine();
        System.out.println("Enter employee's profession: "); // Getting profession
        String profession = keyboard.nextLine();

        System.out.println("Enter employee's mailing address: "); // Getting mailing address
        String mailingAddress = keyboard.nextLine();

        System.out.println("Enter employee's email address: "); // Getting email address
        String emailAddress = keyboard.next();

        System.out.println("Enter employee's phone number: "); // Getting phone number
        String phoneNumber = keyboard.next();

        System.out.println("Is employee in mailing list? 1 for yes, 0 for no: "); // Getting mailing list
        int inMailingList = keyboard.nextInt();

        System.out.println("Enter employee's salary: "); // Getting salary
```

```
350         double salary = keyboard.nextDouble();
351
352         System.out.println("Enter employee's marital status:  "); // Getting marital status
353         String maritalStatus = keyboard.next();
354
355         System.out.println("Enter employee's hire date in yyyy-MM-dd format:  "); // Getting hire date
356         String hireDate = keyboard.next();
357
358         keyboard.nextLine();
359         System.out.println("Enter employee's teams in team1,team2,... format: "); // Getting teams
360         String teams = keyboard.nextLine();
361
362
363         String sql = "{call EnterEmployee(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}"; // Prepare the SQL call for the procedure
364         try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
365
366             // Setting parameters
367             stmt.setString(1, personName);
368             stmt.setString(2, SSN);
369             stmt.setString(3, gender);
370             stmt.setString(4, profession);
371             stmt.setString(5, mailingAddress);
372             stmt.setString(6, emailAddress);
373             stmt.setString(7, phoneNumber);
374             stmt.setInt(8, inMailingList);
375             stmt.setDouble(9, salary);
376             stmt.setString(10, maritalStatus);
377             stmt.setString(11, hireDate);
378             stmt.setString(12, teams);
379
380             stmt.execute(); // Execute the stored procedure
381             System.out.println("Employee inserted successfully."); // Confirm success
382             stmt.close(); // Close the statement
383         } catch (SQLException e) {
384             System.err.println("SQL Error occurred:");
385             System.err.println("SQL State: " + e.getSQLState());
386             System.err.println("Error Code: " + e.getErrorCode());
387             System.err.println("Message: " + e.getMessage());
388         }
389     }
390
391     private static void executeQuery6(Connection connection, Scanner keyboard) {
392         System.out.println("Enter employee's SSN: "); // Getting employee SSN
393         String SSN = keyboard.next();
```

```
391     private static void executeQuery6(Connection connection, Scanner keyboard) {
392         System.out.println("Enter employee's SSN: "); // Getting employee SSN
393         String SSN = keyboard.next();
394
395         keyboard.nextLine();
396         System.out.println("Enter expense amount: "); // Getting expense amount
397         double amount = keyboard.nextDouble();
398
399         keyboard.nextLine();
400         System.out.println("Enter expense description: "); // Getting expense description
401         String description = keyboard.nextLine();
402
403         System.out.println("Enter expense date in yyyy-MM-dd format: "); // Getting expense date
404         String date = keyboard.next();
405
406         String sql = "{call EnterEmployeeExpense(?, ?, ?, ?)}"; // Prepare the SQL call for the procedure
407         try(CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
408
409             // Setting parameters
410             stmt.setString(1, SSN);
411             stmt.setDouble(2, amount);
412             stmt.setString(3, description);
413             stmt.setString(4, date);
414
415             stmt.execute(); // Execute the stored procedure
416             System.out.println("Employee expense inserted successfully."); // Confirm success
417             stmt.close(); // Close the statement
418         } catch (SQLException e) {
419             System.err.println("SQL Error occurred:");
420             System.err.println("SQL State: " + e.getSQLState());
421             System.err.println("Error Code: " + e.getErrorCode());
422             System.err.println("Message: " + e.getMessage());
423         }
424     }
425
426     private static void executeQuery7(Connection connection, Scanner keyboard) {
427         keyboard.nextLine();
428         System.out.println("Enter donor's name: "); // Getting donor name
429         String personName = keyboard.nextLine();
430
431         System.out.println("Enter donor's SSN: "); // Getting donor SSN
432         String SSN = keyboard.next();
433
434         System.out.println("Enter donor's gender: "); // Getting donor gender
```

```
435         String gender = keyboard.next();
436
437         keyboard.nextLine();
438         System.out.println("Enter donor's profession: "); // Getting donor profession
439         String profession = keyboard.nextLine();
440
441         System.out.println("Enter donor's mailing address: "); // Getting donor mail address
442         String mailingAddress = keyboard.nextLine();
443
444         System.out.println("Enter donor's email address: "); // Getting donor email address
445         String emailAddress = keyboard.next();
446
447         System.out.println("Enter donor's phone number: "); // Getting donor phone number
448         String phoneNumber = keyboard.next();
449
450         System.out.println("Is donor in mailing list? 1 for yes, 0 for no: "); // Getting inMailingList
451         int inMailingList = keyboard.nextInt();
452
453         System.out.println("Is donor anonymous? 1 for yes, 0 for n "); // Getting isAnon
454         int isAnon = keyboard.nextInt();
455
456         keyboard.nextLine();
457         System.out.println("Enter donations in format date,amount,type,campaign,check#,card#,cardtype,cardexpdate|...:  "); // Gettin
458         String donations = keyboard.nextLine();
459
460         String sql = "{call EnterDonorWithDonations(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}"; // Prepare the SQL call for the procedure
461         try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
462
463             // Setting parameters
464             stmt.setString(1, personName);
465             stmt.setString(2, SSN);
466             stmt.setString(3, gender);
467             stmt.setString(4, profession);
468             stmt.setString(5, mailingAddress);
469             stmt.setString(6, emailAddress);
470             stmt.setString(7, phoneNumber);
471             stmt.setInt(8, inMailingList);
472             stmt.setInt(9, isAnon);
473             stmt.setString(10, donations);
474
475             stmt.execute(); // Execute the stored procedure
476             System.out.println("Donor and donations inserted successfully."); // Confirm success
477             stmt.close(); // Close the statement
478         } catch (SQLException e) {
```

```
478         } catch (SQLException e) {
479             System.err.println("SQL Error occurred:");
480             System.err.println("SQL State: " + e.getSQLState());
481             System.err.println("Error Code: " + e.getErrorCode());
482             System.err.println("Message: " + e.getMessage());
483         }
484     }
485
486     private static void executeQuery8(Connection connection, Scanner keyboard) {
487         System.out.println("Enter client's SSN: "); // Getting client SSN
488         String SSN = keyboard.next();
489
490         String sql = "{call GetDoctorInfoByClient(?)}"; // Prepare the SQL call for the procedure
491         try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
492             // Setting parameters
493             stmt.setString(1, SSN);
494
495             ResultSet rs = stmt.executeQuery(); // Execute the stored procedure
496
497             while (rs.next()) { // Iterate through the result set
498                 System.out.println("Doctor Name: " + rs.getString("doctor_name") + ", Doctor Phone Number: " + rs.getString("doctor_ph
499             }
500             rs.close();
501             stmt.close(); // Close the statement
502         } catch (SQLException e) {
503             System.err.println("SQL Error occurred:");
504             System.err.println("SQL State: " + e.getSQLState());
505             System.err.println("Error Code: " + e.getErrorCode());
506             System.err.println("Message: " + e.getMessage());
507         }
508     }
509
510     private static void executeQuery9(Connection connection, Scanner keyboard) {
511         System.out.println("Enter start date in yyyy-MM-dd format: "); // Getting start date
512         String startDate = keyboard.next();
513
514         System.out.println("Enter end date in yyyy-MM-dd format: "); // Getting end date
515         String endDate = keyboard.next();
516
517         String sql = "{call GetExpensesByTime(?, ?)}"; // Prepare the SQL call for the procedure
518         try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
519
520             // Setting parameters
```

```
            // Setting parameters
            stmt.setString(1, startDate);
            stmt.setString(2, endDate);

            ResultSet rs = stmt.executeQuery(); // Execute the stored procedure

            while (rs.next()) { // Iterate through the result set
                System.out.println("Employee SSN: " + rs.getString("SSN") + ", Total Expenses: " + rs.getString("total_expenses")); //
            }
            rs.close();
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }

    private static void executeQuery10(Connection connection, Scanner keyboard) {
        System.out.println("Enter client's SSN: "); // Getting client SSN
        String SSN = keyboard.next();

        String sql = "{call GetVolunteersByClient(?)}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            // Setting parameters
            stmt.setString(1, SSN);

            ResultSet rs = stmt.executeQuery(); // Execute the stored procedure

            while (rs.next()) { // Iterate through the result set
                System.out.println("Volunteer SSN: " + rs.getString("SSN") + ", Date Joined: " + rs.getString("date_joined")); // Prir
            }
            rs.close();
            stmt.close(); // Close the statement
            } catch (SQLException e) {
                System.err.println("SQL Error occurred:");
                System.err.println("SQL State: " + e.getSQLState());
                System.err.println("Error Code: " + e.getErrorCode());
                System.err.println("Message: " + e.getMessage());
        }
    }
```

```
    private static void executeQuery11(Connection connection, Scanner keyboard) {
        System.out.println("Enter date in yyyy-MM-dd format: "); // Getting date
        String date = keyboard.next();

        String sql = "{call GetTeamsByDate(?)}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            // Setting parameters
            stmt.setString(1, date);

            ResultSet rs = stmt.executeQuery(); // Execute the stored procedure

            while (rs.next()) { // Iterate through the result set
                System.out.println("Team Name: " + rs.getString("team_name")); // Print each row
            }
            rs.close();
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }

    private static void executeQuery12(Connection connection) {
        String sql = "{call GetPeople()}"; // Prepare the SQL call for the procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement

            ResultSet rs = stmt.executeQuery(); // Execute the stored procedure

            while (rs.next()) { // Iterate through the result set to print info
                System.out.println("Name: " + rs.getString("person_name") + ", SSN: " + rs.getString("SSN") + ", Email: " + rs.getStr:
            }
            rs.close();
            stmt.close(); // Close the statement
        } catch (SQLException e) {
            System.err.println("SQL Error occurred:");
            System.err.println("SQL State: " + e.getSQLState());
            System.err.println("Error Code: " + e.getErrorCode());
            System.err.println("Message: " + e.getMessage());
        }
    }
```

```
608    private static void executeQuery13(Connection connection) {
609        String sql = "{call GetDonorEmployees()}"; // Prepare the SQL call for the procedure
610        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
611
612            ResultSet rs = stmt.executeQuery(); // Execute the stored procedure
613
614            while (rs.next()) { // Iterate through the result set to print info
615                System.out.println("Name: " + rs.getString("person_name") + ", SSN: " + rs.getString("SSN") + ", Total Donations: " +
616            }
617            rs.close();
618            stmt.close(); // Close the statement
619        } catch (SQLException e) {
620            System.err.println("SQL Error occurred:");
621            System.err.println("SQL State: " + e.getSQLState());
622            System.err.println("Error Code: " + e.getErrorCode());
623            System.err.println("Message: " + e.getMessage());
624        }
625    }
626
627    private static void executeQuery14(Connection connection) {
628        String sql = "{call IncreaseSalary()}"; // Prepare the SQL call for the procedure
629        try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
630
631            stmt.execute(); // Execute the stored procedure
632
633            System.out.println("Salaries increased successfully.");
634            stmt.close(); // Close the statement
635        } catch (SQLException e) {
636            System.err.println("SQL Error occurred:");
637            System.err.println("SQL State: " + e.getSQLState());
638            System.err.println("Error Code: " + e.getErrorCode());
639            System.err.println("Message: " + e.getMessage());
640        }
641    }
642
643    private static void executeQuery15(Connection connection) {
644        String sql = "{call DeleteClients()}"; // Prepare the SQL call for the procedure
645        try (CallableStatement stmt = connection.prepareCall(sql))  { // Create CallableStatement
646
647            stmt.execute(); // Execute the stored procedure
648
649            System.out.println("Clients deleted successfully.");
650            stmt.close(); // Close the statement
651        } catch (SQLException e) {
```

```
652             System.err.println("SQL Error occurred:");
653             System.err.println("SQL State: " + e.getSQLState());
654             System.err.println("Error Code: " + e.getErrorCode());
655             System.err.println("Message: " + e.getMessage());
656         }
657     }
658
659     private static void executeImportTeams(Connection connection, Scanner keyboard) throws IOException {
660         System.out.print("Enter the input csv file name: ");
661         String fileName = keyboard.next();
662
663         BufferedReader reader = new BufferedReader(new FileReader(fileName));
664
665         String line;
666         while ((line = reader.readLine()) != null) {
667             String[] parts = line.split(",");  // Assuming the file is CSV
668             String teamName = parts[0].trim(); // Get team name
669             String teamType = parts[1].trim(); // Get team type
670             String dateFormed = parts[2].trim(); // Get date formed
671
672             String sql = "{call EnterTeam(?, ?, ?)}"; // Prepare the SQL call for the procedure
673             try (CallableStatement stmt = connection.prepareCall(sql)) { // Create CallableStatement
674
675                 // Setting parameters
676                 stmt.setString(1, teamName);
677                 stmt.setString(2, teamType);
678                 stmt.setString(3, dateFormed);
679
680                 stmt.execute(); // Execute the stored procedure
681                 stmt.close(); // Close the statement
682             } catch (SQLException e) {
683                 System.err.println("SQL Error occurred:");
684                 System.err.println("SQL State: " + e.getSQLState());
685                 System.err.println("Error Code: " + e.getErrorCode());
686                 System.err.println("Message: " + e.getMessage());
687             }
688         }
```

```
688         }
689         reader.close();
690         System.out.println("Teams inserted successfully from file.");
691     }
692
693     private static void executeExportMailing(Connection connection, Scanner keyboard) throws IOException {
694         System.out.print("Enter the output file name: ");
695         String fileName = keyboard.next();
696
697         BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
698         String sql = "{call GetMailingList()}";
699         try (CallableStatement stmt = connection.prepareCall(sql)) {
700
701             ResultSet rs = stmt.executeQuery(); // Execute getMailingList
702             while (rs.next()) {
703                 String personName = rs.getString("person_name"); // Get name
704                 String mailingAddress = rs.getString("mailing_address"); // Get address
705                 writer.write(personName + ", " + mailingAddress); // Concatenate and write to file
706                 writer.newLine();
707             }
708             rs.close();
709             stmt.close();
710             writer.close();
711             System.out.println("Mailing list exported successfully.");
712         } catch (SQLException e) {
713             System.err.println("SQL Error occurred:");
714             System.err.println("SQL State: " + e.getSQLState());
715             System.err.println("Error Code: " + e.getErrorCode());
716             System.err.println("Message: " + e.getMessage());
717         }
718     }
719
720 }
```

```
IP_Task5b [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Nov 14, 2024, 5:46:36 PM) [pid: 38664]
[INFO   ] Starting the application
[INFO   ] Loading application properties
[INFO   ] Connecting to the database
[INFO   ] Database connection test: cs-dsa-4513-sql-db
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM
```

Task 6. Java Program Execution

6.1. Screenshots showing testing of query 1

```
[INFO   ] Starting the application
[INFO   ] Loading application properties
[INFO   ] Connecting to the database
[INFO   ] Database connection test: cs-dsa-4513-sql-db
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM

Choose an option:

(1) Enter a new team into the database
(2) Enter a new client into the database and associate him or her with one or more teams
(3) Enter a new volunteer into the database and associate him or her with one or more teams
(4) Enter the number of hours a volunteer worked this month for a particular team
(5) Enter a new employee into the database and associate him or her with one or more teams
(6) Enter an expense charged by an employee
(7) Enter a new donor and associate him or her with several donations
(8) Retrieve the name and phone number of the doctor of a particular client
(9) Retrieve the total amount of expenses charged by each employee for a particular period of
time
(10) Retrieve the list of volunteers that are members of teams that support a particular client
(11) Retrieve the names of all teams that were founded after a particular date
(12) Retrieve the names, social security numbers, contact information, and emergency contact
information of all people in the database
(13) Retrieve the name and total amount donated by donors that are also employees.
(14) Increase the salary by 10% of all employees to whom more than one team must report
(15) Delete all clients who do not have health insurance and whose value of importance for
transportation is less than 5
(16) Import: enter new teams from a data file until the file is empty
(17) Export: Retrieve names and mailing addresses of all people on the mailing list and
output them to a data file
(18) Quit.

Choose an option (1-18):
```

Description of queries shown first

```
Choose an option (1-18):
1
Enter team name:
Team B
Enter team type:
Serious
Enter date formed in format yyyy-MM-dd:
2024-03-01
Team inserted successfully.
Choose an option (1-18):
1
Enter team name:
Team C
Enter team type:
Blah
Enter date formed in format yyyy-MM-dd:
2024-04-01
Team inserted successfully.
Choose an option (1-18):
1
Enter team name:
Team D
Enter team type:
School
Enter date formed in format yyyy-MM-dd:
2024-05-01
Team inserted successfully.
Choose an option (1-18):
1
Enter team name:
Team E
Enter team type:
Work
Enter date formed in format yyyy-MM-dd:
2024-06-01
Team inserted successfully.
Choose an option (1-18):
```

Some of the data fed into query 1

	team_name	team_type	date_formed
1	Team A	Fun	2022-02-01
2	Team B	Serious	2024-03-01
3	Team C	Blah	2024-04-01
4	Team D	School	2024-05-01
5	Team E	Work	2024-06-01

Team table after query 1

6.2. Screenshots showing testing of query 2

	person_name	SSN	gender	profession	mailing_addre…	email_address	phone_number	inMailingList
1	Alicia	111-11-1111	Female	Teacher	5085 Shorelin…	Email@email	9999999999	1
2	Nico	222-22-2222	Male	Finance	300 Symmes	email@email	9999999999	1
3	Lauren	333-33-3333	Female	Nurse	Monnett Ave	email@email	9999999999	0
4	Richard	444-44-4444	Male	Business	Apache St	richardemail@…	9999999999	1
5	Melanie	555-55-5555	Female	Queen	Boyd St	melemail@emai…	9999999999	0
6	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Person table after query 2

	SSN	doctor_name	doctor_phone_…	date_assigned
1	111-11-1111	Dr. 1	9999999999	2024-03-03
2	222-22-2222	Dr. 2	9999999999	2024-05-05
3	333-33-3333	Dr. 3	9999999999	2024-04-05
4	444-44-4444	Dr. 4	9999999999	2024-07-01
5	555-55-5555	Dr. 5	9999999999	2024-09-09

Client table after query 3

	SSN	need_name	importance_va…
1	333-33-3333	food	2
2	222-22-2222	transportatio…	3
3	111-11-1111	transporation	5
4	111-11-1111	food	6
5	444-44-4444	food	6
6	333-33-3333	transportatio…	7
7	555-55-5555	food	7
8	555-55-5555	transportatio…	8
9	222-22-2222	food	9

Need table after query 2

	policy_ID	provider_name	provider_addr…	insurance_typ…	SSN
1	1	policy1	add1	health	111-11-1111
2	2	policy2	add1	auto	111-11-1111
3	3	name3	add3	auto	222-22-2222
4	4	name4	add4	home	333-33-3333
5	5	name5	add5	home	444-44-4444
6	6	name6	add6	auto	444-44-4444
7	7	name7	add7	health	555-55-5555

Insurance table after query 2

	client_SSN	team_name	isActive
1	111-11-1111	Team A	1
2	222-22-2222	Team B	1
3	222-22-2222	Team C	1
4	333-33-3333	Team D	1
5	444-44-4444	Team E	1
6	555-55-5555	Team A	1
7	555-55-5555	Team C	1

Cares table after query 2

6.3. Screenshots showing testing of query 3

```
Is volunteer in mailing list? 1 for yes, 0 for no:
0
Enter volunteer's date joined in yyyy-MM-dd format:
2023-12-12
Enter volunteer's last training date in yyyy-MM-dd format:
2024-01-01
Enter volunteer's last training location:
Chicago
Enter volunteers's teams/roles in team1:role1,team2:role2,... format:
Team D:leads
Volunteer inserted successfully.
Choose an option (1-18):
3
Enter volunteer's name:
Karen
Enter volunteer's SSN:
999-99-9999
Enter volunteer's gender:
Female
Enter volunteer's profession:
Management
Enter volunteer's mailing address:
Pecan Ridge Way
Enter volunteer's email address:
karenemail
Enter volunteer's phone number:
9999999999
Is volunteer in mailing list? 1 for yes, 0 for no:
0
Enter volunteer's date joined in yyyy-MM-dd format:
2024-06-06
Enter volunteer's last training date in yyyy-MM-dd format:
2024-09-26
Enter volunteer's last training location:
Durham
Enter volunteers's teams/roles in team1:role1,team2:role2,... format:
Team A:serves,Team C:serves,Team E:leads
Volunteer inserted successfully.
```

Some of the data fed into query 3

	SSN	date_joined	last_training…	last_training…
1	111-11-1111	2023-03-05	2024-10-31	Norman
2	666-66-6666	2024-06-06	2024-11-01	Frisco
3	777-77-7777	2022-06-07	2024-03-20	Dallas
4	888-88-8888	2023-12-12	2024-01-01	Chicago
5	999-99-9999	2024-06-06	2024-09-26	Durham

Volunteer table after query 3

	person_name	SSN	gender	profession	mailing_addre…	email_address	phone_number	inMailingList
1	Alicia	111-11-1111	Female	Teacher	5085 Shorelin…	Email@email	9999999999	1
2	Nico	222-22-2222	Male	Finance	300 Symmes	email@email	9999999999	1
3	Lauren	333-33-3333	Female	Nurse	Monnett Ave	email@email	9999999999	0
4	Richard	444-44-4444	Male	Business	Apache St	richardemail@…	9999999999	1
5	Melanie	555-55-5555	Female	Queen	Boyd St	melemail@emai…	9999999999	0
6	Michael	666-66-6666	Male	Pastor	5085 Shorelin…	mikeyemail	9999999999	1
7	Ashley	777-77-7777	Female	Marketing	Greenville Av…	ashleyemail	9999999999	0
8	Jack	888-88-8888	Male	Soccer	Purdue St	jackemail	9999999999	0
9	Karen	999-99-9999	Female	Management	Pecan Ridge W…	karenemail	9999999999	0

Person table after query 3

	volunteer_SSN	team_name	serves_hours	isActive
1	111-11-1111	Team B	0	1
2	666-66-6666	Team C	0	1
3	777-77-7777	Team E	0	1
4	999-99-9999	Team A	0	1
5	999-99-9999	Team C	0	1

Serves table after query 3

	volunteer_SSN	team_name	leads_hours	isActive
1	666-66-6666	Team B	0	1
2	777-77-7777	Team A	0	1
3	888-88-8888	Team D	0	1
4	999-99-9999	Team E	0	1

Leads table after query 3

6.4. **Screenshots showing testing of query 4**

```
40
Volunteer hours inserted successfully.
Choose an option (1-18):
4
Enter volunteer's SSN:
888-88-8888
Enter team name:
Team D
Enter number of hours to add:
20
Volunteer hours inserted successfully.
Choose an option (1-18):
4
Enter volunteer's SSN:
666-66-6666
Enter team name:
Team B
Enter number of hours to add:
25
Volunteer hours inserted successfully.
Choose an option (1-18):
4
Enter volunteer's SSN:
111-11-1111
Enter team name:
Team A
Enter number of hours to add:
35
Volunteer hours inserted successfully.
Choose an option (1-18):
4
Enter volunteer's SSN:
999-99-9999
Enter team name:
Team E
Enter number of hours to add:
11
Volunteer hours inserted successfully.
Choose an option (1-18):
```

Some of the data fed into query 4

	volunteer_SSN	team_name	leads_hours	isActive
1	666-66-6666	Team B	25	1
2	777-77-7777	Team A	0	1
3	888-88-8888	Team D	20	1
4	999-99-9999	Team E	11	1

Leads table after query 4

	volunteer_SSN	team_name	serves_hours	isActive
1	111-11-1111	Team B	0	1
2	666-66-6666	Team C	0	1
3	777-77-7777	Team E	0	1
4	999-99-9999	Team A	40	1
5	999-99-9999	Team C	0	1

Serves table after query 4

6.5. Screenshots showing testing of query 5

```
Is employee in mailing list? 1 for yes, 0 for no:
0
Enter employee's salary:
180000
Enter employee's marital status:
Single
Enter employee's hire date in yyyy-MM-dd format:
2023-12-01
Enter employee's teams in team1,team2,... format:
Team E
Employee inserted successfully.
Choose an option (1-18):
5
Enter employee's name:
Jillian
Enter employee's SSN:
150-00-0000
Enter employee's gender:
Female
Enter employee's profession:
Accountant
Enter employee's mailing address:
Houston St
Enter employee's email address:
jillianemail
Enter employee's phone number:
9999999999
Is employee in mailing list? 1 for yes, 0 for no:
1
Enter employee's salary:
90000
Enter employee's marital status:
Single
Enter employee's hire date in yyyy-MM-dd format:
2023-04-05
Enter employee's teams in team1,team2,... format:

Employee inserted successfully.
```

Some of the data fed into query 5

	Boyd St –name	SSN	gender	profession	mailing_addre…	email_address	phone_number	inMaili
1	Chris	100-00-0000	Male	Cashier	Apache St	chrisemail	9999999999	1
2	Alicia	111-11-1111	Female	Teacher	5085 Shorelin…	Email@email	9999999999	1
3	Cole	120-00-0000	Male	Senior Exec	Coolwater Cov…	coleemail	9999999999	0
4	Sophia	130-00-0000	Female	Accountant	Monterey Rd	sophiaemail	9999999999	1
5	Macie	140-00-0000	Female	Pyschologist	Ridgecrest Rd	macieemail	9999999999	0
6	Jillian	150-00-0000	Female	Accountant	Houston St	jillianemail	9999999999	1
7	Nico	222-22-2222	Male	Finance	300 Symmes	email@email	9999999999	1
8	Lauren	333-33-3333	Female	Nurse	Monnett Ave	email@email	9999999999	0
9	Richard	444-44-4444	Male	Business	Apache St	richardemail@…	9999999999	1
1…	Melanie	555-55-5555	Female	Queen	Boyd St	melemail@emai…	9999999999	0
1…	Michael	666-66-6666	Male	Pastor	5085 Shorelin…	mikeyemail	9999999999	1
1…	Ashley	777-77-7777	Female	Marketing	Greenville Av…	ashleyemail	9999999999	0
1…	Jack	888-88-8888	Male	Soccer	Purdue St	jackemail	9999999999	0
1…	Karen	999-99-9999	Female	Management	Pecan Ridge W…	karenemail	9999999999	0

People table after query 5

	SSN	salary	marital_statu…	hire_date
1	100-00-0000	8000.00	Single	2024-02-14
2	120-00-0000	100000.00	Married	2024-08-09
3	130-00-0000	65000.00	Married	2024-11-11
4	140-00-0000	180000.00	Single	2023-12-01
5	150-00-0000	90000.00	Single	2023-04-05

Employee table after query 5

	employee_SSN	team_name	report_date	report_descri…
1	120-00-0000	Team B	*NULL*	*NULL*
2	120-00-0000	Team C	*NULL*	*NULL*
3	120-00-0000	Team E	*NULL*	*NULL*
4	130-00-0000	Team A	*NULL*	*NULL*
5	140-00-0000	Team E	*NULL*	*NULL*

Reports table after query 5

6.6. Screenshots showing testing of query 6

```
Enter expense description:
Supplies
Enter expense date in yyyy-MM-dd format:
2024-07-06
Employee expense inserted successfully.
Choose an option (1-18):
6
Enter employee's SSN:
130-00-0000
Enter expense amount:
400
Enter expense description:
Need
Enter expense date in yyyy-MM-dd format:
2023-09-09
Employee expense inserted successfully.
Choose an option (1-18):
6
Enter employee's SSN:
120-00-0000
Enter expense amount:
250
Enter expense description:
Business Needs
Enter expense date in yyyy-MM-dd format:
2024-11-02
Employee expense inserted successfully.
Choose an option (1-18):
6
Enter employee's SSN:
140-00-0000
Enter expense amount:
50
Enter expense description:
Event
Enter expense date in yyyy-MM-dd format:
2023-12-13
Employee expense inserted successfully.
Choose an option (1-18):
```

Some of the data fed into query 6

	SSN	amount	expense_descr…	expense_date
1	120-00-0000	250.00	Business Need…	2024-11-02
2	130-00-0000	400.00	Need	2023-09-09
3	140-00-0000	50.00	Event	2023-12-13
4	150-00-0000	100.00	Lunch	2024-05-05
5	150-00-0000	150.00	Supplies	2024-07-06

Expense table after query 6

6.7. Screenshots showing testing of query 7

```
Enter donor's profession:
Librarian
Enter donor's mailing address:
Colorado St
Enter donor's email address:
eleanoremail
Enter donor's phone number:
9999999999
Is donor in mailing list? 1 for yes, 0 for no:
0
Is donor anonymous? 1 for yes, 0 for n
0
Enter donations in format date,amount,type,campaign,check#,card#,cardtype,cardexpdate|...:
2024-09-15,5000,Card,Campaign4,,4444000044440000,Visa,2027-06-12
Donor and donations inserted successfully.
Choose an option (1-18):
7
Enter donor's name:
Sophia
Enter donor's SSN:
130-00-0000
Enter donor's gender:
Female
Enter donor's profession:
Accountant
Enter donor's mailing address:
mail
Enter donor's email address:
email
Enter donor's phone number:
phone
Is donor in mailing list? 1 for yes, 0 for no:
1
Is donor anonymous? 1 for yes, 0 for n
0
Enter donations in format date,amount,type,campaign,check#,card#,cardtype,cardexpdate|...:
2024-02-24,500,Check,Campaign4,CHK3,,,|2024-05-06,400,Card,Campaign4,,2222333322223333,Visa,2024-09-09
Donor and donations inserted successfully.
Choose an option (1-18):
```

Some of the data fed into query 7

	person_name	SSN	gender	profession	mailing_addre…	email_address	phone_number	inMailingList
1	Chris	100-00-0000	Male	Cashier	Apache St	chrisemail	9999999999	1
2	Alicia	111-11-1111	Female	Teacher	5085 Shorelin…	Email@email	9999999999	1
3	Cole	120-00-0000	Male	Senior Exec	Coolwater Cov…	coleemail	9999999999	0
4	Sophia	130-00-0000	Female	Accountant	Monterey Rd	sophiaemail	9999999999	1
5	Macie	140-00-0000	Female	Pyschologist	Ridgecrest Rd	macieemail	9999999999	0
6	Jillian	150-00-0000	Female	Accountant	Houston St	jillianemail	9999999999	1
7	Scott	200-00-0000	Male	Professor	Symmes St	scottemail	9999999999	1
8	Robert	210-00-0000	Male	Construction	James Garner …	robertemail	9999999999	0
9	Lois	220-00-0000	Female	Government	Ventura St	loisemail	9999999999	1
1…	Nico	222-22-2222	Male	Finance	300 Symmes	email@email	9999999999	1
1…	Eleanor	230-00-0000	Female	Librarian	Colorado St	eleanoremail	9999999999	0
1…	Lauren	333-33-3333	Female	Nurse	Monnett Ave	email@email	9999999999	0
1…	Richard	444-44-4444	Male	Business	Apache St	richardemail@…	9999999999	1
1…	Melanie	555-55-5555	Female	Queen	Boyd St	melemail@emai…	9999999999	0
1…	Michael	666-66-6666	Male	Pastor	5085 Shorelin…	mikeyemail	9999999999	1
1…	Ashley	777-77-7777	Female	Marketing	Greenville Av…	ashleyemail	9999999999	0
1…	Jack	888-88-8888	Male	Soccer	Purdue St	jackemail	9999999999	0
1…	Karen	999-99-9999	Female	Management	Pecan Ridge W…	karenemail	9999999999	0

Person table after query 7

	SSN	isAnon
1	130-00-0000	0
2	200-00-0000	0
3	210-00-0000	0
4	220-00-0000	1
5	230-00-0000	0

Donor table after query 7

	SSN	donation_date	amount	donation_type	campaign	card_number	card_type	card_expirati…
1	130-00-0000	2024-05-06	400.00	Card	Campaign4	2222333322223…	Visa	2024-09-09
2	200-00-0000	2024-07-23	200.00	Card	Campaign2	1111111111111…	AMEX	2027-11-11
3	220-00-0000	2023-10-10	200.00	Card	Campaign3	3333333333333…	Mastercard	2028-06-20
4	230-00-0000	2024-09-15	5000.00	Card	Campaign4	4444000044440…	Visa	2027-06-12
5	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Card Donation table after query 7

	SSN	donation_date	amount	donation_type	campaign	check_number
1	130-00-0000	2024-02-24	500.00	Check	Campaign4	CHK3
2	200-00-0000	2024-05-05	1000.00	Check	Campaign1	CHK1
3	210-00-0000	2024-01-12	10000.00	Check	Campaign2	CHK2

Check Donation table after query 7

6.8. Screenshots showing testing of query 8

```
Choose an option (1-18):
8
Enter client's SSN:
111-11-1111
Doctor Name: Dr. 1, Doctor Phone Number: 9999999999
Choose an option (1-18):
8
Enter client's SSN:
444-44-4444
Doctor Name: Dr. 4, Doctor Phone Number: 9999999999
```

Two runs of query 8

	SSN	doctor_name	doctor_phone_…	date_assigned
1	111-11-1111	Dr. 1	9999999999	2024-03-03
2	222-22-2222	Dr. 2	9999999999	2024-05-05
3	333-33-3333	Dr. 3	9999999999	2024-04-05
4	444-44-4444	Dr. 4	9999999999	2024-07-01
5	555-55-5555	Dr. 5	9999999999	2024-09-09
6	*NULL*	*NULL*	*NULL*	*NULL*

Data used for query 8

6.9. Screenshots showing testing of query 9

```
Choose an option (1-18):
9
Enter start date in yyyy-MM-dd format:
2024-01-01
Enter end date in yyyy-MM-dd format:
2024-11-20
Employee SSN: 120-00-0000, Total Expenses: 250.00
Employee SSN: 150-00-0000, Total Expenses: 250.00
Choose an option (1-18):
9
Enter start date in yyyy-MM-dd format:
2023-09-08
Enter end date in yyyy-MM-dd format:
2024-05-10
Employee SSN: 130-00-0000, Total Expenses: 400.00
Employee SSN: 150-00-0000, Total Expenses: 100.00
Employee SSN: 140-00-0000, Total Expenses: 50.00
```

Two runs of query 9

	SSN	amount	expense_descr…	expense_date
1	120-00-0000	250.00	Business Need…	2024-11-02
2	130-00-0000	400.00	Need	2023-09-09
3	140-00-0000	50.00	Event	2023-12-13
4	150-00-0000	100.00	Lunch	2024-05-05
5	150-00-0000	150.00	Supplies	2024-07-06

Data used for query 9

6.10. Screenshots showing testing of query 10

```
Choose an option (1-18):
10
Enter client's SSN:
555-55-5555
Volunteer SSN: 666-66-6666, Date Joined: 2024-06-06
Volunteer SSN: 777-77-7777, Date Joined: 2022-06-07
Volunteer SSN: 999-99-9999, Date Joined: 2024-06-06
Choose an option (1-18):
10
Enter client's SSN:
333-33-3333
Volunteer SSN: 888-88-8888, Date Joined: 2023-12-12
```

Two runs of query 10

	client_SSN	team_name	isActive
1	111-11-1111	Team A	1
2	222-22-2222	Team B	1
3	222-22-2222	Team C	1
4	333-33-3333	Team D	1
5	444-44-4444	Team E	1
6	555-55-5555	Team A	1
7	555-55-5555	Team C	1

	volunteer_SSN	team_name	serves_hours	isActive
1	111-11-1111	Team B	0	1
2	666-66-6666	Team C	0	1
3	777-77-7777	Team E	0	1
4	999-99-9999	Team A	0	1
5	999-99-9999	Team C	0	1

	volunteer_SSN	team_name	leads_hours	isActive
1	666-66-6666	Team B	0	1
2	777-77-7777	Team A	0	1
3	888-88-8888	Team D	0	1
4	999-99-9999	Team E	0	1

Data used for query 10

6.11. Screenshots showing testing of query 11

```
Choose an option (1-18):
11
Enter date in yyyy-MM-dd format:
2023-01-01
Team Name: Team B
Team Name: Team C
Team Name: Team D
Team Name: Team E
Choose an option (1-18):
11
Enter date in yyyy-MM-dd format:
2024-04-05
Team Name: Team D
Team Name: Team E
```

Two runs of query 11

	team_name	team_type	date_formed
1	Team A	Fun	2022-02-01
2	Team B	Serious	2024-03-01
3	Team C	Blah	2024-04-01
4	Team D	School	2024-05-01
5	Team E	Work	2024-06-01

Data used for query 11

6.12. Screenshots showing testing of query 12

```
Choose an option (1-18):
12
Name: Chris, SSN: 100-00-0000, Email: chrisemail, Phone: 9999999999
Name: Alicia, SSN: 111-11-1111, Email: Email@email, Phone: 9999999999
Name: Cole, SSN: 120-00-0000, Email: coleemail, Phone: 9999999999
Name: Sophia, SSN: 130-00-0000, Email: sophiaemail, Phone: 9999999999
Name: Macie, SSN: 140-00-0000, Email: macieemail, Phone: 9999999999
Name: Jillian, SSN: 150-00-0000, Email: jillianemail, Phone: 9999999999
Name: Scott, SSN: 200-00-0000, Email: scottemail, Phone: 9999999999
Name: Robert, SSN: 210-00-0000, Email: robertemail, Phone: 9999999999
Name: Lois, SSN: 220-00-0000, Email: loisemail, Phone: 9999999999
Name: Nico, SSN: 222-22-2222, Email: email@email, Phone: 9999999999
Name: Eleanor, SSN: 230-00-0000, Email: eleanoremail, Phone: 9999999999
Name: Lauren, SSN: 333-33-3333, Email: email@email, Phone: 9999999999
Name: Richard, SSN: 444-44-4444, Email: richardemail@email, Phone: 9999999999
Name: Melanie, SSN: 555-55-5555, Email: melemail@email, Phone: 9999999999
Name: Michael, SSN: 666-66-6666, Email: mikeyemail, Phone: 9999999999
Name: Ashley, SSN: 777-77-7777, Email: ashleyemail, Phone: 9999999999
Name: Jack, SSN: 888-88-8888, Email: jackemail, Phone: 9999999999
Name: Karen, SSN: 999-99-9999, Email: karenemail, Phone: 9999999999
```

Run of query 12

	person_name	SSN	gender	profession	mailing_addre…	email_address	phone_number	inMailingList
1	Chris	100-00-0000	Male	Cashier	Apache St	chrisemail	9999999999	1
2	Alicia	111-11-1111	Female	Teacher	5085 Shorelin…	Email@email	9999999999	1
3	Cole	120-00-0000	Male	Senior Exec	Coolwater Cov…	coleemail	9999999999	0
4	Sophia	130-00-0000	Female	Accountant	Monterey Rd	sophiaemail	9999999999	1
5	Macie	140-00-0000	Female	Pyschologist	Ridgecrest Rd	macieemail	9999999999	0
6	Jillian	150-00-0000	Female	Accountant	Houston St	jillianemail	9999999999	1
7	Scott	200-00-0000	Male	Professor	Symmes St	scottemail	9999999999	1
8	Robert	210-00-0000	Male	Construction	James Garner …	robertemail	9999999999	0
9	Lois	220-00-0000	Female	Government	Ventura St	loisemail	9999999999	1
1…	Nico	222-22-2222	Male	Finance	300 Symmes	email@email	9999999999	1
1…	Eleanor	230-00-0000	Female	Librarian	Colorado St	eleanoremail	9999999999	0
1…	Lauren	333-33-3333	Female	Nurse	Monnett Ave	email@email	9999999999	0
1…	Richard	444-44-4444	Male	Business	Apache St	richardemail@…	9999999999	1
1…	Melanie	555-55-5555	Female	Queen	Boyd St	melemail@emai…	9999999999	0
1…	Michael	666-66-6666	Male	Pastor	5085 Shorelin…	mikeyemail	9999999999	1
1…	Ashley	777-77-7777	Female	Marketing	Greenville Av…	ashleyemail	9999999999	0
1…	Jack	888-88-8888	Male	Soccer	Purdue St	jackemail	9999999999	0
1…	Karen	999-99-9999	Female	Management	Pecan Ridge W…	karenemail	9999999999	0

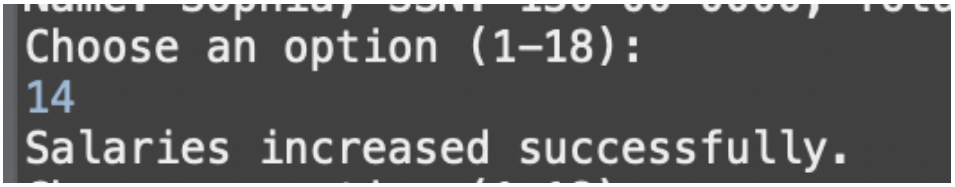
Data used for query 12

6.13. Screenshots showing testing of query 13

Run of query 13

When looking at the employee and donor table, there is only one overlap with SSN 130-00-0000. When you look at the donations table after query 7, you see that 130-00-0000 is associated with $900 worth of donations. Thus, the query ran successfully.

6.14. Screenshots showing testing of query 14

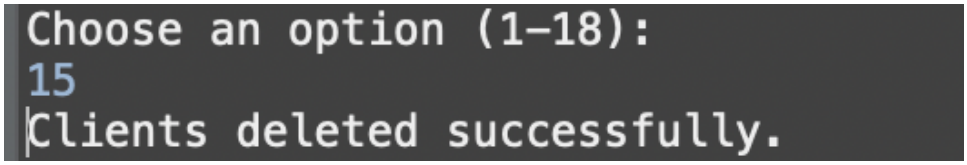


Run of query 14

	SSN	salary	marital_statu…	hire_date
1	100-00-0000	8000.00	Single	2024-02-14
2	120-00-0000	110000.00	Married	2024-08-09
3	130-00-0000	65000.00	Married	2024-11-11
4	140-00-0000	180000.00	Single	2023-12-01
5	150-00-0000	90000.00	Single	2023-04-05

Employee table after query 14 (employee 2's salary increased as they had more than one team reporting to them)

6.15. Screenshots showing testing of query 15



Run of query 15

	SSN	doctor_name	doctor_phone_…	date_assigned
1	111-11-1111	Dr. 1	9999999999	2024-03-03
2	333-33-3333	Dr. 3	9999999999	2024-04-05
3	444-44-4444	Dr. 4	9999999999	2024-07-01
4	555-55-5555	Dr. 5	9999999999	2024-09-09

Client table after query 15 (client 222-22-2222 was deleted because they did not have health insurance and the importance value for transportation was below 5)

6.16. . Screenshots showing the testing of the import and export options

```
Choose an option (1-18):
16
Enter the input csv file name: teams.csv
Teams inserted successfully from file.
```

Run of query 16

Team 1	Medical	4/2/22
Team 2	Support	4/3/23
Team 3	Operation	4/4/23

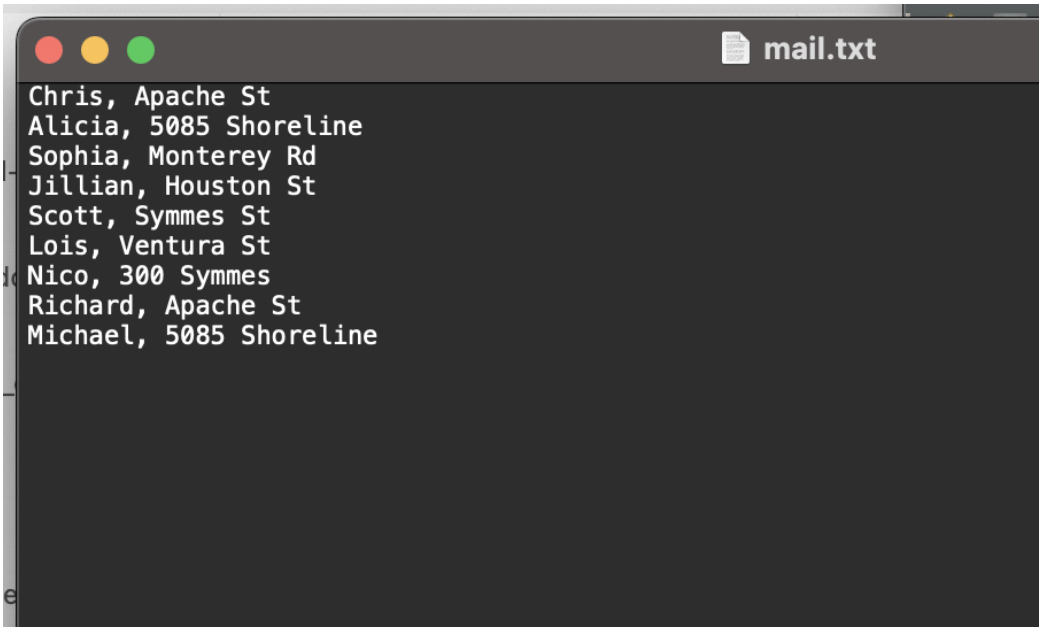
Sample teams.csv file used to import teams

	team_name	team_type	date_formed
1	Team 1	Medical	2022-04-02
2	Team 2	Support	2023-04-03
3	Team 3	Operation	2023-04-04
4	Team A	Fun	2022-02-01
5	Team B	Serious	2024-03-01
6	Team C	Blah	2024-04-01
7	Team D	School	2024-05-01
8	Team E	Work	2024-06-01

Teams table after query 16

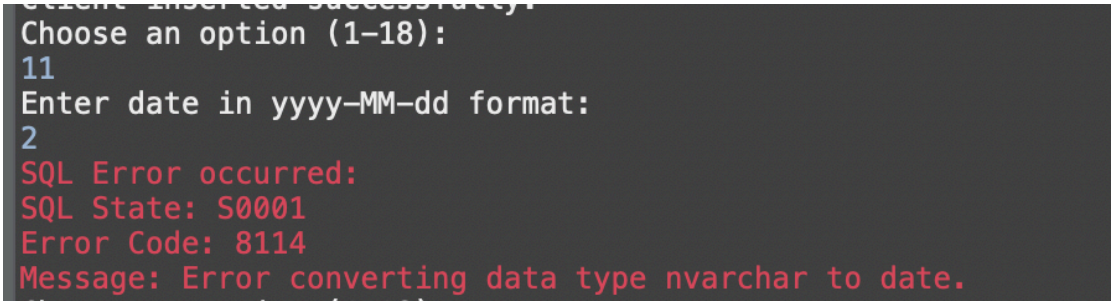
```
Choose an option (1-18):
17
Enter the output file name: mail.txt
Mailing list exported successfully.
```

Run of query 17



Output file mail.txt after query 17

6.17. Screenshots showing testing of three types of errors

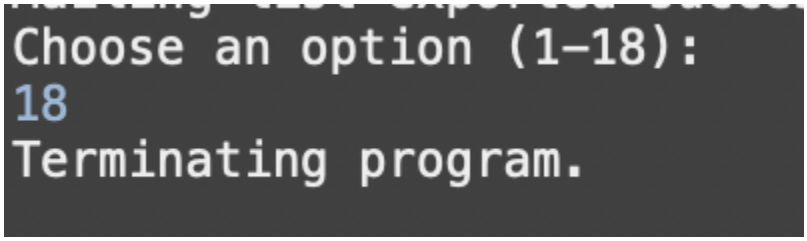


Error when inputting wrong date format

Primary key constraint error on team table

Primary key constraint error on expense table

6.18. Screenshots showing the testing of the quit option.



Run of quit option.