COURSE NAME: DATABASE MANAGEMENT SYSTEMS

COURSE NUMBER: CS 4513-001

SEMESTER AND YEAR: FALL SEMESTER 2024

INSTRUCTOR NAME: Dr. Le Gruenwald

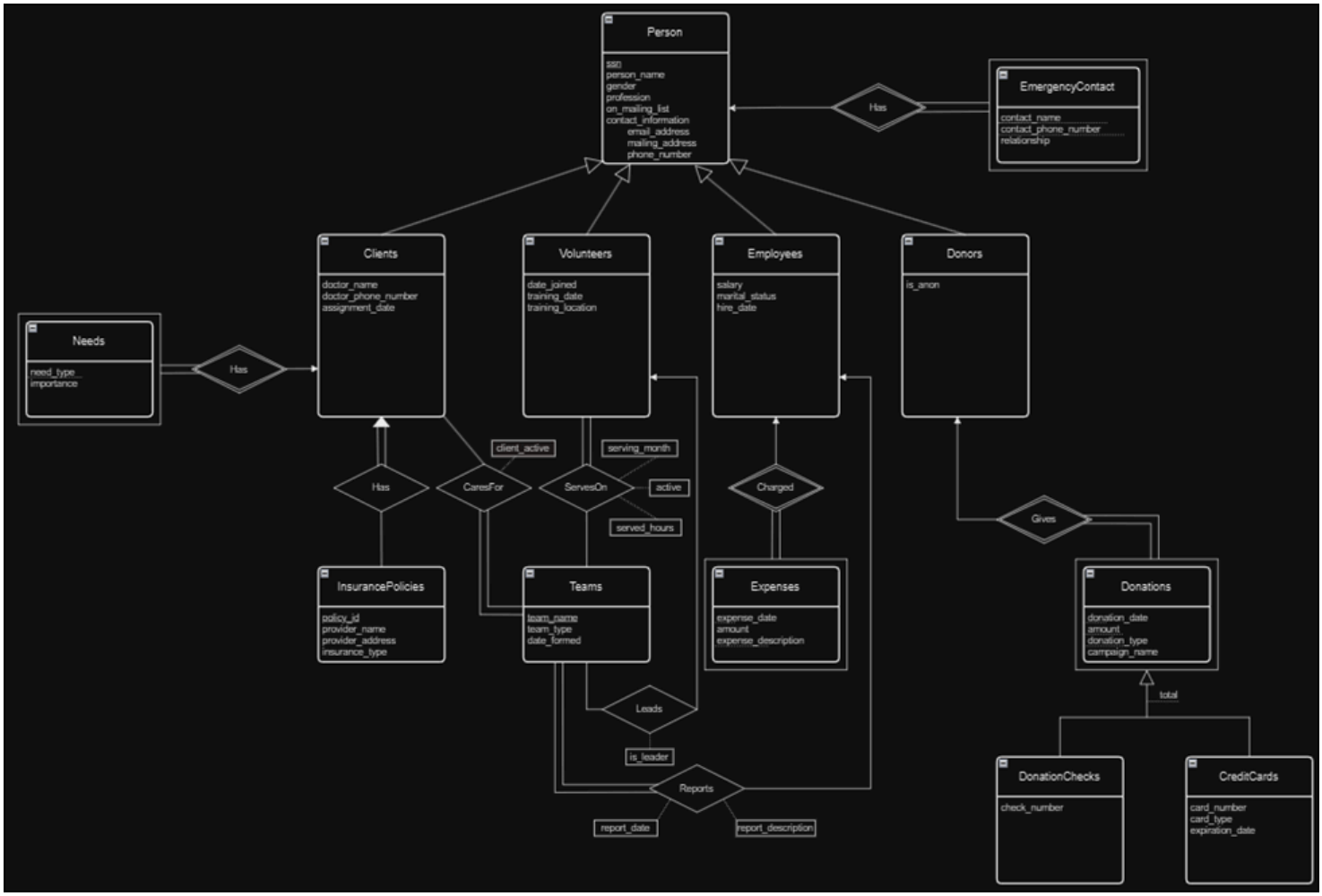
STUDENT NAME: Tanner Benbrook

OU ID: 113561037

EMAIL: Tanner.M.Benbrook-1@ou.edu

A PATIENT ASSISTANT NETWORK DATABASE SYSTEM

Task 1:


Person
ssn
person_name
gender
profession
on_mailing_list
contact_information
email_address
mailing_address
phone_number
Has
EmergencyContact
contact_name
contact_phone_number
relationship
Clients
doctor_name
doctor_phone_number
assignment_date
Volunteers
date_joined
training_date
training_location
Employees
salary
marital_status
hire_date
Donors
is_anon
Needs
need_type
importance
Has
client_active
serving_month
Has
CaresFor
ServesOn
active
served_hours
Charged
Gives
InsurancePolicies
policy_id
provider_name
provider_address
insurance_type
Teams
team_name
team_type
date_formed
Expenses
expense_date
amount
expense_description
Donations
donation_date
amount
donation_type
campaign_name
Leads
is_leader
total
Reports
report_date
report_description
DonationChecks
check_number
CreditCards
card_number
card_type
expiration_date

Task 2:

Clients *(ssn, person_name, gender, profession, on_mailing_list, mailing_address, phone_number, on_mailing_list, assignment_date, doctor_name, doctor_phone_number)*

Volunteers *(ssn, person_name, gender, profession, on_mailing_list, mailing_address, phone_number, on_mailing_list, date_joined, training_date, training_location)*

Employees *(ssn, person_name, gender, profession, on_mailing_list, mailing_address, phone_number, on_mailing_list, salary, marital_status, hire_date)*

Donors *(ssn, person_name, gender, profession, on_mailing_list, mailing_address, phone_number, on_mailing_list, is_anon)*

ClientEmergencyContacts *(client_ssn, contact_name, contact_phone_number, relationship)*

Has *(client_ssn, contact_name, contact_phone_number)*

DonorEmergencyContacts *(donor_ssn, contact_name, contact_phone_number, relationship)*

Has *(donor_ssn, contact_name, contact_phone_number)*

VolunteerEmergencyContacts *(volunteer_ssn, contact_name, contact_phone_number, relationship)*

Has *(volunteer_ssn, contact_name, contact_phone_number)*

EmployeeEmergencyContacts *(employee_ssn, contact_name, contact_phone_number, relationship)*

Has *(employee_ssn, contact_name, contact_phone_number)*

InsurancePolicies*(policy_id, provider_name, provider_address, insurance_type)*

Has*(ssn, policy_id)*

Needs*(ssn, need_type, importance)*

Teams*(team_name, team_type, date_formed)*

Expenses*(ssn, expense_date, amount, expense_description)*

DonationChecks*(ssn, check_number, donation_date, amount, donation_type, campaign_name)*

CreditCards*(ssn, card_number, card_type, expiration_date, donation_date, amount, donation_type, campaign_name)*

CaresFor*(ssn, team_name, client_active)*

Reports*(ssn, team_name, report_date, report_description)*

ServesOn*(ssn, team_name, serving_month, served_hours, active)*

Has*(ssn, need_type)*

Leads*(ssn ,team_name, is_leader)*

Charged*(ssn, expense_date, expense_description)*

Gives*(ssn, donation_date, amount, donation_type)*

Task 3:

Task 3.1:

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justification
Clients	2. Insertion 8. Random Search 10. Random Search 12. Random Search 15 Deletion	 SSN SSN SSN SSN	1/week 1/week 4/year 1/week 4/year	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Volunteers	3. Insertion 10. Random Search 12. Random Search	 SSN SSN	2/month 4/year 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Employees	5. Insertion 6. Insertion 12. Random Search 13. Random Search 14. Insertion	 SSN SSN SSN SSN	1/year 1/day 1/week 1/year 1/year	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Donors	7. Insertion 12. Random Search 13. Random Search	 SSN SSN	1/day 1/week 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
ClientEmergency Contacts	2. Insertion 12. Random Search	SSN SSN	1/week 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Volunteer Emergency Contacts	3. Insertion 12. Random Search	SSN SSN	2/month 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Employee Emergency Contacts	5. Insertion 12. Random Search	SSN SSN	1/year 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.

DonorEmergency Contacts	7. Insertion 12. Random Search	SSN SSN	1/day 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
InsurancePolicies	2. Insertion 15. Deletion	insurance_type	1/week 4/year	Extendable Hashing based off primary key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Needs	2. Insertion 15. Deletion	need_type	1/week 4/year	Extendable Hashing based off primary key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Teams	1. Insertion 3. Insertion 4. Insertion 5. Insertion 10. Random Search 11. Range Search 14. Insertion	team_name SSN team_name team_name date_formed team_name	1/month 2/month 30/month 1/year 4/year 1/month 1/year	B+-Tree	supports efficient insertions and is optimized for range queries due to its linked leaf node structure.
Expenses	6. Insertion 9. Range Search	SSN SSN	1/day 1/month	B+-Tree	supports efficient insertions and is optimized for range queries due to its linked leaf node structure.
DonationChecks	7. Insertion 13. Random Search	SSN SSN	1/day 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
CreditCards	7. Insertion 13. Random Search	SSN SSN	1/day 1/week	Extendable Hashing with SSN as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
CaresFor	2. Insertion 10. Random Search	team_name team_name and SSN	1/week 4/year	Extendable Hashing with SSN and team_name as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.

ServesOn	3. Insertion 4. Insertion 10. Random Search	team_name SSN and team_name SSN and team_name	2/month 30/month 4/year	Extendable Hashing with SSN and team_name as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Reports	1. Insertion 5. Insertion 14. Random Search	team_name SSN and team_name SSN	1/month 1/year 1/year	Extendable Hashing with SSN and team_name as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Has	2. Insertion 15. Deletion	SSN and policy_id SSN and policy_id	1/week 4/year	Extendable Hashing with SSN and policy_id as the hash key.	Unpredictable size for tables and very fast insertions, deletions, and random searches.
Leads	3. Insertion	SSN and team_name	2/month	Heap file	Fast for insertion.

Task 3.2:

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization/ Index Requirement	Justification
Clients	2. Insertion 8. Random Search 10. Random Search 12. Random Search 15 Deletion	 SSN SSN SSN SSN	1/week 1/week 4/year 1/week 4/year	Indexing Required.	Indexing based off of Clients primary key because of frequent query frequencies.
Volunteers	3. Insertion 10. Random Search 12. Random Search	 SSN SSN	2/month 4/year 1/week	Indexing Required.	Indexing based off of Volunteers primary key because of frequent query frequencies.
Employees	5. Insertion 6. Insertion 12. Random Search 13. Random Search 14. Insertion	 SSN SSN SSN SSN	1/year 1/day 1/week 1/year 1/year	Indexing Required.	Indexing based off of Employees primary key because of frequent query frequencies.
Donors	7. Insertion 12. Random Search 13. Random Search	 SSN SSN	1/day 1/week 1/week	Indexing Required.	Indexing based off of Donors primary key because of frequent query frequencies.
ClientEmergency Contacts	2. Insertion 12. Random Search	SSN SSN	1/week 1/week	Indexing Required.	Indexing based off of SSN since query #12 is very frequent and is a Search.
Volunteer Emergency Contacts	3. Insertion 12. Random Search	SSN SSN	2/month 1/week	Indexing Required.	Indexing based off of SSN since query #12 is very frequent and is a Search.
Employee Emergency Contacts	5. Insertion 12. Random Search	SSN SSN	1/year 1/week	Indexing Required.	Indexing based off of SSN since query #12 is very frequent and is a Search.

DonorEmergency Contacts	7. Insertion 12. Random Search	SSN SSN	1/day 1/week	Indexing Required.	Indexing based off of SSN since query #12 is very frequent and is a Search.
InsurancePolicies	2. Insertion 15. Deletion	insurance_type	1/week 4/year	No Indexing Required.	No indexing just insertion and deletion queries, no need to create an index since there is no searching required.
Needs	2. Insertion 15. Deletion	need_type	1/week 4/year	No Indexing Required.	No indexing just insertion and deletion queries, no need to create an index since there is no searching required.
Teams	1. Insertion 3. Insertion 4. Insertion 5. Insertion 10. Random Search 11. Range Search 14. Insertion	team_name SSN team_name team_name date_formed team_name	1/month 2/month 30/month 1/year 4/year 1/month 1/year	Indexing Required.	Primary indexing on Teams primary key because query #4 is very frequent.
Expenses	6. Insertion 9. Range Search	SSN SSN	1/day 1/month	Indexing Required.	#Query 9 specifies Expenses to be sorted based on amount.
DonationChecks	7. Insertion 13. Random Search	SSN SSN	1/day 1/week	Indexing Required.	Query #13 needs to be sorted based off of amount in DonationChecks.
CreditCards	7. Insertion 13. Random Search	SSN SSN	1/day 1/week	Indexing Required.	Query #13 needs to be sorted based off of amount in CreditCards.
CaresFor	2. Insertion 10. Random Search	team_name team_name and SSN	1/week 4/year	No Indexing Required.	Although insertion queries are frequent, the Random Search frequency is not frequent at all. No indexing.

ServesOn	3. Insertion 4. Insertion 10. Random Search	team_name SSN and team_name SSN and team_name	2/month 30/month 4/year	No Indexing Required.	Although insertion queries are frequent, the Random Search frequency is not frequent at all. No indexing.
Reports	1. Insertion 5. Insertion 14. Random Search	team_name SSN and team_name SSN	1/month 1/year 1/year	No Indexing Required.	Since the Random search is only once a year, this table doesn't need any indexing.
Has	2. Insertion 15. Deletion	SSN and policy_id SSN and policy_id	1/week 4/year	No Indexing Required.	Only insertion and deletion queries. No searching at all. No Indexing.
Leads	3. Insertion	SSN and team_name	2/month	No Indexing Required.	Only insertion query so no indexing required.

Task 4:

```
DROP TABLE IF EXISTS DonorEmergencyContacts;
DROP TABLE IF EXISTS VolunteerEmergencyContacts;
DROP TABLE IF EXISTS EmployeeEmergencyContacts;
DROP TABLE IF EXISTS ClientEmergencyContacts;
DROP TABLE IF EXISTS Has;
DROP TABLE IF EXISTS Gives;
DROP TABLE IF EXISTS Leads;
DROP TABLE IF EXISTS Reports;
DROP TABLE IF EXISTS CaresFor;
DROP TABLE IF EXISTS ServesOn;
DROP TABLE IF EXISTS CreditCards;
DROP TABLE IF EXISTS DonationChecks;
DROP TABLE IF EXISTS Donors;
DROP TABLE IF EXISTS Expenses;
DROP TABLE IF EXISTS Employees;
DROP TABLE IF EXISTS Teams;
DROP TABLE IF EXISTS InsurancePolicies;
DROP TABLE IF EXISTS Volunteers;
DROP TABLE IF EXISTS Needs;
DROP TABLE IF EXISTS Clients;

-- Clients Table

CREATE TABLE Clients (
    ssn INT PRIMARY KEY NOT NULL,
    person_name VARCHAR(256) NOT NULL,
    gender VARCHAR(256) NOT NULL,
    profession VARCHAR(256) NOT NULL,
    on_mailing_list BIT NOT NULL,
    mailing_address VARCHAR(256) NOT NULL,
    phone_number VARCHAR(256) NOT NULL,
    email_address VARCHAR(256) NOT NULL,
    assignment_date DATE NOT NULL,
    doctor_name VARCHAR(256) NOT NULL,
    doctor_phone_number VARCHAR(256) NOT NULL
);

-- Volunteers Table

CREATE TABLE Volunteers (
    ssn INT PRIMARY KEY NOT NULL,
    person_name VARCHAR(256) NOT NULL,
    gender VARCHAR(256) NOT NULL,
    profession VARCHAR(256) NOT NULL,
    on_mailing_list BIT NOT NULL,
```

```
    mailing_address VARCHAR(256) NOT NULL,
    phone_number VARCHAR(256) NOT NULL,
    email_address VARCHAR(256) NOT NULL,
    date_joined DATE NOT NULL,
    training_date DATE NOT NULL,
    training_location VARCHAR(256) NOT NULL
);

-- Employees Table

CREATE TABLE Employees (
    ssn INT PRIMARY KEY NOT NULL,
    person_name VARCHAR(256) NOT NULL,
    gender VARCHAR(256) NOT NULL,
    profession VARCHAR(256) NOT NULL,
    on_mailing_list BIT NOT NULL,
    mailing_address VARCHAR(256) NOT NULL,
    phone_number VARCHAR(256) NOT NULL,
    email_address VARCHAR(256) NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    marital_status VARCHAR(256) NOT NULL,
    hire_date DATE NOT NULL
);

-- Donors Table

CREATE TABLE Donors (
    ssn INT PRIMARY KEY NOT NULL,
    person_name VARCHAR(256) NOT NULL,
    gender VARCHAR(256) NOT NULL,
    profession VARCHAR(256) NOT NULL,
    on_mailing_list BIT NOT NULL,
    mailing_address VARCHAR(256) NOT NULL,
    phone_number VARCHAR(256) NOT NULL,
    email_address VARCHAR(256) NOT NULL,
    is_anon BIT NOT NULL
);

-- ClientEmergencyContacts Table
CREATE TABLE DonorEmergencyContacts (
    donor_ssn INT NOT NULL,
    contact_name VARCHAR(256) NOT NULL,
    contact_phone_number VARCHAR(256) NOT NULL,
    relationship VARCHAR(256) NOT NULL,
    PRIMARY KEY (donor_ssn, contact_name, contact_phone_number),
    FOREIGN KEY (donor_ssn) REFERENCES Donors(ssn)
);
```

```sql
-- VolunteerEmergencyContacts Table
CREATE TABLE VolunteerEmergencyContacts (
    volunteer_ssn INT NOT NULL,
    contact_name VARCHAR(256) NOT NULL,
    contact_phone_number VARCHAR(256) NOT NULL,
    relationship VARCHAR(256) NOT NULL,
    PRIMARY KEY (volunteer_ssn, contact_name, contact_phone_number),
    FOREIGN KEY (volunteer_ssn) REFERENCES Volunteers(ssn)
);

-- EmployeeEmergencyContacts Table
CREATE TABLE EmployeeEmergencyContacts (
    employee_ssn INT NOT NULL,
    contact_name VARCHAR(256) NOT NULL,
    contact_phone_number VARCHAR(256) NOT NULL,
    relationship VARCHAR(256) NOT NULL,
    PRIMARY KEY (employee_ssn, contact_name, contact_phone_number),
    FOREIGN KEY (employee_ssn) REFERENCES Employees(ssn)
);

-- ClientEmergencyContacts Table
CREATE TABLE ClientEmergencyContacts (
    client_ssn INT NOT NULL,
    contact_name VARCHAR(256) NOT NULL,
    contact_phone_number VARCHAR(256) NOT NULL,
    relationship VARCHAR(256) NOT NULL,
    PRIMARY KEY (client_ssn, contact_name, contact_phone_number),
    FOREIGN KEY (client_ssn) REFERENCES Clients(ssn)
);

-- Insurance Policies Table

CREATE TABLE InsurancePolicies (
    policy_id VARCHAR(256)  NOT NULL,
    provider_name VARCHAR(256),
    provider_address VARCHAR(256),
    insurance_type VARCHAR(256),
    PRIMARY KEY (policy_id)
);

-- Teams Table

CREATE TABLE Teams (
    team_name VARCHAR(256) NOT NULL,
    team_type VARCHAR(256),
    date_formed DATE,
```

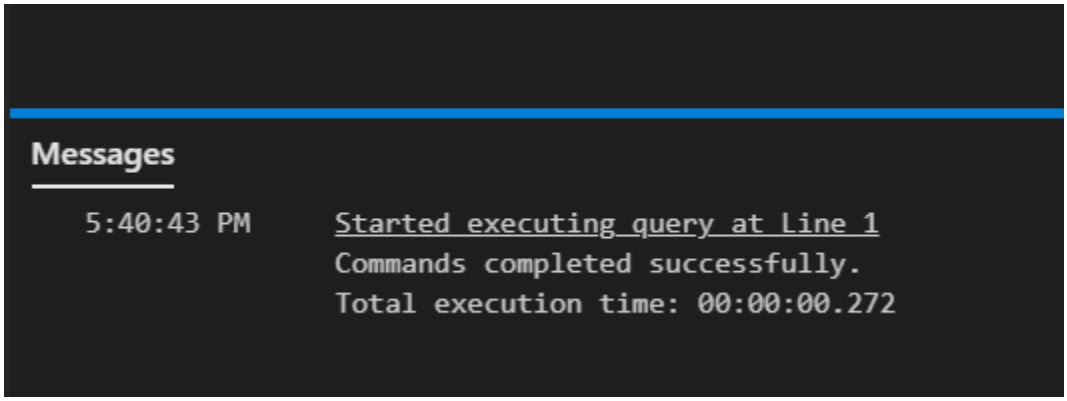
```
    PRIMARY KEY (team_name)
);

-- Expenses Table

CREATE TABLE Expenses (
    ssn INT NOT NULL,
    expense_date DATE NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    expense_description VARCHAR(256) NOT NULL,
    PRIMARY KEY (ssn, expense_date, expense_description),
    FOREIGN KEY (ssn) REFERENCES Employees(ssn)
);

-- DonationChecks Table

CREATE TABLE DonationChecks (
    ssn INT NOT NULL,
    check_number VARCHAR(256) NOT NULL,
    donation_date DATE NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    donation_type VARCHAR(256) NOT NULL,
    campaign_name VARCHAR(256) NOT NULL,
    PRIMARY KEY (ssn, donation_date, amount, donation_type),
    FOREIGN KEY (ssn) REFERENCES Donors(ssn)
);

-- CreditCards Table

CREATE TABLE CreditCards (
    ssn INT NOT NULL,
    card_number VARCHAR(256) NOT NULL,
    card_type VARCHAR(256) NOT NULL,
    expiration_date VARCHAR(256) NOT NULL,
    donation_date DATE NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    donation_type VARCHAR(256) NOT NULL,
    campaign_name VARCHAR(256) NOT NULL,
    PRIMARY KEY (ssn, donation_date, amount, donation_type),
    FOREIGN KEY (ssn) REFERENCES Donors(ssn)
);

-- ServesOn Table

CREATE TABLE ServesOn (
    ssn INT NOT NULL,
    team_name VARCHAR(256) NOT NULL,
```

```
    serving_month VARCHAR(256) NOT NULL,
    served_hours INT NOT NULL,
    active BIT NOT NULL,
    PRIMARY KEY (ssn, team_name, serving_month),
    FOREIGN KEY (ssn) REFERENCES Volunteers(ssn),
    FOREIGN KEY (team_name) REFERENCES Teams(team_name)
);

-- CaresFor Table

CREATE TABLE CaresFor (
    ssn INT NOT NULL,
    team_name VARCHAR(256) NOT NULL,
    client_active BIT NOT NULL,
    PRIMARY KEY (ssn, team_name),
    FOREIGN KEY (ssn) REFERENCES Clients(ssn) ON DELETE CASCADE,
    FOREIGN KEY (team_name) REFERENCES Teams(team_name)
);

-- Reports Table

CREATE TABLE Reports (
    ssn INT NOT NULL,
    team_name VARCHAR(256) NOT NULL,
    report_date DATE NOT NULL,
    report_description VARCHAR(512) NOT NULL,
    PRIMARY KEY (ssn, team_name),
    FOREIGN KEY (ssn) REFERENCES Employees(ssn),
    FOREIGN KEY (team_name) REFERENCES Teams(team_name)
);

-- Leads Table

CREATE TABLE Leads (
    ssn INT NOT NULL,
    team_name VARCHAR(256) NOT NULL,
    is_leader BIT NOT NULL,
    PRIMARY KEY (ssn, team_name),
    FOREIGN KEY (ssn) REFERENCES Volunteers(ssn),
    FOREIGN KEY (team_name) REFERENCES Teams(team_name)
);

-- Has Table

CREATE TABLE Has (
    ssn INT NOT NULL,
    policy_id VARCHAR(256) NOT NULL,
```

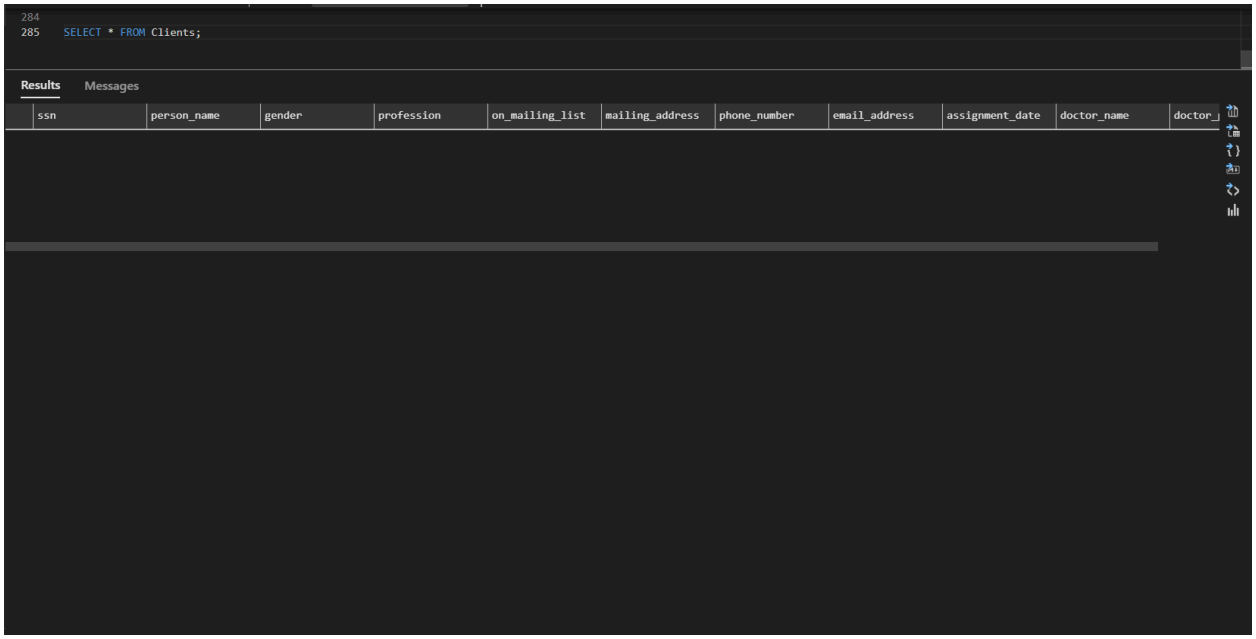
```
    PRIMARY KEY (ssn, policy_id),
    FOREIGN KEY (ssn) REFERENCES Clients(ssn),
    FOREIGN KEY (policy_id) REFERENCES InsurancePolicies(policy_id)
);

-- Needs Table

CREATE TABLE Needs (
    ssn INT NOT NULL,
    need_type VARCHAR(256) NOT NULL,
    importance INT NOT NULL,
    PRIMARY KEY (ssn, need_type),
    FOREIGN KEY (ssn) REFERENCES Clients(ssn),
    CHECK (importance >= 1 AND importance <= 10)
);

-- --Creating indexes
--Clients index
DROP INDEX IF EXISTS Clients.C_ssn;
CREATE INDEX C_ssn ON Clients(ssn);
--Volunteers index
DROP INDEX IF EXISTS Volunteers.V_ssn;
CREATE INDEX V_ssn ON Volunteers(ssn);
--Employees index
DROP INDEX IF EXISTS Employees.E_ssn;
CREATE INDEX E_ssn ON Employees(ssn);
--Donors index
DROP INDEX IF EXISTS Donors.D_ssn;
CREATE INDEX D_ssn ON Donors(ssn);
--ClientEmergencyContacts index
DROP INDEX IF EXISTS ClientEmergencyContacts.CEM_ssn;
CREATE INDEX CEM_ssn ON ClientEmergencyContacts(client_ssn);
--VolunteerEmergencyContacts index
DROP INDEX IF EXISTS VolunteerEmergencyContacts.VEM_ssn;
CREATE INDEX VEM_ssn ON VolunteerEmergencyContacts(volunteer_ssn);
--EmployeeEmergencyContacts index
DROP INDEX IF EXISTS EmployeeEmergencyContacts.EEM_ssn;
CREATE INDEX EEM_ssn ON EmployeeEmergencyContacts(employee_ssn);
--DonorsEmergencyContacts index
DROP INDEX IF EXISTS DonorEmergencyContacts.CEM_ssn;
CREATE INDEX CEM_ssn ON DonorEmergencyContacts(donor_ssn);
--Teams index
DROP INDEX IF EXISTS Teams.T_name;
CREATE INDEX T_name ON Teams(team_name);
--Expenses index
DROP INDEX IF EXISTS Expenses.E_amount;
CREATE INDEX E_amount ON Expenses(amount);
```

```
--DonationChecks index
DROP INDEX IF EXISTS DonationChecks.DC_amount;
CREATE INDEX DC_amount ON DonationChecks(amount);
--CreditCards index
DROP INDEX IF EXISTS CreditCards.CC_amount;
CREATE INDEX CC_amount ON CreditCards(amount);
```

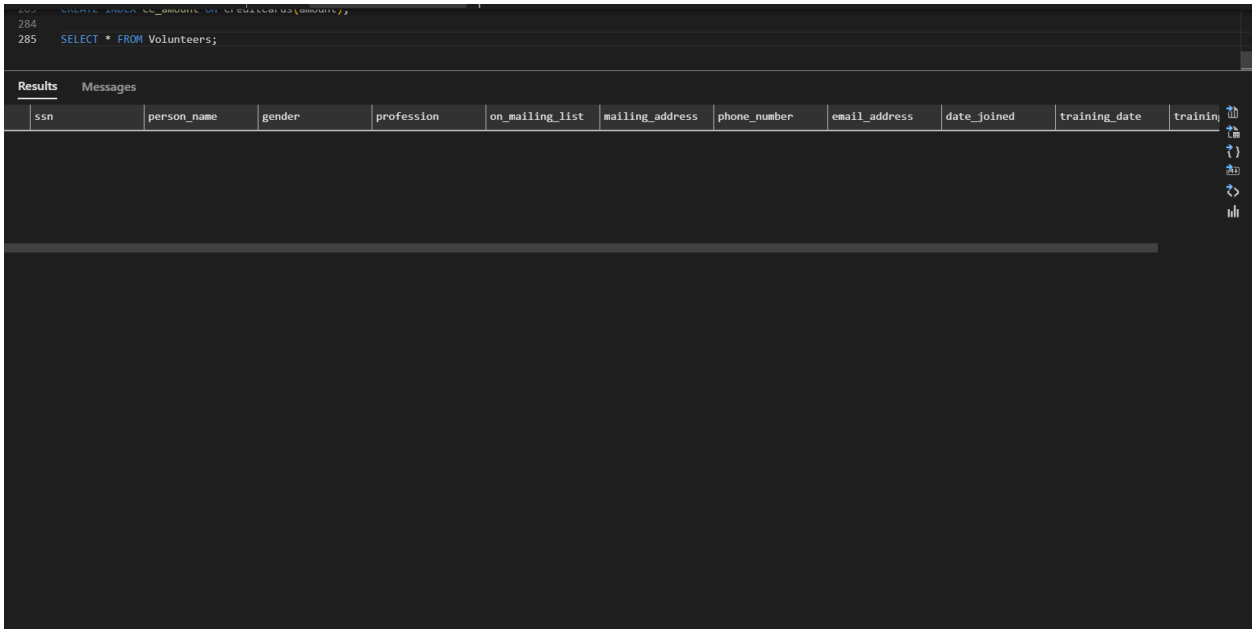
Showing all table creation compiling:



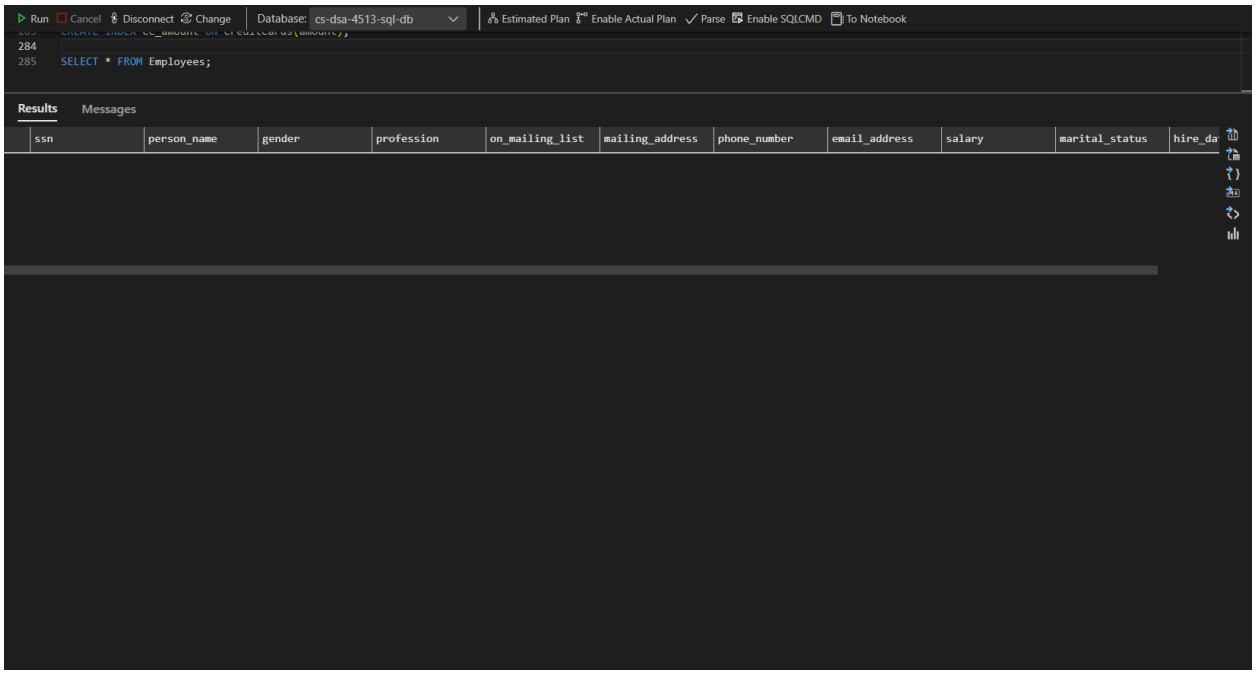
Creation of Client Table:



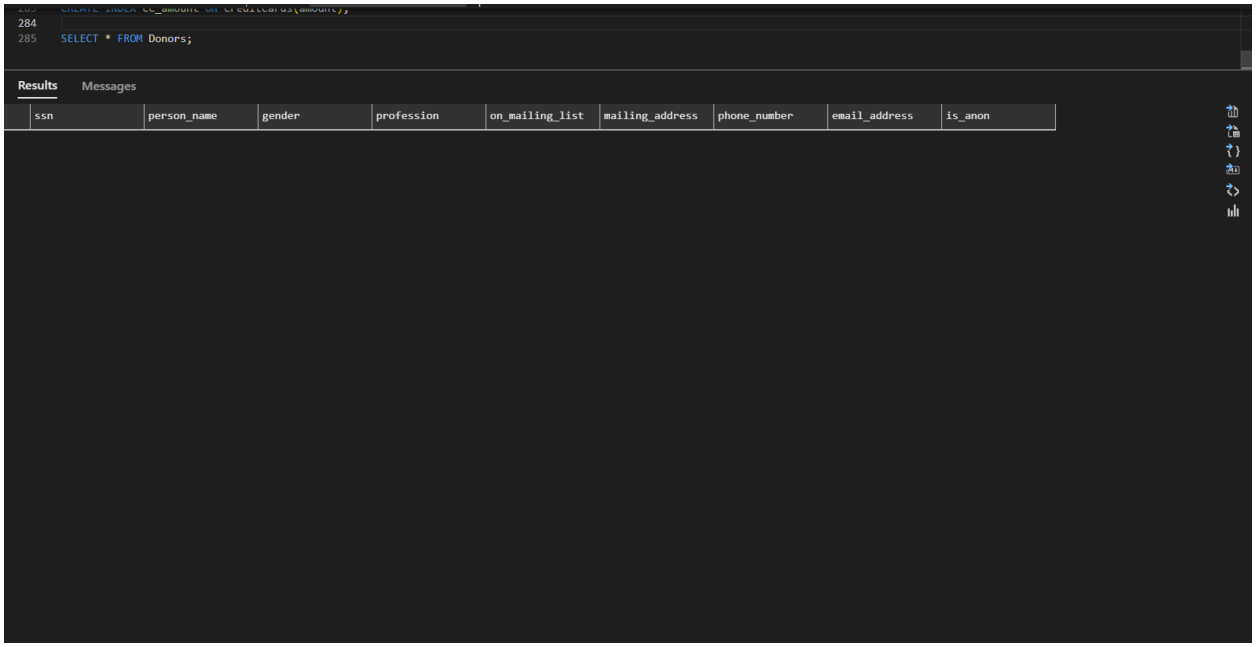
Creation of Volunteers Table:



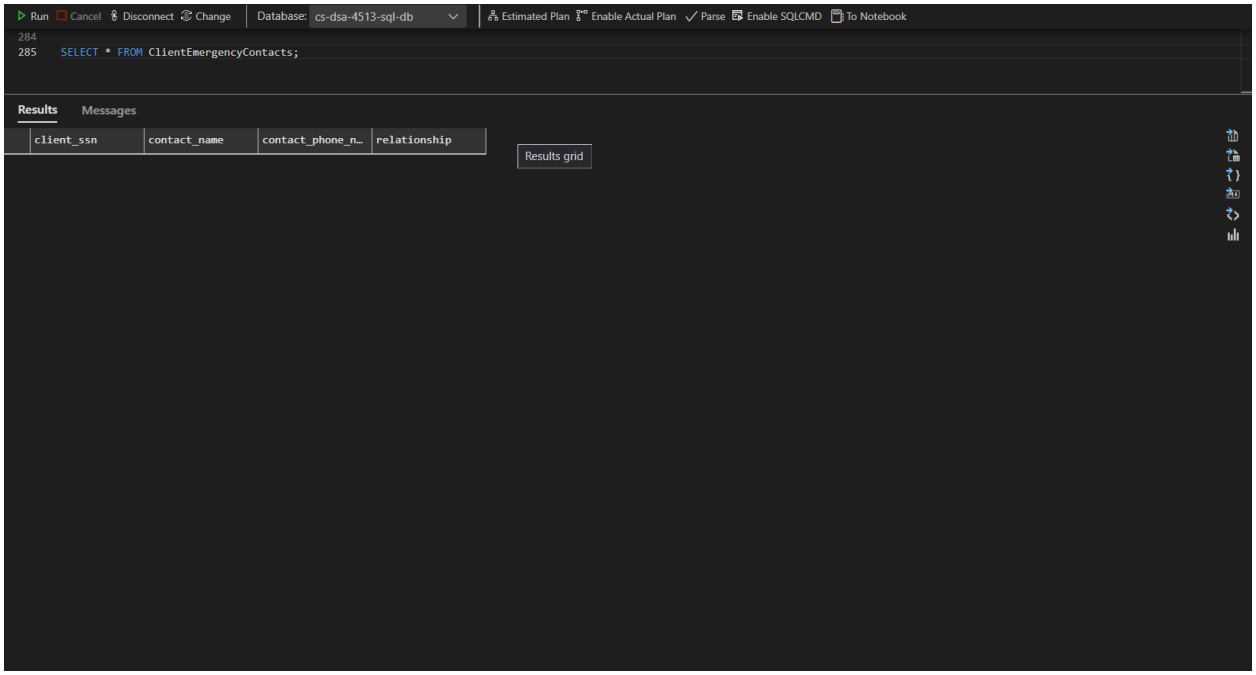
Creation of Employees Table:



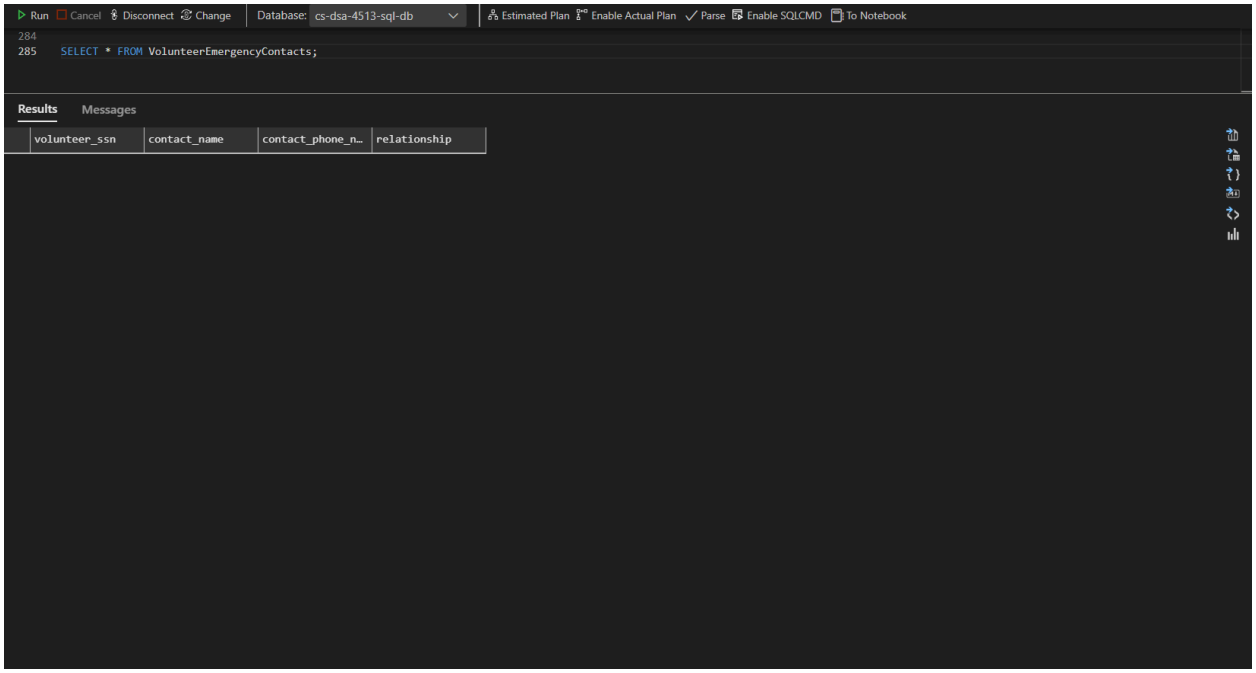
Creation of Donors Table:



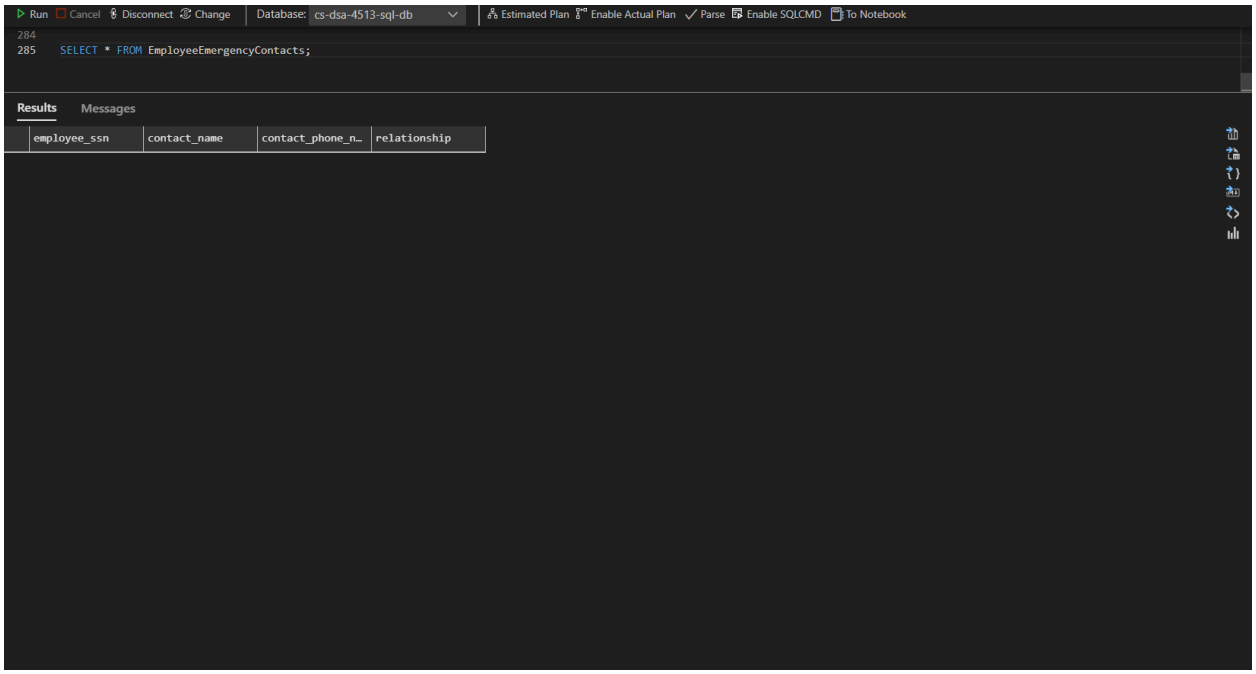
Creation of ClientEmergencyContacts Table:



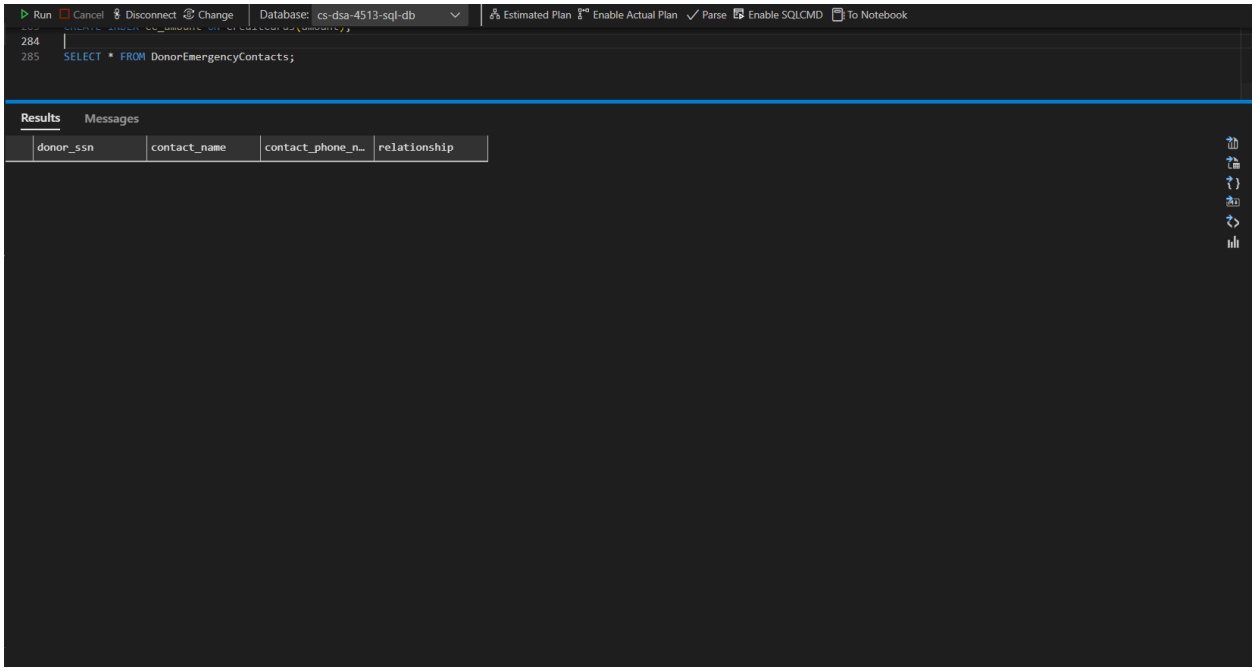
Creation of VolunteerEmergencyContacts Table:



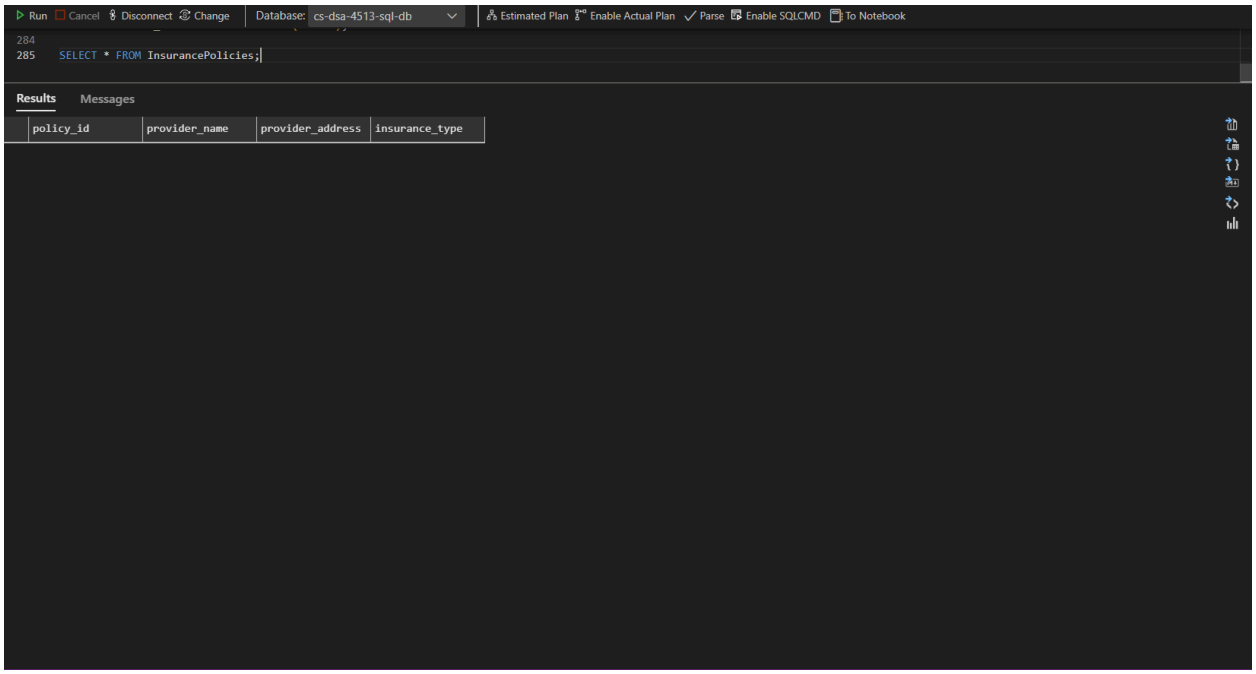
Creation of EmployeeEmergencyContacts Table:



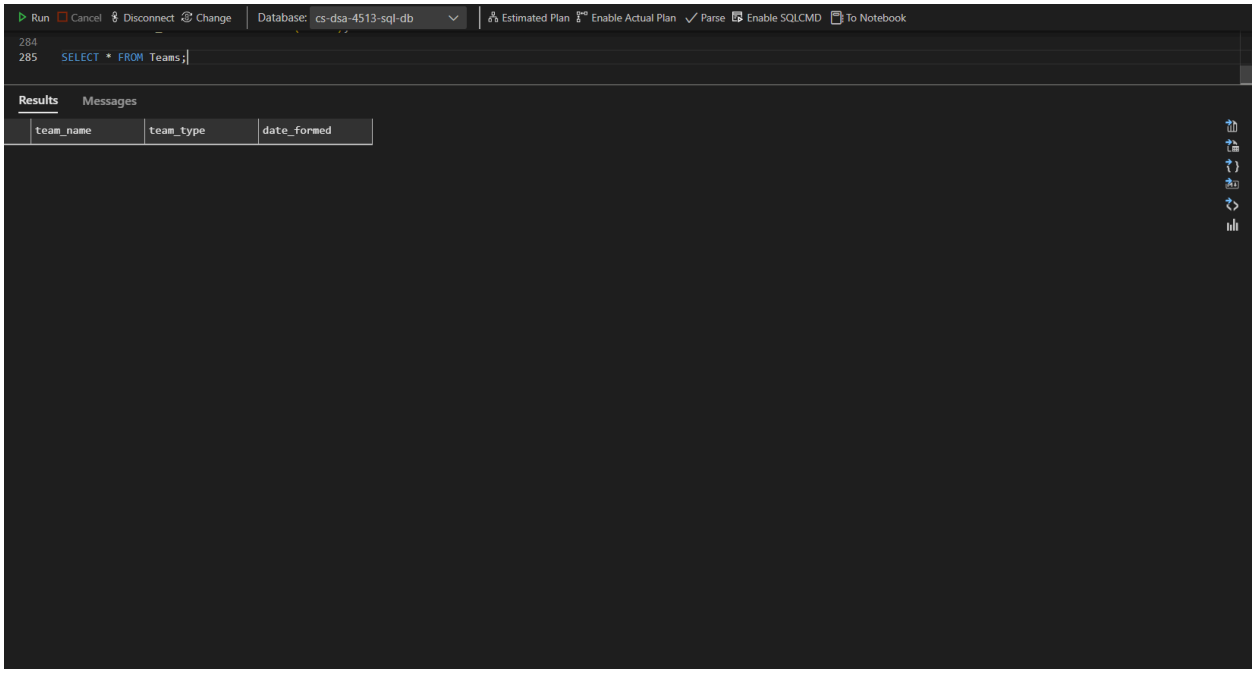
Creation of DonorEmergencyContacts Table:



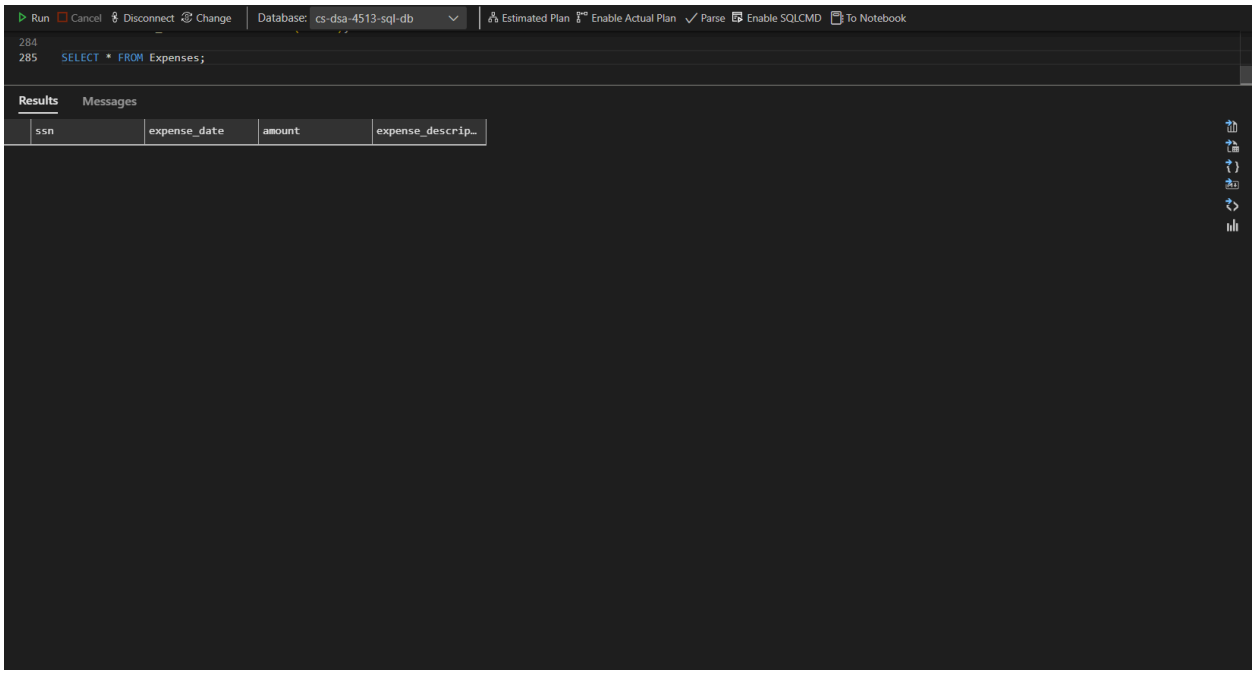
Creation of InsurancePolicies Table:



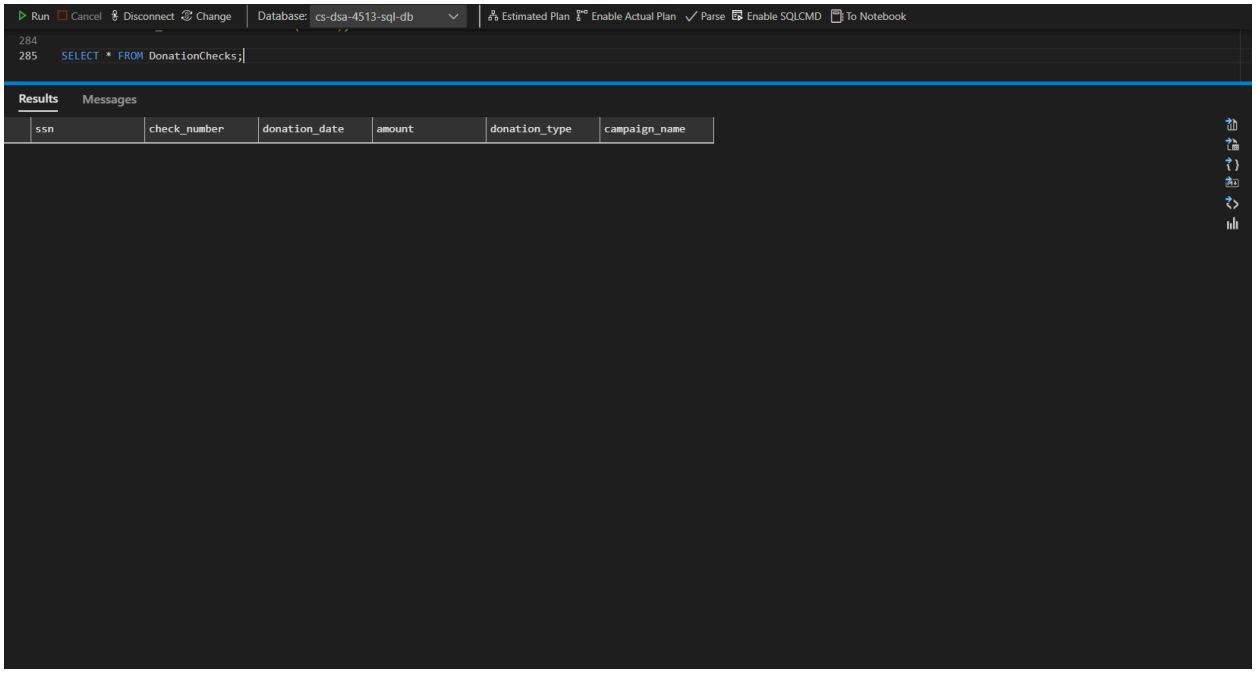
Creation of Teams Table:



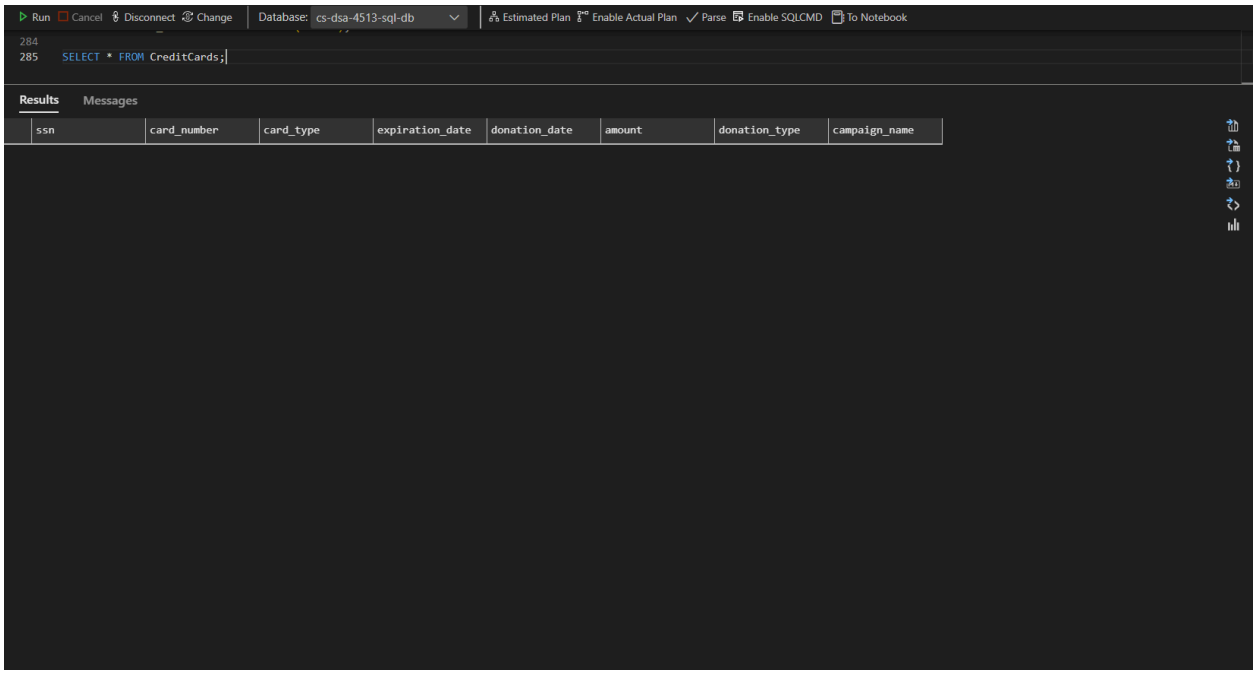
Creation of Expenses Table:



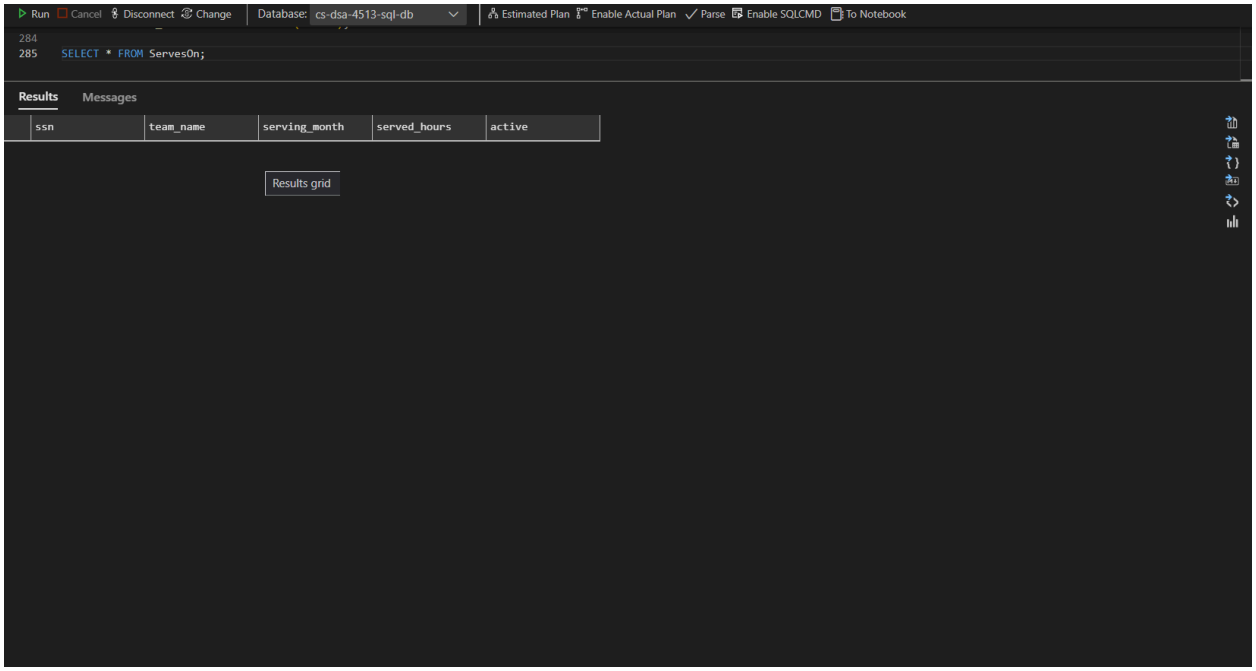
Creation of DonationChecks Table:



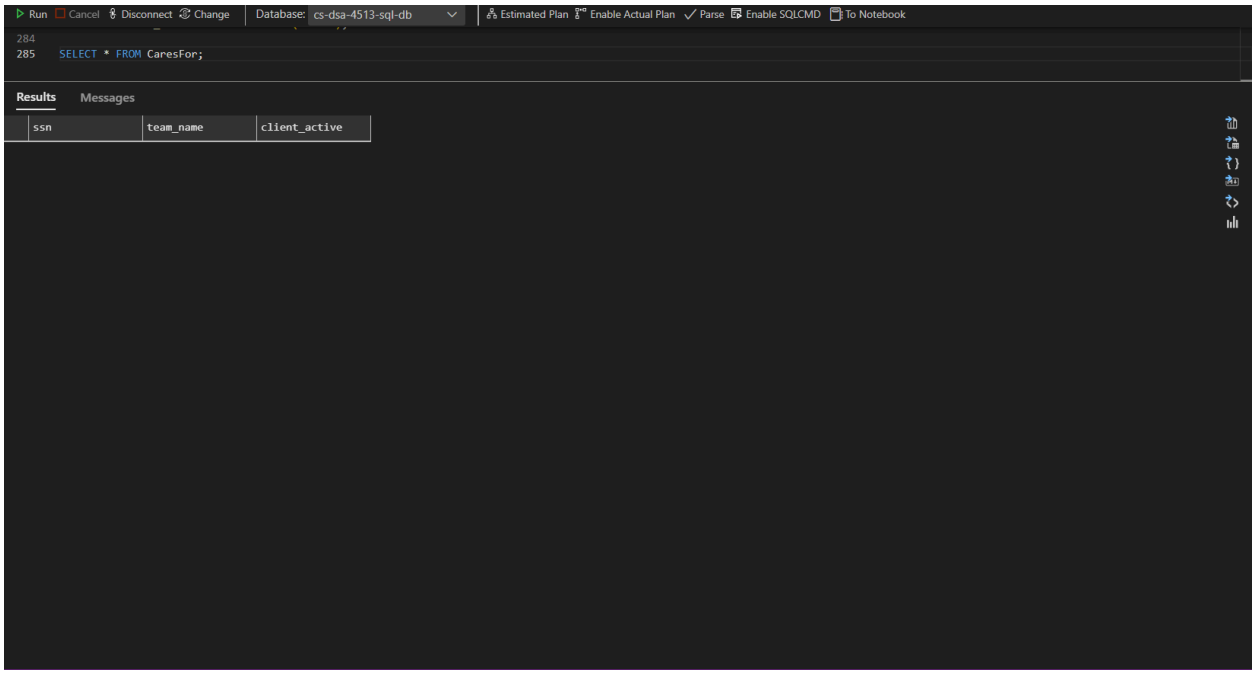
Creation of CreditCards Table:



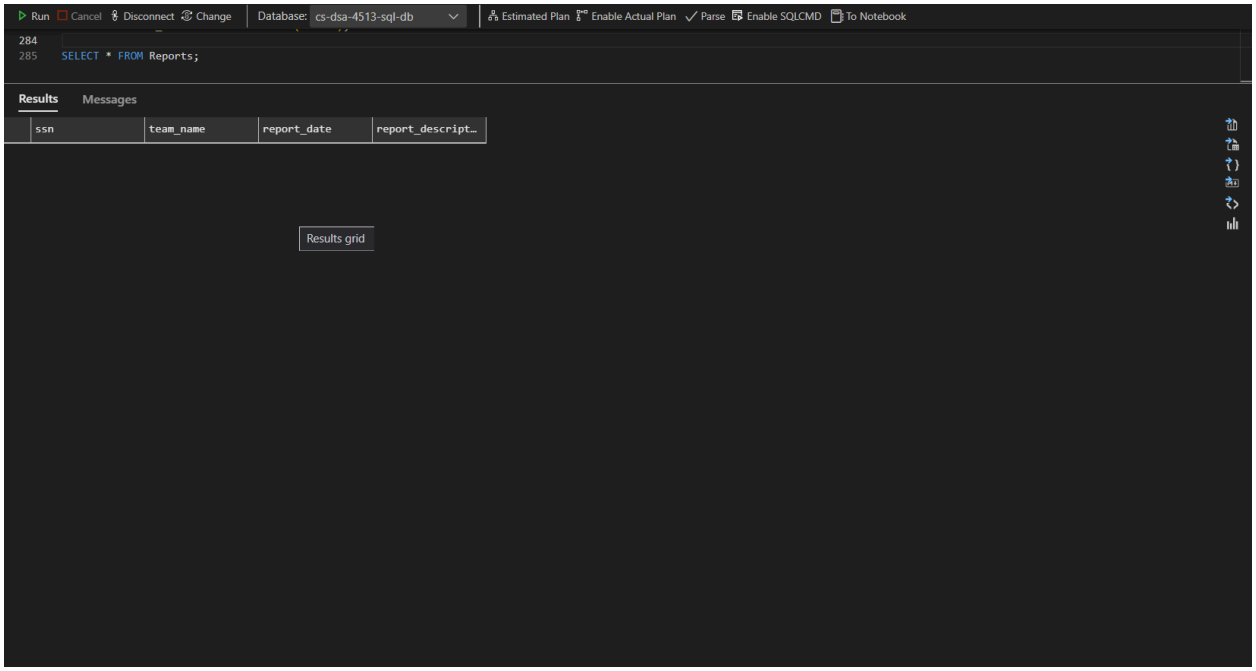
Creation of ServesOn Table:



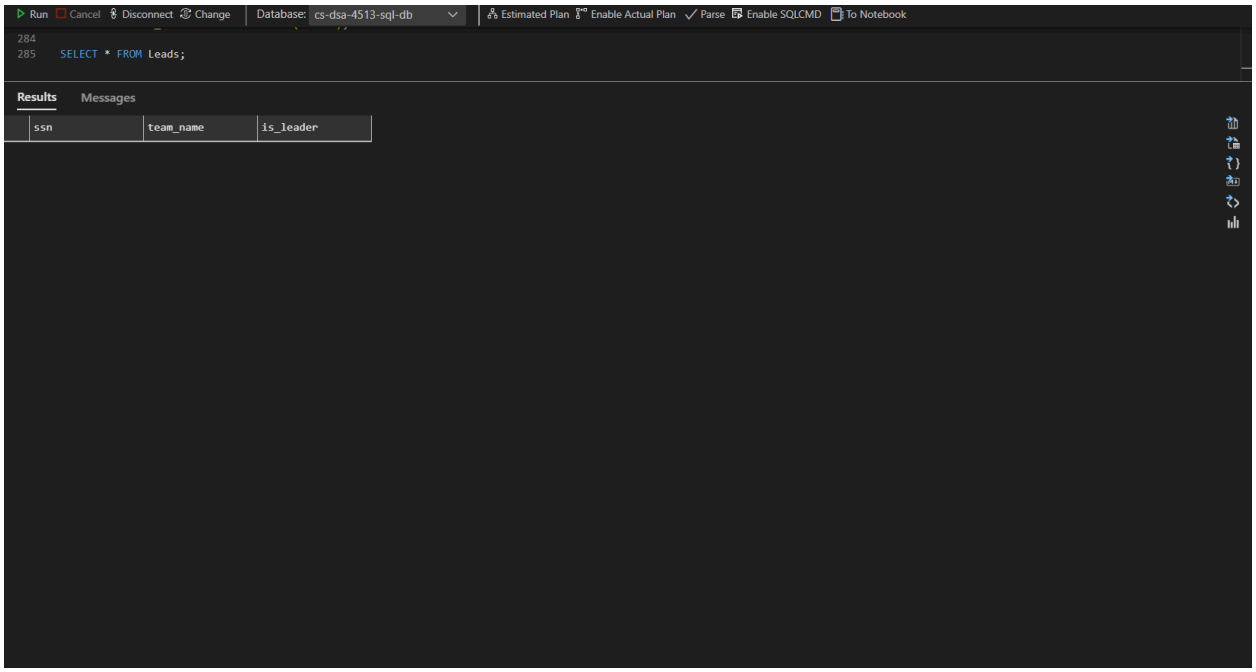
Creation of CaresFor Table:



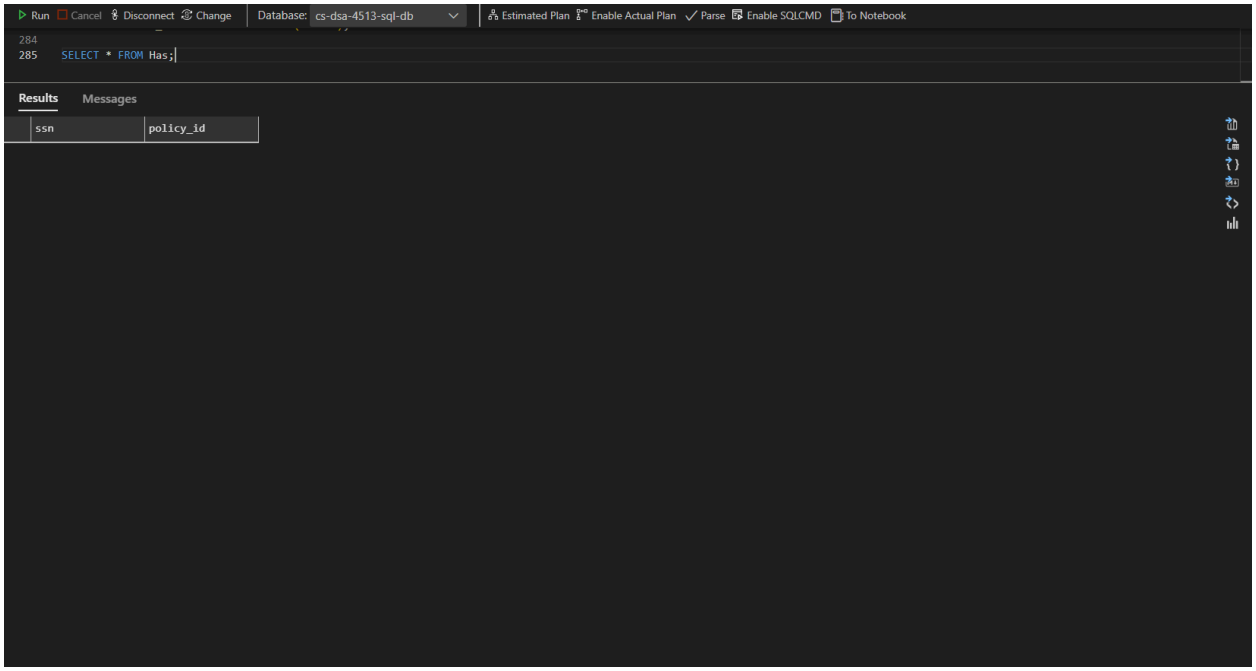
Creation of Reports Table:



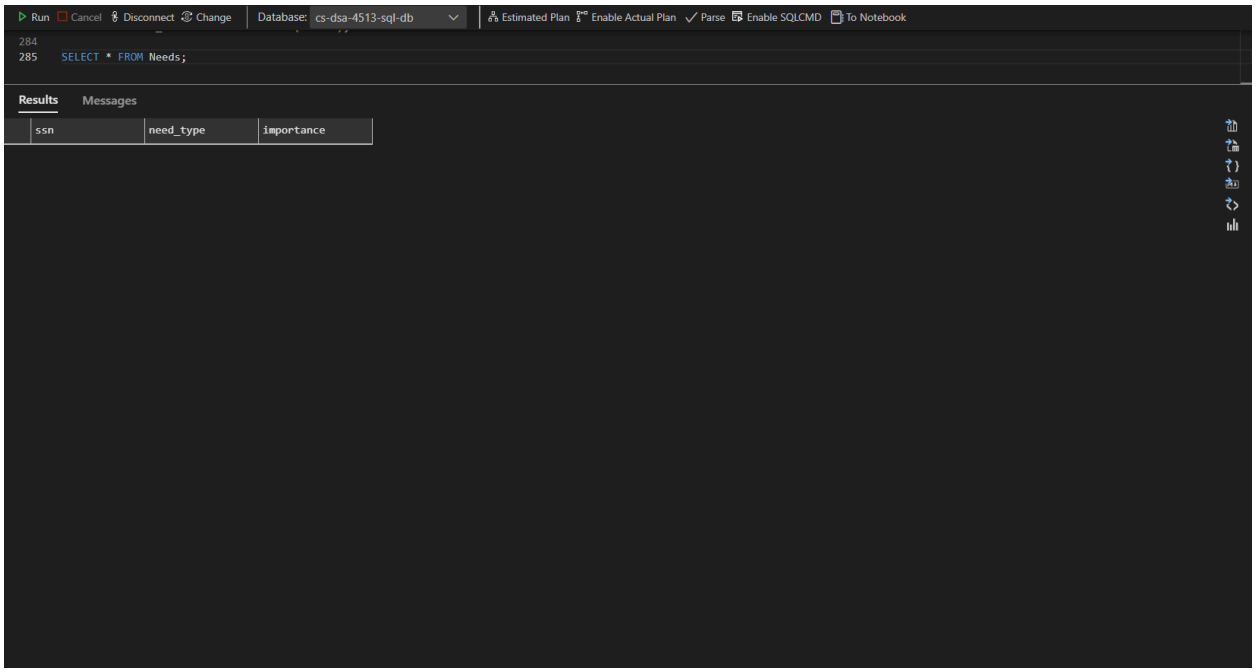
Creation of Leads Table:



Creation of Has Table:



Creation of Needs Table:



Task 5:

Task 5.1:

Stored Procedures:

```
-- Query 1: Insert a new team into the database
DROP PROCEDURE IF EXISTS insertTeam;
GO

CREATE PROCEDURE insertTeam
(
    @team_name VARCHAR(256), -- Team name
    @team_type VARCHAR(256), -- Team type
    @date_formed DATE -- Date formed
)
AS
BEGIN
    INSERT INTO Teams (team_name, team_type, date_formed) VALUES (@team_name,
@team_type, @date_formed);
END;
GO

-- Query 2: Insert a new client into the database and associate with one or more teams
DROP PROCEDURE IF EXISTS insertClient;
GO

CREATE PROCEDURE insertClient
(
    @ssn INT, -- Social Security Number
    @person_name VARCHAR(256), -- Person name
    @gender VARCHAR(256), -- gender
    @profession VARCHAR(256), -- profession
    @on_mailing_list BIT, -- on mailing list
    @mailing_address VARCHAR(256), -- mailing address
    @email_address VARCHAR(256), -- email address
    @phone_number VARCHAR(256), -- phone number
    @assignment_date DATE, -- assignment date
    @doctor_name VARCHAR(256), -- doctor name
    @doctor_phone_number VARCHAR(256), -- doctor phone number
    -- @emergency_contact_name VARCHAR(256), -- emergency contact name
    -- @contact_phone_number VARCHAR(256), -- contact phone number
    -- @relationship VARCHAR(256), -- relationship
    @need_type VARCHAR(256), -- need type
    @importance INT, -- importance
    @policy_id VARCHAR(256), -- policy id
    @provider_name VARCHAR(256), -- provider name
```

```
    @provider_address VARCHAR(256), -- provider address
    @insurance_type VARCHAR(256) -- insurance type
)
AS
BEGIN

    -- Insert into Clients table
    INSERT INTO Clients (ssn, person_name, gender, profession, on_mailing_list,
mailing_address, phone_number, email_address, assignment_date, doctor_name,
doctor_phone_number)
    VALUES (@ssn, @person_name, @gender, @profession, @on_mailing_list,
@mailing_address, @phone_number, @email_address, @assignment_date, @doctor_name,
@doctor_phone_number);

    -- Insert into Needs table
    INSERT INTO Needs (ssn, need_type, importance)
    VALUES (@ssn, @need_type, @importance);

    -- Insert into InsurancePolicies table
    INSERT INTO InsurancePolicies (policy_id, provider_name, provider_address,
insurance_type)
    VALUES (@policy_id, @provider_name, @provider_address, @insurance_type);

END;
GO

-- Query 3: Insert a new volunteer into the database and associate with one or more
teams
DROP PROCEDURE IF EXISTS insertVolunteer;
GO

CREATE PROCEDURE insertVolunteer
(
    @ssn INT, -- Social Security Number
    @person_name VARCHAR(256), -- Person name
    @gender VARCHAR(256), -- gender
    @profession VARCHAR(256), -- profession
    @on_mailing_list BIT, -- on mailing list
    @mailing_address VARCHAR(256), -- mailing address
    @email_address VARCHAR(256), -- email address
    @phone_number VARCHAR(256), -- phone number
    @date_joined DATE, -- date joined
    @training_date DATE, -- training date
    @training_location VARCHAR(256) -- training location
)
AS
BEGIN
```

```
    -- Insert into Volunteers table
    INSERT INTO Volunteers (ssn, person_name, gender, profession, on_mailing_list,
mailing_address, phone_number, email_address, date_joined, training_date,
training_location)
    VALUES (@ssn, @person_name, @gender, @profession, @on_mailing_list,
@mailing_address, @phone_number, @email_address, @date_joined, @training_date,
@training_location);

END;
GO

-- Query 4: Insert number of hours a volunteer worked this month for a particular team
DROP PROCEDURE IF EXISTS InsertVolunteerHours;
GO

CREATE PROCEDURE InsertVolunteerHours
(
    @ssn INT, -- ssn
    @team_name VARCHAR(256), -- team_name
    @serving_month VARCHAR(256), -- serving_month
    @served_hours INT, -- served_hours
    @active BIT -- active
)
AS
BEGIN

    -- Update ServeOn table for hours from a volunteer if month is already present
    IF EXISTS (SELECT 1 FROM ServesOn WHERE ssn = @ssn AND team_name = @team_name AND
serving_month = @serving_month)
    BEGIN
        UPDATE ServesOn
        SET served_hours = @served_hours
        WHERE ssn = @ssn AND team_name = @team_name AND serving_month =
@serving_month;
    END
    ELSE
    BEGIN
        -- Insert into ServesOn table
        INSERT INTO ServesOn (ssn, team_name, serving_month, served_hours, active)
        VALUES (@ssn, @team_name, @serving_month, @served_hours, @active);
    END;
END;
GO
```

```
-- Query 5: Insert a new employee into the database and associate with one or more
teams
DROP PROCEDURE IF EXISTS insertEmployee;
GO

CREATE PROCEDURE insertEmployee
(
    @ssn INT, -- ssn
    @person_name VARCHAR(256), -- Person name
    @gender VARCHAR(256), -- gender
    @profession VARCHAR(256), -- profession
    @on_mailing_list BIT, -- on mailing list
    @mailing_address VARCHAR(256), -- mailing address
    @phone_number VARCHAR(256), -- phone number
    @email_address VARCHAR(256), -- email address
    @salary DECIMAL(10,2), -- salary
    @marital_status VARCHAR(256), -- marital status
    @hire_date DATE -- hire date

)
AS
BEGIN
    -- Insert into Employees table
    INSERT INTO Employees (ssn, person_name, gender, profession, on_mailing_list,
mailing_address, phone_number, email_address, salary, marital_status, hire_date)
    VALUES (@ssn, @person_name, @gender, @profession, @on_mailing_list,
@mailing_address, @phone_number, @email_address, @salary, @marital_status,
@hire_date);

END;
GO


-- Query 6: Insert an expense charged by an employee
DROP PROCEDURE IF EXISTS insertExpense;
GO

CREATE PROCEDURE insertExpense
(
    @ssn INT, -- ssn
    @expense_date DATE, -- Expense date
    @amount DECIMAL(10, 2), -- Amount
    @expense_description VARCHAR(256) -- Expense description
)
AS
BEGIN
    INSERT INTO Expenses (ssn, expense_date, amount, expense_description)
```

```
    VALUES (@ssn, @expense_date, @amount, @expense_description);
END;
GO

-- Query 7: Insert a new donor and associate with multiple donations
DROP PROCEDURE IF EXISTS insertDonor;
GO

CREATE PROCEDURE insertDonor
(
    @ssn INT, -- ssn
    @person_name VARCHAR(256), -- Person name
    @gender VARCHAR(256), -- gender
    @profession VARCHAR(256), -- profession
    @on_mailing_list BIT, -- on mailing list
    @mailing_address VARCHAR(256), -- mailing address
    @phone_number VARCHAR(256), -- phone number
    @email_address VARCHAR(256), -- email address
    @is_anonymous BIT -- is anonymous
)
AS
BEGIN
    -- Insert into Donors table
    INSERT INTO Donors (ssn, person_name, gender, profession, on_mailing_list,
mailing_address, phone_number, email_address, is_anon)
    VALUES (@ssn, @person_name, @gender, @profession, @on_mailing_list,
@mailing_address, @phone_number, @email_address, @is_anonymous);


END;
GO


-- Query 8: Retrieve the name and phone number of the doctor of a particular client
(1/week)
DROP PROCEDURE IF EXISTS retrieveDoctorInfo;
GO

CREATE PROCEDURE retrieveDoctorInfo
(
    @ssn INT -- ssn
)
AS
BEGIN
    -- Retrieve the name and phone number of the doctor of a particular client
    SELECT doctor_name, doctor_phone_number
    FROM Clients
```

```
    WHERE ssn = @ssn;
END;
GO

-- Query 9: Retrieve total expenses charged by each employee for a particular period,
sorted by amount (1/month)
DROP PROCEDURE IF EXISTS retrieveTotalExpenses;
GO

CREATE PROCEDURE retrieveTotalExpenses
(
    @start_date DATE, -- Start date
    @end_date DATE -- End date
)
AS
BEGIN
    -- Retrieve total expenses charged by each employee for a particular period,
sorted by amount
    SELECT ssn, SUM(amount) AS total_expenses
    FROM Expenses
    WHERE expense_date BETWEEN @start_date AND @end_date
    GROUP BY 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 (4/year).
DROP PROCEDURE IF EXISTS GetVolunteersForClient;
GO

CREATE PROCEDURE GetVolunteersForClient
(
    @ssn INT -- ssn
)
AS
BEGIN
    SELECT v.person_name
    FROM Volunteers v
    JOIN ServesOn s ON v.ssn = s.ssn
    JOIN CaresFor c ON s.team_name = c.team_name
    WHERE c.ssn = @ssn;
END
GO

-- Query 11: Retrieve the names of all teams founded after a specified date (1/month)
DROP PROCEDURE IF EXISTS getTeamsAfterDate;
```

```
GO

CREATE PROCEDURE getTeamsAfterDate
(
    @date_formed DATE -- Date formed
)
AS
BEGIN
    -- Retrieve names of teams founded after a specified date
    SELECT team_name
    FROM Teams
    WHERE date_formed > @date_formed;
END;
GO

-- Query 12: Retrieve names, SSNs, contact information, and emergency contacts for all
people (1/week)
DROP PROCEDURE IF EXISTS getAllPeopleInfo;
GO

CREATE PROCEDURE getAllPeopleInfo
AS
BEGIN
    -- Retrieve names, SSNs, contact information, and emergency contacts for all
people
    SELECT e.person_name, e.ssn, e.mailing_address, e.phone_number, e.email_address,
ec.contact_name AS emergency_contact_name, ec.contact_phone_number AS
emergency_contact_phone, ec.relationship AS emergency_contact_relationship
    FROM Employees e
    LEFT JOIN EmployeeEmergencyContacts ec ON e.ssn = ec.employee_ssn
    UNION ALL
    SELECT v.person_name, v.ssn, v.mailing_address, v.phone_number, v.email_address,
vc.contact_name AS emergency_contact_name, vc.contact_phone_number AS
emergency_contact_phone, vc.relationship AS emergency_contact_relationship
    FROM Volunteers v
    LEFT JOIN VolunteerEmergencyContacts vc ON v.ssn = vc.volunteer_ssn
    UNION ALL
    SELECT c.person_name, c.ssn, c.mailing_address, c.phone_number, c.email_address,
cc.contact_name AS emergency_contact_name, cc.contact_phone_number AS
emergency_contact_phone, cc.relationship AS emergency_contact_relationship
    FROM Clients c
    LEFT JOIN ClientEmergencyContacts cc ON c.ssn = cc.client_ssn
    UNION ALL
    SELECT d.person_name, d.ssn, d.mailing_address, d.phone_number, d.email_address,
dc.contact_name AS emergency_contact_name, dc.contact_phone_number AS
emergency_contact_phone, dc.relationship AS emergency_contact_relationship
    FROM Donors d
```

```
    LEFT JOIN DonorEmergencyContacts dc ON d.ssn = dc.donor_ssn;
END;
GO

-- Query 13: Retrieve the name and total donations by donors who are also employees,
sorted by amount (1/week)

DROP PROCEDURE IF EXISTS GetEmployeeDonors;
GO

CREATE PROCEDURE GetEmployeeDonors
AS
BEGIN
    -- Retrieve total donations by donors who are also employees, sorted by amount
    SELECT
        d.person_name,
        CASE
            WHEN d.is_anon = 1 THEN 'Yes'
            ELSE 'No'
        END AS is_anon,
        SUM(all_don.amount) AS total_donations
    FROM Donors d
    JOIN (
        -- Combine DonationCard and DonationCheck
        SELECT ssn, amount
        FROM CreditCards
        UNION ALL
        SELECT ssn, amount
        FROM DonationChecks
    ) all_don ON d.ssn = all_don.ssn
    JOIN Employees e ON d.ssn = e.ssn
    GROUP BY d.person_name, d.is_anon
    ORDER BY total_donations DESC;
END;
GO

-- Query 14: Increase salary by 10% for employees with reports from more than one team
(1/year)
DROP PROCEDURE IF EXISTS increaseSalaryForMultiTeamEmployees;
GO

CREATE PROCEDURE increaseSalaryForMultiTeamEmployees
AS
BEGIN
    -- Increase salary by 10% for employees reporting on multiple teams
    UPDATE Employees
    SET salary = salary * 1.1
```

```
    WHERE ssn IN (
        SELECT ssn
        FROM Reports
        GROUP BY 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 (4/year).

DROP PROCEDURE IF EXISTS deleteClients;
GO

CREATE PROCEDURE deleteClients
AS
BEGIN
    DECLARE @deleteClients TABLE (ssn INT); -- Temporary table to store SSNs of
clients to delete

    -- Insert the SSNs of clients who meet the criteria for deletion
    -- Insert the SSNs of clients who meet the criteria for deletion
    INSERT INTO @deleteClients (ssn)
    SELECT c.ssn
    FROM Clients c
    JOIN Has h ON c.ssn = h.ssn
    JOIN InsurancePolicies ip ON h.policy_id = ip.policy_id
    WHERE ip.insurance_type != 'Health'
    AND c.ssn IN (
        SELECT ssn
        FROM Needs
        WHERE need_type = 'Transportation'
        AND importance < 5
    );

    -- Delete from the 'Needs' table
    DELETE FROM Needs
    WHERE ssn IN (SELECT ssn FROM @deleteClients);

    -- Delete from the 'Has' table
    DELETE FROM Has
    WHERE ssn IN (SELECT ssn FROM @deleteClients);

    -- Delete from the 'InsurancePolicies' table if not referenced in the 'Has' table
    DELETE FROM InsurancePolicies
    WHERE policy_id NOT IN (SELECT policy_id FROM Has);
```

```
    -- Delete from 'ClientEmergencyContacts' table
    DELETE FROM ClientEmergencyContacts
    WHERE client_ssn IN (SELECT ssn FROM @deleteClients);

    -- Finally, delete from 'Clients' table
    DELETE FROM Clients
    WHERE ssn IN (SELECT ssn FROM @deleteClients);
END
GO

-- Query 17: Retrieve names and mailing addresses of all people on the mailing list
DROP PROCEDURE IF EXISTS executeExportMailingList;
GO

CREATE PROCEDURE executeExportMailingList
AS
BEGIN
    -- Retrieve names and mailing addresses of all people on the mailing list
    SELECT person_name, mailing_address
    FROM Employees
    WHERE on_mailing_list = 1
    UNION ALL
    SELECT person_name, mailing_address
    FROM Volunteers
    WHERE on_mailing_list = 1
    UNION ALL
    SELECT person_name, mailing_address
    FROM Clients
    WHERE on_mailing_list = 1
    UNION ALL
    SELECT person_name, mailing_address
    FROM Donors
    WHERE on_mailing_list = 1;
END;
```

Successful Compilation of Stored Procedures:

```
4:09:51 PMStarted executing query at Line 1
Commands completed successfully.
4:09:51 PMStarted executing query at Line 4
Commands completed successfully.
4:09:51 PMStarted executing query at Line 16
Commands completed successfully.
4:09:51 PMStarted executing query at Line 20
Commands completed successfully.
4:09:51 PMStarted executing query at Line 61
Commands completed successfully.
4:09:51 PMStarted executing query at Line 65
```

```
Commands completed successfully.
4:09:51 PMStarted executing query at Line 89
Commands completed successfully.
4:09:51 PMStarted executing query at Line 93
Commands completed successfully.
4:09:51 PMStarted executing query at Line 120
Commands completed successfully.
4:09:51 PMStarted executing query at Line 125
Commands completed successfully.
4:09:51 PMStarted executing query at Line 149
Commands completed successfully.
4:09:51 PMStarted executing query at Line 154
Commands completed successfully.
4:09:51 PMStarted executing query at Line 168
Commands completed successfully.
4:09:51 PMStarted executing query at Line 172
Commands completed successfully.
4:09:51 PMStarted executing query at Line 194
Commands completed successfully.
4:09:51 PMStarted executing query at Line 199
Commands completed successfully.
4:09:51 PMStarted executing query at Line 212
Commands completed successfully.
4:09:51 PMStarted executing query at Line 216
Commands completed successfully.
4:09:51 PMStarted executing query at Line 232
Commands completed successfully.
4:09:51 PMStarted executing query at Line 236
Commands completed successfully.
4:09:51 PMStarted executing query at Line 250
Commands completed successfully.
4:09:51 PMStarted executing query at Line 254
Commands completed successfully.
4:09:51 PMStarted executing query at Line 267
Commands completed successfully.
4:09:51 PMStarted executing query at Line 271
Commands completed successfully.
4:09:51 PMStarted executing query at Line 293
Commands completed successfully.
4:09:51 PMStarted executing query at Line 298
Commands completed successfully.
4:09:51 PMStarted executing query at Line 311
Commands completed successfully.
4:09:51 PMStarted executing query at Line 316
Commands completed successfully.
4:09:51 PMStarted executing query at Line 331
Commands completed successfully.
4:09:51 PMStarted executing query at Line 336
Commands completed successfully.
4:09:51 PMStarted executing query at Line 378
Commands completed successfully.
4:09:51 PMStarted executing query at Line 382
Commands completed successfully.
Total execution time: 00:00:01.092
```

Task 5.2:

Java Code:

```
package org.example;
import java.io.*;
import java.sql.*;
import java.sql.Date;
import java.util.*;
import java.util.logging.Logger;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.math.BigDecimal;

public class Benbrook_Tanner_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(Benbrook_Tanner_IP_Task5b.class.getName()); //
Initialize the logger
   }

   public static void main(String[] args) throws Exception {

       Properties properties = new Properties(); // Load database properties

properties.load(Benbrook_Tanner_IP_Task5b.class.getClassLoader().getResourceAsS
tream("application.properties")); // Load properties file

       Connection connection =
DriverManager.getConnection(properties.getProperty("url"), properties); //
Establish connection

       Scanner inputScanner = new Scanner(System.in); // Initialize scanner for
user input
       int query = 0; // Initialize query input variable
       System.out.print("WELCOME TO PAN, YOUR OWN PERSONALIZED DATABASE
SYSTEM!\n");

       while (query != 18) { // Loop until user selects option 18 (quit)
           System.out.println("""
                   Choose an option:\s
                   1: Insert a Team into the database\s
                   2: Insert a Client into the database\s
                   3: Insert a Volunteer into the database\s
                   4: Insert number of hours volunteer worked this month\s
                   5: Insert an Employee into the database\s
                   6: Insert an Expense associated with an Employee\s
```

```
                    7: Insert a Donor into the database\s
                    8: Retrieve Doctor information\s
                    9: Retrieve Employee charges for a particular time period\s
                    10: Retrieve Volunteers that are members of teams that
support a particular client\s
                    11: Retrieve names of teams founded after particular dates\s
                    12: Retrieve information of all people in database\s
                    13: Retrieve name and amount donated by donors that are also
employees\s
                    14: Increase salary of employees with more than one team
reporting to them\s
                    15: Delete Clients who don't have health insurance and value
of importance for transportation is less than 5\s
                    16: Import a new team from data file\s
                    17: Export names and mailing address of people on mailing
list\s
                    18: Quit\s"""); // Prompt user
            query = inputScanner.nextInt(); // Get user input

            switch (query) { // Handle different query options
                case 1:
                    executeInsertTeamProcedure(connection, inputScanner);
                    break;
                case 2:
                    executeInsertClientProcedure(connection, inputScanner);
                    break;
                case 3:
                    executeInsertVolunteerProcedure(connection, inputScanner);
                    break;
                case 4:
                    executeInsertVolunteerHoursProcedure(connection,
inputScanner);
                    break;
                case 5:
                    executeInsertEmployeeProcedure(connection, inputScanner);
                    break;
                case 6:
                    executeInsertExpenseProcedure(connection, inputScanner);
                    break;
                case 7:
                    executeInsertDonorProcedure(connection, inputScanner);
                    break;
                case 8:
                    executeRetrieveDoctorInfoProcedure(connection,
inputScanner);
                    break;
                case 9:
                    executeRetrieveTotalExpensesProcedure(connection,
inputScanner);
```

```
                    break;
                case 10:
                    executeGetVolunteersForClientProcedure(connection,
inputScanner);
                    break;
                case 11:
                    executeGetTeamsFoundedAfterDateProcedure(connection,
inputScanner);
                    break;
                case 12:
                    executeGetPeopleInfoProcedure(connection, inputScanner);
                    break;
                case 13:
                    executeGetDonorsThatAreEmployeesProcedure(connection,
inputScanner);
                    break;
                case 14:
                    executeIncreaseSalaryForEmployeesProcedure(connection,
inputScanner);
                    break;
                case 15:
                    executeDeleteClientsProcedure(connection);
                    break;
                case 16:
                    executeImportTeamProcedure(connection, inputScanner);
                    break;
                case 17:
                    executeExportMailingList(connection, inputScanner);
                    break;
                case 18:
                    System.out.println("You have exited PAN. Have a great
day!"); // Exit program
                    break;
                default:
                    System.out.println("Invalid option. Please select a valid
option (1-18)."); // Handle invalid input
            }
        }

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

    // Execute Query 1 - Insert Team
    private static void executeInsertTeamProcedure(Connection connection,
Scanner keyboard) throws SQLException {
        System.out.println("Enter the Team Name: "); // Prompt for team name
        String teamName = keyboard.next(); // Get team name

        System.out.println("Enter the Team Type: "); // Prompt for team type
```

```
        String teamType = keyboard.next(); // Get team type

        System.out.println("Enter the date formed (yyyy-MM-dd): "); // Prompt
for team formed date
        String dateFormed = keyboard.next();

        // handle date format
        if (!isValidDate(dateFormed)) {
            System.err.println("Invalid date format. Please enter the date in
yyyy-MM-dd format.");
            return;
        }
        String sql = "{call insertTeam(?, ?, ?)}"; // Prepare the SQL call for
the procedure
        CallableStatement stmt = connection.prepareCall(sql); // Create
CallableStatement

        stmt.setString(1, teamName); // Set team name parameter
        stmt.setString(2, teamType); // Set team type parameter
        stmt.setString(3, dateFormed); // Set date formed parameter

        stmt.execute(); // Execute the stored procedure
        System.out.println("Team inserted successfully."); // Confirm success
        stmt.close(); // Close the statement
    }

    // Execute Query 2 - Insert Client
    private static void executeInsertClientProcedure(Connection connection,
Scanner keyboard) throws SQLException {
        // Prompt for client SSN
        System.out.println("Enter the client ssn:");
        int cSSN = keyboard.nextInt();
        keyboard.nextLine(); // need nextLine when reading Int

        // Prompt for client name
        System.out.println("Enter the client name: ");
        String cName = keyboard.nextLine();

        // Prompt for client gender
        System.out.println("Enter the client gender:");
        String cGender = keyboard.nextLine();

        // Prompt for client profession
        System.out.println("Enter the client profession:");
        String cProfession = keyboard.nextLine();

        // Prompt for client mailing list status
        System.out.println("Is Client on the mailing list? (1 for yes, 0 for
no):");
```

```
        int clientMailingListStatus = keyboard.nextInt();
        keyboard.nextLine();

        // Prompt for client mailing address
        System.out.println("Enter the client mailing address:");
        String clientMailingAddress = keyboard.nextLine();

        // Prompt for client phone number
        System.out.println("Enter the client phone number: ");
        String cPhoneNumber = keyboard.nextLine();

        // Prompt for client email
        System.out.println("Enter the client email: ");
        String cEmail = keyboard.nextLine();

        // Prompt for client assignment date
        System.out.println("Enter the client assignment date (yyyy-MM-dd): ");
        String cAssignmentDate = keyboard.nextLine();
        if (!isValidDate(cAssignmentDate)) {
            System.err.println("Invalid date format for assignment date. Please
enter the date in yyyy-MM-dd format.");
            return;
        }

        // Prompt for client doctor name
        System.out.println("Enter the doctor name of client:");
        String doctorName = keyboard.nextLine();

        // Prompt for client doctor phone number
        System.out.println("Enter the client doctor's phone number:");
        String doctorPhoneNumber = keyboard.nextLine();

        // Collect multiple team names
        List<String> teamNames = new ArrayList<>();
        System.out.println("How many teams care for the client? ");
        int teams = keyboard.nextInt();
        keyboard.nextLine();
        for (int i = 0; i < teams; i++) {
            System.out.println("Enter the team " + (i + 1) + " name:");
            teamNames.add(keyboard.nextLine());
        }

        // Prompt for client need type
        System.out.println("Enter the client's need type: ");
        String cNeedType = keyboard.nextLine();

        // Prompt for client need importance
        System.out.println("Enter the client's need importance (1-10): ");
        int cNeedImportance = keyboard.nextInt();
```

```
        keyboard.nextLine();

        // Prompt for client insurance policy id
        System.out.println("Enter the client's insurance policy id: ");
        String cInsurancePolicyId = keyboard.nextLine();

        // Prompt for client insurance provider
        System.out.println("Enter the client's insurance provider: ");
        String cInsuranceProvider = keyboard.nextLine();

        // Prompt for client insurance provider address
        System.out.println("Enter the client's insurance provider's address: ");
        String cInsuranceProviderAddress = keyboard.nextLine();

        // Prompt for client insurance type
        System.out.println("Enter client's insurance type: ");
        String cInsuranceType = keyboard.nextLine();

        // Execute main client insertion procedure
        String sql = "{call insertClient(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}";
        CallableStatement callableStatement = connection.prepareCall(sql);
        callableStatement.setInt(1, cSSN);
        callableStatement.setString(2, cName);
        callableStatement.setString(3, cGender);
        callableStatement.setString(4, cProfession);
        callableStatement.setInt(5, clientMailingListStatus);
        callableStatement.setString(6, clientMailingAddress);
        callableStatement.setString(7, cEmail);
        callableStatement.setString(8, cPhoneNumber);
        callableStatement.setString(9, cAssignmentDate);
        callableStatement.setString(10, doctorName);
        callableStatement.setString(11, doctorPhoneNumber);
        callableStatement.setString(12, cNeedType);
        callableStatement.setInt(13, cNeedImportance);
        callableStatement.setString(14, cInsurancePolicyId);
        callableStatement.setString(15, cInsuranceProvider);
        callableStatement.setString(16, cInsuranceProviderAddress);
        callableStatement.setString(17, cInsuranceType);

        callableStatement.execute();
        System.out.println("Client has been inserted successfully.");
        callableStatement.close();

        // Insert emergency contacts
        String emergencyContactSql = "INSERT INTO ClientEmergencyContacts
(client_ssn, contact_name, " +
                "contact_phone_number, relationship) VALUES (?, ?, ?, ?)";
        System.out.println("How many emergency contacts does the client have?");
        int emergencyContacts = keyboard.nextInt();
```

```
        keyboard.nextLine();
        for (int i = 0; i < emergencyContacts; i++) {
            System.out.println("Please enter the name of emergency contact " +
(i + 1) + ":");
            String ecName = keyboard.nextLine();
            System.out.println("Please enter the phone number of emergency
contact " + (i + 1) + ":");
            String ecPhoneNumber = keyboard.nextLine();
            System.out.println("Please enter the relationship of emergency
contact " + (i + 1) + ":");
            String ecRelationship = keyboard.nextLine();
            try (CallableStatement ecStmt =
connection.prepareCall(emergencyContactSql)) {
                ecStmt.setInt(1, cSSN);
                ecStmt.setString(2, ecName);
                ecStmt.setString(3, ecPhoneNumber);
                ecStmt.setString(4, ecRelationship);
                ecStmt.execute();
            }
        }

        // Insert additional team names into CaresFor table
        String caresFor = "INSERT INTO CaresFor (ssn, team_name, client_active)
VALUES (?, ?, ?)";
        for (int i = 0; i < teamNames.size(); i++) { // Start from 1 since the
first team was already added
            try (CallableStatement caresForStmt =
connection.prepareCall(caresFor)) {
                caresForStmt.setInt(1, cSSN);
                caresForStmt.setString(2, teamNames.get(i));
                // Get client active status
                System.out.println("Is client active for team " +
teamNames.get(i) + "? (1 for yes, 0 for no):");
                int active = keyboard.nextInt();
                keyboard.nextLine();
                caresForStmt.setInt(3, active);
                caresForStmt.execute();
            }
        }
        // Insert into Has table
        String has = "INSERT INTO Has (ssn, policy_id) VALUES (?, ?)";
        try (CallableStatement hasStmt = connection.prepareCall(has)) {
            hasStmt.setInt(1, cSSN);
            hasStmt.setString(2, cInsurancePolicyId);
            hasStmt.execute();
        }
    }

    // query 3
```

```
   // Method to insert a new volunteer into the database
   private static void executeInsertVolunteerProcedure(Connection connection,
Scanner inputScanner) throws SQLException {
       // Prompt for volunteer SSN
       System.out.println("Enter volunteer ssn:");
       int volunteerSSN = inputScanner.nextInt();
       inputScanner.nextLine(); // Consume newline left-over

       // Prompt for volunteer name
       System.out.println("Enter volunteer name: ");
       String name = inputScanner.nextLine();

       // Prompt for volunteer gender
       System.out.println("Enter volunteer gender:");
       String gender = inputScanner.nextLine();

       // Prompt for volunteer profession
       System.out.println("Enter volunteer profession:");
       String profession = inputScanner.nextLine();

       // Prompt for volunteer mailing list status
       System.out.println("Enter volunteer mailing list status (1 for yes, 0
for no):");
       int mailingListStatus = inputScanner.nextInt();
       inputScanner.nextLine(); // Consume newline left-over

       // Prompt for volunteer mailing address
       System.out.println("Enter volunteer mailing address:");
       String mailingAddress = inputScanner.nextLine();

       // Prompt for volunteer phone number
       System.out.println("Enter volunteer phone number:");
       String phoneNumber = inputScanner.nextLine();

       // Prompt for volunteer email
       System.out.println("Enter volunteer email:");
       String email = inputScanner.nextLine();

       // Prompt for volunteer date joined
       System.out.println("Enter volunteer date joined (yyyy-MM-dd):");
       String dateJoined = inputScanner.nextLine();
       if (!isValidDate(dateJoined)) {
           System.err.println("Invalid date format for date joined. Please
enter the date in yyyy-MM-dd format.");
           return;
       }

       // Prompt for volunteer training date
       System.out.println("Enter volunteer training date (yyyy-MM-dd):");
```

```
        String trainingDate = inputScanner.nextLine();
        if (!isValidDate(trainingDate)) {
            System.err.println("Invalid date format for training date. Please
enter the date in yyyy-MM-dd format.");
            return;
        }

        // Prompt for volunteer training location
        System.out.println("Enter volunteer training location:");
        String trainingLocation = inputScanner.nextLine();

        // Prepare the main volunteer insertion procedure with 14 parameters
        String sql = "{call insertVolunteer(?,?,?,?,?,?,?,?,?,?,?,?)}";
        CallableStatement callableStatement = connection.prepareCall(sql);
        // Set parameters for `insertVolunteer`
        callableStatement.setInt(1, volunteerSSN); // ssn (INT)
        callableStatement.setString(2, name); // person_name (VARCHAR)
        callableStatement.setString(3, gender); // gender (VARCHAR)
        callableStatement.setString(4, profession); // profession (VARCHAR)
        callableStatement.setInt(5, mailingListStatus); // on_mailing_list
(BIT/INT)
        callableStatement.setString(6, mailingAddress); // mailing_address
(VARCHAR)
        callableStatement.setString(7, email); // email_address (VARCHAR)
        callableStatement.setString(8, phoneNumber); // phone_number (VARCHAR)
        callableStatement.setString(9, dateJoined); // date_joined (DATE as
String)
        callableStatement.setString(10, trainingDate); // training_date (DATE as
String)
        callableStatement.setString(11, trainingLocation); // training_location
(VARCHAR)

        // Execute the stored procedure
        callableStatement.execute();
        System.out.println("Volunteer inserted successfully.");
        callableStatement.close();

        // Insert emergency contacts
        String emergencyContactSql = "INSERT INTO VolunteerEmergencyContacts
(volunteer_ssn, contact_name, " +
                "contact_phone_number, relationship) VALUES (?, ?, ?, ?)";
        System.out.println("How many emergency contacts does the client have?");
        int emergencyContacts = inputScanner.nextInt();
        inputScanner.nextLine();
        for (int i = 0; i < emergencyContacts; i++) {
            System.out.println("Please enter the name of emergency contact " +
(i + 1) + ":");
            String Name = inputScanner.nextLine();
```

```
            System.out.println("Please enter the phone number of emergency
contact " + (i + 1) + ":");
            String PhoneNumber = inputScanner.nextLine();
            System.out.println("Please enter the relationship of emergency
contact " + (i + 1) + ":");
            String Relationship = inputScanner.nextLine();
            try (CallableStatement ecStmt =
connection.prepareCall(emergencyContactSql)) {
                ecStmt.setInt(1, volunteerSSN);
                ecStmt.setString(2, Name);
                ecStmt.setString(3, PhoneNumber);
                ecStmt.setString(4, Relationship);
                ecStmt.execute();
            }
        }

        // Collect team names
        List<String> teamNames = new ArrayList<>();
        System.out.println("How many teams does the volunteer serve on?");
        int numberOfTeams = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over
        for (int i = 0; i < numberOfTeams; i++) {
            System.out.println("Enter team name:");
            teamNames.add(inputScanner.nextLine());
        }

        // Insert additional team names into ServesOn table
        for (String teamName : teamNames) {
            String servesOnSql = "INSERT INTO ServesOn (ssn, team_name,
serving_month, served_hours, active) VALUES (?, ?, ?, ?, ?)";
            try (CallableStatement servesOnStmt =
connection.prepareCall(servesOnSql)) {
                servesOnStmt.setInt(1, volunteerSSN);
                servesOnStmt.setString(2, teamName);

                // Prompt for serving month
                System.out.println("Enter the number of months a volunteer has
served on a team " + teamName + ":");
                int monthsServed = inputScanner.nextInt();
                inputScanner.nextLine(); // Consume newline left-over

                // Prompt for active status
                System.out.println("Is the volunteer active for team " +
teamName + "? (1 for yes, 0 for no):");
                int active = inputScanner.nextInt();
                inputScanner.nextLine(); // Consume newline left-over
                servesOnStmt.setInt(5, active);

                for (int j = 0; j < monthsServed; ++j) {
```

```
                    // Prompt for month served
                    System.out.println("Enter the month served by volunteer on
team " + teamName + ":");
                    String servedMonth = inputScanner.nextLine();
                    servesOnStmt.setString(3, servedMonth);

                    // Prompt for hours served
                    System.out.println("Enter the number of hours served by
volunteer on team " + teamName + ":");
                    int servedHours = inputScanner.nextInt();
                    inputScanner.nextLine(); // Consume newline left-over
                    servesOnStmt.setInt(4, servedHours);

                    // Execute the insert statement
                    servesOnStmt.execute();


                }
            }
        }

        // Insert additional team leadership information if applicable
        String leadsSql = "INSERT INTO Leads (ssn, team_name, is_leader) VALUES
(?, ?, ?)";
        for (String teamName : teamNames) {
            try (CallableStatement leadsStmt = connection.prepareCall(leadsSql))
{
                leadsStmt.setInt(1, volunteerSSN);
                leadsStmt.setString(2, teamName);

                // Prompt for leadership status
                System.out.println("Is the volunteer the team leader for: '" +
teamName + "'? (1 for yes, 0 for no):");
                int isLeader = inputScanner.nextInt();
                inputScanner.nextLine(); // Consume newline left-over
                leadsStmt.setInt(3, isLeader);

                // Execute the insert statement
                leadsStmt.execute();
                System.out.println("Assigned volunteer as team leader for: '" +
teamName + "'.");
            }
        }
    }

    // Query 4
    private static void executeInsertVolunteerHoursProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
```

```
        // Prompt for volunteer SSN
        System.out.println("Enter the volunteer ssn:");
        int ssn = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prompt for team name
        System.out.println("Enter the team name:");
        String teamName = inputScanner.nextLine();

        // Prompt for serving month
        System.out.println("Enter the serving month (e.g., January):");
        String servingMonth = inputScanner.nextLine();

        // Prompt for hours worked
        System.out.println("Enter the hours worked:");
        int hoursWorked = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prompt for active status
        System.out.println("Is the volunteer currently active? (1 for yes, 0 for
no):");
        int isActive = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prepare the SQL call for the procedure
        String sql = "{call InsertVolunteerHours(?,?,?,?,?)}";
        CallableStatement callableStatement = connection.prepareCall(sql);

        // Set parameters for `insertVolunteerHours`
        callableStatement.setInt(1, ssn); // ssn (INT)
        callableStatement.setString(2, teamName); // team_name (VARCHAR)
        callableStatement.setString(3, servingMonth); // serving_month (VARCHAR)
        callableStatement.setInt(4, hoursWorked); // hours_worked (INT)
        callableStatement.setInt(5, isActive); // active (BIT/INT)

        // Execute the stored procedure
        callableStatement.execute();
        System.out.println("Volunteer hours updated successfully.");
        callableStatement.close();
    }

    // Query 5
    // Method to insert a new employee into the database
    private static void executeInsertEmployeeProcedure(Connection connection,
Scanner inputScanner) throws SQLException {
        // Prompt for employee SSN
        System.out.print("Please enter the employee's SSN: ");
        int employeeSSN = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over
```

```
        // Prompt for employee name
        System.out.print("Please enter the employee's name: ");
        String employeeName = inputScanner.nextLine();

        // Prompt for employee gender
        System.out.print("Please enter the employee's gender: ");
        String employeeGender = inputScanner.nextLine();

        // Prompt for employee profession
        System.out.print("Please enter the employee's profession: ");
        String employeeProfession = inputScanner.nextLine();

        // Prompt for employee mailing list status
        System.out.print("Is the employee on the mailing list? (1 for yes, 0 for
no): ");
        int mailingListStatus = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prompt for employee mailing address
        System.out.print("Please enter the employee's mailing address: ");
        String mailingAddress = inputScanner.nextLine();

        // Prompt for employee phone number
        System.out.print("Please enter the employee's phone number: ");
        String phoneNumber = inputScanner.nextLine();

        // Prompt for employee email address
        System.out.print("Please enter the employee's email address: ");
        String emailAddress = inputScanner.nextLine();

        // Prompt for employee salary
        System.out.print("Please enter the employee's salary: ");
        BigDecimal salary = inputScanner.nextBigDecimal();
        inputScanner.nextLine(); // Consume newline left-over

        // Prompt for employee marital status
        System.out.print("Please enter the employee's marital status: ");
        String maritalStatus = inputScanner.nextLine();

        // Prompt for employee hire date
        System.out.print("Please enter the employee's hire date (yyyy-MM-dd):
");
        String hireDate = inputScanner.nextLine();

        // Prepare the SQL call for the procedure
        String sql = "{call insertEmployee(?,?,?,?,?,?,?,?,?,?,?)}";
        CallableStatement callableStatement = connection.prepareCall(sql);
```

```
        // Set parameters for the stored procedure
        callableStatement.setInt(1, employeeSSN); // Set the SSN parameter
        callableStatement.setString(2, employeeName); // Set the name parameter
        callableStatement.setString(3, employeeGender); // Set the gender
parameter
        callableStatement.setString(4, employeeProfession); // Set the
profession parameter
        callableStatement.setInt(5, mailingListStatus); // Set the mailing list
status parameter
        callableStatement.setString(6, mailingAddress); // Set the mailing
address parameter
        callableStatement.setString(7, phoneNumber); // Set the phone number
parameter
        callableStatement.setString(8, emailAddress); // Set the email address
parameter
        callableStatement.setBigDecimal(9, salary); // Set the salary parameter
        callableStatement.setString(10, maritalStatus); // Set the marital
status parameter
        callableStatement.setString(11, hireDate); // Set the hire date
parameter

        // Execute the stored procedure
        callableStatement.execute();
        System.out.println("Employee inserted successfully.");
        callableStatement.close();

        // Insert emergency contacts
        String emergencyContactSql = "INSERT INTO EmployeeEmergencyContacts
(employee_ssn, contact_name, " +
                "contact_phone_number, relationship) VALUES (?, ?, ?, ?)";
        System.out.println("How many emergency contacts does the client have?");
        int emergencyContacts = inputScanner.nextInt();
        inputScanner.nextLine();
        for (int i = 0; i < emergencyContacts; i++) {
            System.out.println("Please enter the name of emergency contact " +
(i + 1) + ":");
            String Name = inputScanner.nextLine();
            System.out.println("Please enter the phone number of emergency
contact " + (i + 1) + ":");
            String PhoneNumber = inputScanner.nextLine();
            System.out.println("Please enter the relationship of emergency
contact " + (i + 1) + ":");
            String Relationship = inputScanner.nextLine();
            try (CallableStatement ecStmt =
connection.prepareCall(emergencyContactSql)) {
                ecStmt.setInt(1, employeeSSN);
                ecStmt.setString(2, Name);
                ecStmt.setString(3, PhoneNumber);
                ecStmt.setString(4, Relationship);
```

```
                ecStmt.execute();
            }
        }

        // Collect team names
        List<String> teamNames = new ArrayList<>();
        System.out.print("How many teams does the employee belong to? ");
        int numberOfTeams = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over
        for (int i = 0; i < numberOfTeams; i++) {
            System.out.print("Please enter the name of team " + (i + 1) + ": ");
            teamNames.add(inputScanner.nextLine());
        }

        // Insert additional team names into Reports table
        String reportsSql = "INSERT INTO Reports (ssn, team_name, report_date,
report_description) VALUES (?, ?, ?, ?)";
        for (String teamName : teamNames) {
            try (CallableStatement reportsStmt =
connection.prepareCall(reportsSql)) {
                reportsStmt.setInt(1, employeeSSN);
                reportsStmt.setString(2, teamName);

                // Prompt for report date
                System.out.print("Please enter the report date (yyyy-MM-dd) for
team " + teamName + ": ");
                String reportDate = inputScanner.nextLine();
                reportsStmt.setString(3, reportDate);

                // Prompt for report description
                System.out.print("Please enter the report description for team "
+ teamName + ": ");
                String reportDescription = inputScanner.nextLine();
                reportsStmt.setString(4, reportDescription);

                // Execute the insert statement
                reportsStmt.execute();
            }
        }
    }

    // Query 6
    // Method to insert a new expense into the database
    private static void executeInsertExpenseProcedure(Connection connection,
Scanner inputScanner) throws SQLException {
        // Prompt for employee SSN
        System.out.print("Please enter the employee's SSN: ");
        int employeeSSN = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over
```

```
        // Prompt for expense date
        System.out.print("Please enter the expense date (yyyy-MM-dd): ");
        String expenseDate = inputScanner.next();

        // Handle date format
        if (!isValidDate(expenseDate)) {
            System.err.println("Invalid date format. Please enter the date in
yyyy-MM-dd format.");
            return;
        }

        Date sqlExpenseDate = null;
        BigDecimal expenseAmount = null;

        // Prompt for expense amount
        System.out.print("Please enter the expense amount: ");
        String expenseAmountInput = inputScanner.next();

        // Handle decimal parsing
        try {
            expenseAmount = new BigDecimal(expenseAmountInput);
        } catch (NumberFormatException e) {
            System.err.println("Invalid decimal format. Please enter a valid
decimal number.");
            e.printStackTrace();
            return; // Exit the method if decimal parsing fails
        }

        // Prompt for expense description
        System.out.print("Please enter the expense description: ");
        String expenseDescription = inputScanner.next();

        // Prepare the SQL call for the procedure
        String sql = "{call insertExpense(?, ?, ?, ?)}";
        CallableStatement callableStatement = connection.prepareCall(sql);

        // Set parameters for the stored procedure
        callableStatement.setInt(1, employeeSSN); // Set the SSN parameter
        callableStatement.setString(2, expenseDate); // Set the expense date
parameter
        callableStatement.setBigDecimal(3, expenseAmount); // Set the expense
amount parameter
        callableStatement.setString(4, expenseDescription); // Set the expense
description parameter

        // Execute the stored procedure
        callableStatement.execute();
        System.out.println("Expense inserted successfully.");
```

```
        callableStatement.close();
    }

    // Query 7
    private static void executeInsertDonorProcedure(Connection connection,
Scanner inputScanner) throws SQLException {
        // Prompt for donor SSN
        System.out.print("Please enter the donor's SSN: ");
        int donorSSN = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prompt for donor name
        System.out.print("Please enter the donor's name: ");
        String donorName = inputScanner.nextLine();

        // Prompt for donor gender
        System.out.print("Please enter the donor's gender: ");
        String donorGender = inputScanner.nextLine();

        // Prompt for donor profession
        System.out.print("Please enter the donor's profession: ");
        String donorProfession = inputScanner.nextLine();

        // Prompt for donor mailing list status
        System.out.print("Is the donor on the mailing list? (1 for yes, 0 for
no): ");
        int mailingListStatus = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prompt for donor mailing address
        System.out.print("Please enter the donor's mailing address: ");
        String mailingAddress = inputScanner.nextLine();

        // Prompt for donor phone number
        System.out.print("Please enter the donor's phone number: ");
        String phoneNumber = inputScanner.nextLine();

        // Prompt for donor email
        System.out.print("Please enter the donor's email address: ");
        String emailAddress = inputScanner.nextLine();

        // Prompt for donor anonymity status
        System.out.print("Is the donor anonymous? (1 for yes, 0 for no): ");
        int isAnonymous = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prepare the SQL call for the procedure
        String sql = "{call insertDonor(?, ?, ?, ?, ?, ?, ?, ?, ?)}";
```

```
        CallableStatement callableStatement = connection.prepareCall(sql);

        // Set parameters for the stored procedure
        callableStatement.setInt(1, donorSSN); // Set the SSN parameter
        callableStatement.setString(2, donorName); // Set the name parameter
        callableStatement.setString(3, donorGender); // Set the gender parameter
        callableStatement.setString(4, donorProfession); // Set the profession
parameter
        callableStatement.setInt(5, mailingListStatus); // Set the mailing list
status parameter
        callableStatement.setString(6, mailingAddress); // Set the mailing
address parameter
        callableStatement.setString(7, emailAddress); // Set the email address
parameter
        callableStatement.setString(8, phoneNumber); // Set the phone number
parameter
        callableStatement.setInt(9, isAnonymous); // Set the anonymous status
parameter

        // Execute the stored procedure
        callableStatement.execute();
        System.out.println("Donor inserted successfully.");
        callableStatement.close();

        String emergencyContactSql = "INSERT INTO DonorEmergencyContacts
(donor_ssn, contact_name, " +
                "contact_phone_number, relationship) VALUES (?, ?, ?, ?)";
        System.out.println("How many emergency contacts does the client have?");
        int emergencyContacts = inputScanner.nextInt();
        inputScanner.nextLine();
        for (int i = 0; i < emergencyContacts; i++) {
            System.out.println("Please enter the name of emergency contact " +
(i + 1) + ":");
            String Name = inputScanner.nextLine();
            System.out.println("Please enter the phone number of emergency
contact " + (i + 1) + ":");
            String PhoneNumber = inputScanner.nextLine();
            System.out.println("Please enter the relationship of emergency
contact " + (i + 1) + ":");
            String Relationship = inputScanner.nextLine();
            try (CallableStatement ecStmt =
connection.prepareCall(emergencyContactSql)) {
                ecStmt.setInt(1, donorSSN);
                ecStmt.setString(2, Name);
                ecStmt.setString(3, PhoneNumber);
                ecStmt.setString(4, Relationship);
                ecStmt.execute();
            }
        }
```

```
        // Ask for number of donations
        System.out.print("How many donations does the donor wish to make? ");
        int numberOfDonations = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Insert additional donation information based on credit card or check
        for (int i = 0; i < numberOfDonations; i++) {
            // Prompt for donation date
            System.out.print("Please enter the donation date (yyyy-MM-dd): ");
            String donationDate = inputScanner.nextLine();

            // Prompt for donation amount
            System.out.print("Please enter the donation amount: ");
            BigDecimal donationAmount = inputScanner.nextBigDecimal();
            inputScanner.nextLine(); // Consume newline left-over

            // Prompt for donation type
            System.out.print("Please enter the donation type (credit card or
check): ");
            String donationType = inputScanner.nextLine();

            // Prompt for campaign name
            System.out.print("Please enter the campaign name: ");
            String campaignName = inputScanner.nextLine();

            if (donationType.equalsIgnoreCase("credit card")) {
                // Prompt for card number
                System.out.print("Please enter the card number: ");
                String cardNumber = inputScanner.nextLine();

                // Prompt for card type
                System.out.print("Please enter the card type: ");
                String cardType = inputScanner.nextLine();

                // Prompt for card expiration date
                System.out.print("Please enter the card expiration date
(yyyy-MM-dd): ");
                String expirationDate = inputScanner.nextLine();

                // Prepare the SQL call for the procedure
                String creditCardSql = "INSERT INTO CreditCards (ssn,
card_number, card_type, expiration_date, donation_date, amount, donation_type,
campaign_name) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
                try (CallableStatement creditCardStmt =
connection.prepareCall(creditCardSql)) {
                    creditCardStmt.setInt(1, donorSSN);
                    creditCardStmt.setString(2, cardNumber);
                    creditCardStmt.setString(3, cardType);
```

```
                    creditCardStmt.setString(4, expirationDate);
                    creditCardStmt.setString(5, donationDate);
                    creditCardStmt.setBigDecimal(6, donationAmount);
                    creditCardStmt.setString(7, donationType);
                    creditCardStmt.setString(8, campaignName);
                    creditCardStmt.execute();
                }

            } else if (donationType.equalsIgnoreCase("check")) {
                // Prompt for check number
                System.out.print("Please enter the check number: ");
                String checkNumber = inputScanner.nextLine();

                // Prepare the SQL call for the procedure
                String checkSql = "INSERT INTO DonationChecks (ssn,
check_number, donation_date, amount, donation_type, campaign_name) VALUES (?,
?, ?, ?, ?, ?)";
                try (CallableStatement checkStmt =
connection.prepareCall(checkSql)) {
                    checkStmt.setInt(1, donorSSN);
                    checkStmt.setString(2, checkNumber);
                    checkStmt.setString(3, donationDate);
                    checkStmt.setBigDecimal(4, donationAmount);
                    checkStmt.setString(5, donationType);
                    checkStmt.setString(6, campaignName);
                    checkStmt.execute();
                }

            } else {
                System.out.println("Invalid donation type. Please enter either
'credit card' or 'check'.");
            }
        }
    }

    //Query 8
    private static void executeRetrieveDoctorInfoProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
        // Prompt for client SSN
        System.out.print("Please enter the client's SSN: ");
        int clientSSN = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline left-over

        // Prepare the SQL call for the procedure
        String sql = "{call retrieveDoctorInfo(?)}";
        try (CallableStatement callableStatement = connection.prepareCall(sql))
{
            // Set the SSN parameter
            callableStatement.setInt(1, clientSSN);
```

```
            // Execute the query and process the result set
            try (ResultSet resultSet = callableStatement.executeQuery()) {
                while (resultSet.next()) {
                    // Retrieve and print doctor information
                    System.out.println("Doctor's Name: " +
resultSet.getString("doctor_name"));
                    System.out.println("Doctor's Phone Number: " +
resultSet.getString("doctor_phone_number"));
                }
            }
        }
    }

    //Query 9
    private static void executeRetrieveTotalExpensesProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
        // Prompt for start date
        System.out.print("Please enter the start date (YYYY-MM-DD): ");
        String startDate = inputScanner.next();

        // Prompt for end date
        System.out.print("Please enter the end date (YYYY-MM-DD): ");
        String endDate = inputScanner.next();

        // Prepare the SQL call for the procedure
        String sql = "{call retrieveTotalExpenses(?, ?)}";
        try (CallableStatement callableStatement = connection.prepareCall(sql))
{
            // Set the start and end date parameters
            callableStatement.setString(1, startDate);
            callableStatement.setString(2, endDate);

            // Execute the query and process the result set
            try (ResultSet resultSet = callableStatement.executeQuery()) {
                while (resultSet.next()) {
                    // Retrieve and print employee SSN and total expenses
                    System.out.println("Employee SSN: " +
resultSet.getInt("ssn"));
                    System.out.println("Total Expenses: " +
resultSet.getDouble("total_expenses"));
                }
            }
        }
    }

    // query 10
    private static void executeGetVolunteersForClientProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
```

```
        // Prompt for client SSN
        System.out.println("Enter the SSN of the client:");
        // Read the client SSN from the user
        int clientSSN = inputScanner.nextInt();
        inputScanner.nextLine();  // Consume newline left-over
        String sql = "{call GetVolunteersForClient(?)}";

        // Execute the stored procedure
        try (CallableStatement stmt = connection.prepareCall(sql)) {
            stmt.setInt(1, clientSSN);
            // Execute the query and process the result set
            try (ResultSet rs = stmt.executeQuery()) {
                boolean hasResults = false;
                System.out.println("Volunteers supporting client with SSN " +
clientSSN + ":");
                // Iterate over the result set and print the volunteer names
                while (rs.next()) {
                    hasResults = true;
                    String volunteerName = rs.getString("person_name");
                    System.out.println("- " + volunteerName);
                }

                // Print message if no volunteers found
                if (!hasResults) {
                    System.out.println("No volunteers found for this client.");
                }
            }
        } catch (SQLException e) {
            System.err.println("An error occurred while retrieving volunteer
information: " + e.getMessage());
            throw e;  // Rethrow exception if further handling is needed
        }
    }

    // Helper method to validate date format
    private static boolean isValidDate(String dateString) {
        // Define date format
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        // Set strict parsing for the date input into the console
        dateFormat.setLenient(false); // Strict date parsing
        try {
            dateFormat.parse(dateString); // Attempt to parse the date
            return true;
        } catch (ParseException e) {
            return false;
        }
    }

    // query 11
```

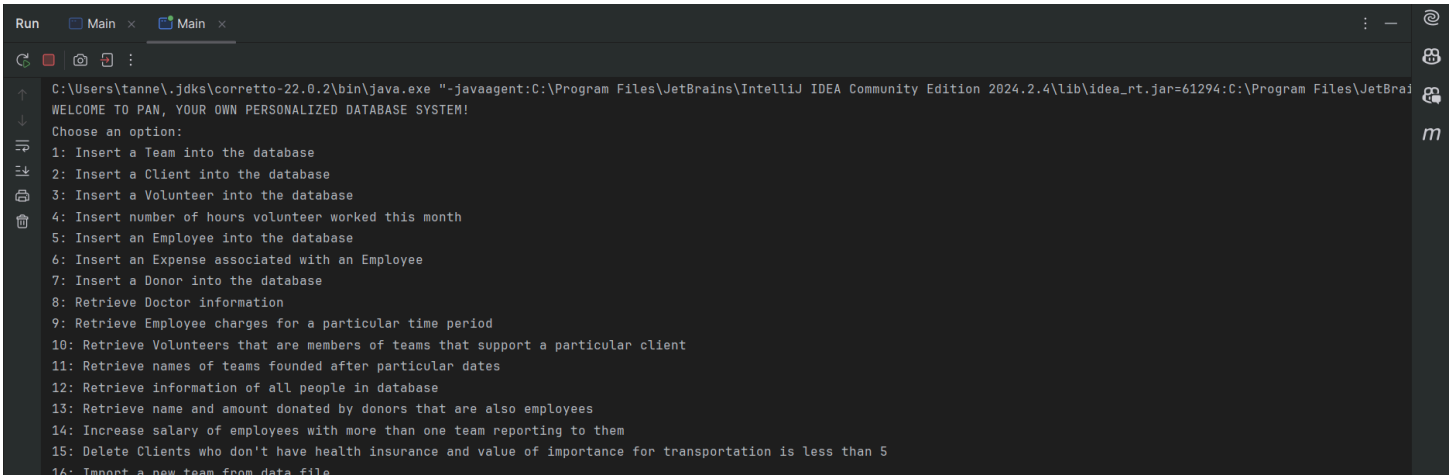
```
   private static void executeGetTeamsFoundedAfterDateProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
       // Prompt for the date to retrieve teams founded after
       System.out.println("Enter the date to retrieve teams founded after
(YYYY-MM-DD):");
       // Read the date from the user
       String dateFormed = inputScanner.next();
       // Prepare the SQL call for the procedure
       String sql = "{call getTeamsAfterDate(?)}";
       // Execute the stored procedure
       try (CallableStatement stmt = connection.prepareCall(sql)) {
           stmt.setString(1, dateFormed);
           try (ResultSet rs = stmt.executeQuery()) {
               while (rs.next()) {
                   System.out.println("Team name: " +
rs.getString("team_name"));
               }
           }
       }
   }

   // query 12
   private static void executeGetPeopleInfoProcedure(Connection connection,
Scanner inputScanner) throws SQLException {
       // Prepare the SQL call for the procedure
       String sql = "{call getAllPeopleInfo()}";
       // Execute the stored procedure
       try (CallableStatement stmt = connection.prepareCall(sql)) {
           try (ResultSet rs = stmt.executeQuery()) {
               while (rs.next()) {
                   System.out.println("Name: " + rs.getString("person_name"));
                   System.out.println("SSN: " + rs.getInt("ssn"));
                   System.out.println("Mailing Address: " +
rs.getString("mailing_address"));
                   System.out.println("Phone Number: " +
rs.getString("phone_number"));
                   System.out.println("Email Address: " +
rs.getString("email_address"));
                   System.out.println("Emergency Contact Name: " +
rs.getString("emergency_contact_name"));
                   System.out.println("Emergency Contact Phone: " +
rs.getString("emergency_contact_phone"));
                   System.out.println("Emergency Contact Relationship: " +
rs.getString("emergency_contact_relationship"));
               }
           }
       }
   }
```

```
   // query 13
   private static void executeGetDonorsThatAreEmployeesProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
       // Prepare the SQL call for the procedure
       String sql = "{call getEmployeeDonors()}";
       // Execute the stored procedure
       try (CallableStatement stmt = connection.prepareCall(sql)) {
           try (ResultSet rs = stmt.executeQuery()) {
               while (rs.next()) {
                   System.out.println("Name: " + rs.getString("person_name"));
                   System.out.println("Is the donor anonymous?: " +
rs.getString("is_anon"));
                   System.out.println("Total Donations: " +
rs.getDouble("total_donations"));
               }
           }
       }
   }

   // query 14
   private static void executeIncreaseSalaryForEmployeesProcedure(Connection
connection, Scanner inputScanner) throws SQLException {
       // Prepare the SQL call for the procedure
       String sql = "{call increaseSalaryForMultiTeamEmployees()}";
       try (CallableStatement stmt = connection.prepareCall(sql)) {
           stmt.execute();
       }
   }

   // query 15
   private static void executeDeleteClientsProcedure(Connection connection)
throws SQLException {
       // Prepare the SQL call for the procedure
       String sql = "{call deleteClients()}";
       try (CallableStatement stmt = connection.prepareCall(sql)) {
           stmt.execute();
           System.out.println("Successfully deleted uninsured clients with low
transportation importance.");
       } catch (SQLException e) {
           System.err.println("Error executing delete procedure: " +
e.getMessage());
           throw e;
       }
   }

   // query 16
   private static void executeImportTeamProcedure(Connection connection,
Scanner keyboard) throws SQLException, IOException {
       System.out.print("Enter the input file name (e.g., teams.txt): ");
```

```
        String fileName = keyboard.next();

        // Get the directory path of the current class
        String basePath = new java.io.File(".").getCanonicalPath();
        String PathToFile = basePath + java.io.File.separator + fileName;

        // Read the file line by line and insert team details
        try (BufferedReader reader = new BufferedReader(new
FileReader(PathToFile))) {
            String line;
            while ((line = reader.readLine()) != null) {
                // Parse the line to extract team details
                String[] teamData = line.split(",");
                if (teamData.length != 3) {
                    System.out.println("Invalid data format in line: " + line);
                    continue;
                }

                // Extract team details
                String teamName = teamData[0].trim();
                String teamType = teamData[1].trim();
                Date dateOfCreation = Date.valueOf(teamData[2].trim()); //
Assumes format yyyy-MM-dd

                // Call the InsertTeam stored procedure
                String query = "{CALL InsertTeam(?, ?, ?)}";
                try (CallableStatement callableStatement =
connection.prepareCall(query)) {
                    callableStatement.setString(1, teamName);
                    callableStatement.setString(2, teamType);
                    callableStatement.setDate(3, dateOfCreation);

                    // Execute the stored procedure
                    callableStatement.executeUpdate();
                    System.out.println("Inserted team: " + teamName);
                } catch (SQLException e) {
                    System.out.println("Error inserting team '" + teamName + "':
" + e.getMessage());
                }
            }
            // Close the reader
            System.out.println("Team import completed from file: " +
PathToFile);
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + PathToFile);
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        }
    }
```

```
   // Query 17: Export: Retrieve names and mailing addresses of all people on
the mailing list and output them to a data file instead of screen (the user
must be asked to enter the output file name).
   private static void executeExportMailingList(Connection connection, Scanner
keyboard) throws SQLException, IOException {
       System.out.print("Enter the output file name (e.g., output.txt): ");
       String nameOfFile = keyboard.next();

       // Get the directory path of the current class
       String base = new java.io.File(".").getCanonicalPath();
       String file = base + java.io.File.separator + nameOfFile;

       // Prepare the SQL call for the procedure
       String query = "{CALL executeExportMailingList()}";
       try (CallableStatement callableStatement =
connection.prepareCall(query);
            ResultSet resultSet = callableStatement.executeQuery()) {

           try (FileWriter fWriter = new FileWriter(file)) {
               while (resultSet.next()) {
                   String name = resultSet.getString("person_name");
                   String address = resultSet.getString("mailing_address");

                   fWriter.write("Name: " + name + ", Address: " + address +
"\n");
               }
               System.out.println("Data successfully exported to " + file);
           } catch (IOException e) {
               System.out.println("Error writing to file: " + e.getMessage());
           }
       } catch (SQLException e) {
           System.out.println("Error executing stored procedure: " +
e.getMessage());
       }
   }
}
```

Successful Compilation of Java Code:



Task 6:

Task 6.1:

```
WELCOME TO PAN, YOUR OWN PERSONALIZED DATABASE SYSTEM!
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
1
Enter the Team Name:
team1
Enter the Team Type:
Leadership
Enter the date formed (yyyy-MM-dd):
2020-01-01
Team inserted successfully.
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
1
Enter the Team Name:
team2
Enter the Team Type:
functional
Enter the date formed (yyyy-MM-dd):
2020-02-01
Team inserted successfully.
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
1
Enter the Team Name:
team3
Enter the Team Type:
cross-functional
Enter the date formed (yyyy-MM-dd):
2020-03-01
Team inserted successfully.
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
1
Enter the Team Name:
team4
Enter the Team Type:
virtual
Enter the date formed (yyyy-MM-dd):
2020-04-01
Team inserted successfully.
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
1
Enter the Team Name:
team5
Enter the Team Type:
project
Enter the date formed (yyyy-MM-dd):
2020-05-01
Team inserted successfully.
```

Showing Team Table after 5 Queries:

```
45    SELECT * FROM Teams;
```

Results | Messages

	team_name	team_type	date_formed
1	team1	Leadership	2020-01-01
2	team2	functional	2020-02-01
3	team3	cross-functional	2020-03-01
4	team4	virtual	2020-04-01
5	team5	project	2020-05-01

Task 6.2:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
2
Enter the client ssn:
1
Enter the client name:
Tanner
Enter the client gender:
male
```

```
Enter the client profession:
software developer
Is Client on the mailing list? (1 for yes, 0 for no):
1
Enter the client mailing address:
Norman, OK
Enter the client phone number:
999-9999-9999
Enter the client email:
tanner@email.com
Enter the client assignment date (yyyy-MM-dd):
2020-01-01
Enter the doctor name of client:
jim
Enter the client doctor's phone number:
999-9988-8888
How many teams care for the client?
1
Enter the team 1 name:
team1
Enter the client's need type:
Transportation
Enter the client's need importance (1-10):
3
Enter the client's insurance policy id:
1
Enter the client's insurance provider:
state farm
```

```
Enter the client's insurance provider's address:
insurance address 1
Enter client's insurance type:
Life
Client has been inserted successfully.
How many emergency contacts does the client have?
1
Please enter the name of emergency contact 1:
heather
Please enter the phone number of emergency contact 1:
999-9898-9999
Please enter the relationship of emergency contact 1:
mom
Is client active for team team1? (1 for yes, 0 for no):
1
```

After 5 Client Queries:

Showing Clients Table:

```
44  -- SELECT * FROM Reports;
45  SELECT * FROM Clients;
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_nam
1	1	Tanner	male	software developer	1	Norman, OK	999-9999-9999	tanner@email.com	2020-01-01	jim
2	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan
3	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy
4	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm
5	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben

Showing Needs Table:

```
45    SELECT * FROM Needs;
```

Results Messages

	ssn	need_type	importance
1	1	Transportation	3
2	2	Transportation	6
3	3	Transportation	6
4	4	Transportation	3
5	5	Transportation	8

Showing Has Table:

```
44
45    SELECT * FROM Has;
```

Results Messages

	ssn	policy_id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

Showing InsurancePolicies:

```
44
45  SELECT * FROM InsurancePolicies;
```

Results | Messages

	policy_id	provider_name	provider_address	insurance_type
1	1	state farm	insurance address 1	Life
2	2	state farm	norman, ok	Life
3	3	state farm	norman, ok	Health
4	4	state farm	norman, ok	Health
5	5	state farm	norman, ok	Home and auto

Showing ClientEmergencyContacts:

```
45  SELECT * FROM ClientEmergencyContacts;
```

Results | Messages

	client_ssn	contact_name	contact_phone_number	relationship
1	1	heather	999-9898-9999	mom
2	2	jimmy hendrix	987-999-9999	dad
3	3	Travis scott	984-999-9999	dad
4	4	Future	981-999-9999	dad
5	5	Donald J. Trump	978-999-9999	dad

Showing CaresFor:

```
44
45  SELECT * FROM CaresFor;
```

Results Messages

	ssn	team_name	client_active
1	1	team1	1
2	2	team2	1
3	3	team3	1
4	4	team4	1
5	5	team5	1

Task 6.3:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
3
Enter volunteer ssn:
10
Enter volunteer name:
jimothy
Enter volunteer gender:
male
```

```
Enter volunteer profession:
business
Enter volunteer mailing list status (1 for yes, 0 for no):
1
Enter volunteer mailing address:
norman, ok
Enter volunteer phone number:
968-999-9999
Enter volunteer email:
jimothy@email.com
Enter volunteer date joined (yyyy-MM-dd):
2020-02-05
Enter volunteer training date (yyyy-MM-dd):
2020-02-05
Enter volunteer training location:
norman
Volunteer inserted successfully.
How many emergency contacts does the client have?
1
Please enter the name of emergency contact 1:
braden
Please enter the phone number of emergency contact 1:
967-999-9999
Please enter the relationship of emergency contact 1:
dad
How many teams does the volunteer serve on?
1
```

```
Enter team name:
team5
Enter the number of months a volunteer has served on a team team5:
1
Is the volunteer active for team team5? (1 for yes, 0 for no):
1
Enter the month served by volunteer on team team5:
may
Enter the number of hours served by volunteer on team team5:
40
Is the volunteer the team leader for: 'team5'? (1 for yes, 0 for no):
1
Assigned volunteer as team leader for: 'team5'.
```

After 5 Volunteers Entries:

Showing Volunteers Table:

```
SELECT * FROM Volunteers;
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	date_joined	training_date
1	6	ben stiller	male	business	1	norman, ok	977-999-9999	benstiller@email.com	2020-02-01	2020-02-01
2	7	jack	male	business	1	norman, ok	975-999-9999		2020-02-02	2020-02-02
3	8	doherty	male	business	1	norman, ok	973-999-9999	doherty@email.com	2020-02-03	2020-02-03
4	9	yuji	male	business	1	norman, ok	970-999-9999	yuji@email.com	2020-02-04	2020-02-04
5	10	jimothy	male	business	1	norman, ok	968-999-9999	jimothy@email.com	2020-02-05	2020-02-05

Showing VolunteerEmergencyContacts Table:

```
7
8   SELECT * FROM VolunteerEmergencyContacts;
9
```

Results | Messages

	volunteer_ssn	contact_name	contact_phone_number	relationship
1	6	sydney sweeney	976-999-9999	mom
2	7	tanner	974-999-9999	dad
3	8	tanner	972-999-9999	dad
4	9	braydon	969-999-9999	dad
5	10	braden	967-999-9999	dad

Showing ServesOn Table:

```
8   SELECT * FROM ServesOn;
9
```

Results | Messages

	ssn	team_name	serving_month	served_hours	active
1	6	team1	january	40	1
2	7	team2	february	40	1
3	8	team3	march	40	1
4	9	team4	april	40	1
5	10	team5	may	40	1

Showing Leads Table:

```
7
8   SELECT * FROM Leads;
9
```

Results Messages

	ssn	team_name	is_leader
1	6	team1	1
2	7	team2	1
3	8	team3	1
4	9	team4	0
5	10	team5	1

Task 6.4:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
4
```

```
Enter the volunteer ssn:
6
Enter the team name:
team1
Enter the serving month (e.g., January):
january
Enter the hours worked:
30
Is the volunteer currently active? (1 for yes, 0 for no):
1
Volunteer hours updated successfully.
```

After 5 Insertions:

Showing ServesOn Table:

```
8   SELECT * FROM ServesOn;
9
```

Results Messages

	ssn	team_name	serving_month	served_hours	active
1	6	team1	january	30	1
2	7	team2	february	30	1
3	8	team3	march	30	1
4	9	team4	april	30	1
5	10	team5	may	30	1

Task 6.5:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
5
```

```
Please enter the employee's SSN: 15
Please enter the employee's name: mountie
Please enter the employee's gender: male
Please enter the employee's profession: business
Is the employee on the mailing list? (1 for yes, 0 for no): 1
Please enter the employee's mailing address: norman, ok
Please enter the employee's phone number: 951-999-9999
Please enter the employee's email address: mountie@email.com
Please enter the employee's salary: 99999
Please enter the employee's marital status: married
Please enter the employee's hire date (yyyy-MM-dd): 2020-03-05
Employee inserted successfully.
How many emergency contacts does the client have?
1
Please enter the name of emergency contact 1:
cole
Please enter the phone number of emergency contact 1:
950-999-9999
Please enter the relationship of emergency contact 1:
dad
How many teams does the employee belong to? 1
Please enter the name of team 1: team5
Please enter the report date (yyyy-MM-dd) for team team5: 2020-03-05
Please enter the report description for team team5: description
```

After 5 Insertions:

Showing Employees Table:

```
9   SELECT * FROM Employees;
10
11
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	salary	marital_status	hi
1	11	tucker	male	business	1	norman, ok	960-999-9999	tucker@email.com	99999.00	married	20
2	12	Braden	male	business	1	norman, ok	958-999-9999	Braden@email.com	99999.00	married	20
3	13	tanner	male	business	1	norman, ok	955-999-9999	tanner@email.com	99999.00	married	20
4	14	cole	male	business	1	norman, ok	953-999-9999	cole@email.com	99999.00	married	20
5	15	mountie	male	business	1	norman, ok	951-999-9999	mountie@email.com	99999.00	married	20

Showing EmployeeEmergencyContacts Table:

```
5
6
7   SELECT * FROM EmployeeEmergencyContacts;
8
```

Results | Messages

	employee_ssn	contact_name	contact_phone_number	relationship
1	11	tanner	959-999-9999	dad
2	12	tanner	957-999-9999	dad
3	13	tucker	954-999-9999	dad
4	14	tanner	952-999-9999	dad
5	15	cole	950-999-9999	dad

Showing Reports Table:

```
6
7   SELECT * FROM Reports;
8
```

Results Messages

	ssn	team_name	report_date	report_description
1	11	team1	2020-03-01	description
2	12	team2	2020-03-02	description
3	13	team3	2020-03-03	description
4	14	team4	2020-03-04	description
5	15	team5	2020-03-05	description

Task 6.6:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
6
Please enter the employee's SSN: 11
Please enter the expense date (yyyy-MM-dd): 2020-03-01
Please enter the expense amount: 100
Please enter the expense description: description
Expense inserted successfully.
```

After 5 Insertions:

Showing Expenses Table:

```
6
7   SELECT * FROM Expenses;
8
```

Results | Messages

	ssn	expense_date	amount	expense_description
1	11	2020-03-01	100.00	description
2	12	2020-03-02	100.00	description
3	13	2020-03-03	100.00	description
4	14	2020-03-04	100.00	description
5	15	2020-03-05	100.00	description

Task 6.7:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
7
```

```
Please enter the donor's SSN: 11
Please enter the donor's name: tucker
Please enter the donor's gender: male
Please enter the donor's profession: business
Is the donor on the mailing list? (1 for yes, 0 for no): 1
Please enter the donor's mailing address: norman, ok
Please enter the donor's phone number: 960-999-9999
Please enter the donor's email address: tucker@email.com
Is the donor anonymous? (1 for yes, 0 for no): 1
Donor inserted successfully.
How many emergency contacts does the client have?
1
Please enter the name of emergency contact 1:
tanner
Please enter the phone number of emergency contact 1:
949-999-9999
Please enter the relationship of emergency contact 1:
dad
How many donations does the donor wish to make? 1
Please enter the donation date (yyyy-MM-dd): 2020-04-05
Please enter the donation amount: 100
Please enter the donation type (credit card or check): check
Please enter the campaign name: campaign
Please enter the check number: 55555
```

After 5 Insertions:

Showing Donors Table:

```
7   SELECT * FROM Donors;
8
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	is_anon
1	11	tucker	male	business	1	norman, ok	tucker@email.com	960-999-9999	1
2	16	tanner	male	business	1	norman, ok	tanner@email.com	940-999-9999	1
3	17	mountie	male	business	1	norman, ok	mountie@email.com	943-999-9999	1
4	18	braydon	male	business	1	norman, ok	braydon@email.com	945-999-9999	1
5	19	emily	female	business	1	norman, ok	emily@email.com	947-999-9999	1

Showing CreditCards Table:

```
7  SELECT * FROM CreditCards;
8
```

Results | Messages

	ssn	card_number	card_type	expiration_date	donation_date	amount	donation_type	campaign_name
1	17	22222	visa	2020-04-02	2020-04-02	100.00	credit card	campaign

Showing DonationChecks Table:

```
6
7  SELECT * FROM DonationChecks;
8
```

Results | Messages

	ssn	check_number	donation_date	amount	donation_type	campaign_name
1	11	55555	2020-04-05	100.00	check	campaign
2	16	11111	2020-04-01	100.00	check	campaign
3	18	33333	2020-04-03	100.00	check	campaign
4	19	44444	2020-04-04	100.00	check	campaign

Task 6.8:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
8
Please enter the client's SSN: 1
Doctor's Name: jim
Doctor's Phone Number: 999-9988-8888
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
8
Please enter the client's SSN: 2
Doctor's Name: jonathan
Doctor's Phone Number: 988-999-9999
```

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_name	doctor_phone_number
1	1	Tanner	male	software developer	1	Norman, OK	999-9999-9999	tanner@email.com	2020-01-01	jim	999-9988-8888
2	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan	988-999-9999
3	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy	985-999-9999
4	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm	982-999-9999
5	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben	979-999-9999

Task 6.9:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
9
```

```
Please enter the start date (YYYY-MM-DD): 2020-03-01
Please enter the end date (YYYY-MM-DD): 2020-03-02
Employee SSN: 11
Total Expenses: 100.0
Employee SSN: 12
Total Expenses: 100.0
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
9
```

```
Please enter the start date (YYYY-MM-DD): 2020-03-03
Please enter the end date (YYYY-MM-DD): 2020-03-04
Employee SSN: 13
Total Expenses: 200.0
Employee SSN: 14
Total Expenses: 100.0
```

```
7    SELECT * FROM Expenses;
8
```

Results | Messages

	ssn	expense_date	amount	expense_description
1	11	2020-03-01	100.00	description
2	12	2020-03-02	100.00	description
3	13	2020-03-03	100.00	description
4	13	2020-03-04	100.00	description
5	14	2020-03-04	100.00	description
6	15	2020-03-05	100.00	description

Task 6.10:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
10
```

```
Enter the SSN of the client:
1
Volunteers supporting client with SSN 1:
- ben stiller
```

```
Enter the SSN of the client:
2
Volunteers supporting client with SSN 2:
- jack
```

```
SELECT * FROM Clients;
SELECT * FROM CaresFor;
SELECT * FROM Teams;
SELECT * FROM Reports;
SELECT * FROM Volunteers;
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_name	doctor_phone_number
1	1	Tanner	male	software developer	1	Norman, OK	999-9999-9999	tanner@email.com	2020-01-01	jim	999-9988-8888
2	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan	988-999-9999
3	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy	985-999-9999
4	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm	982-999-9999
5	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben	979-999-9999

	ssn	team_name	client_active
1	1	team1	1
2	2	team2	1
3	3	team3	1
4	4	team4	1
5	5	team5	1

	team_name	team_type	date_formed
1	team1	Leadership	2020-01-01
2	team2	functional	2020-02-01
3	team3	cross-functional	2020-03-01
4	team4	virtual	2020-04-01
5	team5	project	2020-05-01

	ssn	team_name	report_date	report_description
1	11	team1	2020-03-01	description
2	12	team2	2020-03-02	description
3	13	team3	2020-03-03	description
4	14	team4	2020-03-04	description
5	15	team5	2020-03-05	description

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	date_joined	training_date	training_location
1	6	ben stiller	male	business	1	norman, ok	977-999-9999	benstiller@email.com	2020-02-01	2020-02-01	norman
2	7	jack	male	business	1	norman, ok	975-999-9999		2020-02-02	2020-02-02	norman
3	8	doherty	male	business	1	norman, ok	973-999-9999	doherty@email.com	2020-02-03	2020-02-03	norman
4	9	yuji	male	business	1	norman, ok	970-999-9999	yuji@email.com	2020-02-04	2020-02-04	norman
5	10	jimothy	male	business	1	norman, ok	968-999-9999	jimothy@email.com	2020-02-05	2020-02-05	norman

Task 6.11:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
11
```

```
Enter the date to retrieve teams founded after (YYYY-MM-DD):
2020-03-01
Team name: team4
Team name: team5
```

```
Enter the date to retrieve teams founded after (YYYY-MM-DD):
2020-01-01
Team name: team2
Team name: team3
Team name: team4
Team name: team5
```

```
2
3   SELECT * FROM Teams;
4
```

Results Messages

	team_name	team_type	date_formed
1	team1	Leadership	2020-01-01
2	team2	functional	2020-02-01
3	team3	cross-functional	2020-03-01
4	team4	virtual	2020-04-01
5	team5	project	2020-05-01

Task 6.12:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
12
```

```
Name: tucker
SSN: 11
Mailing Address: norman, ok
Phone Number: 960-999-9999
Email Address: tucker@email.com
Emergency Contact Name: tanner
Emergency Contact Phone: 959-999-9999
Emergency Contact Relationship: dad
Name: Braden
SSN: 12
Mailing Address: norman, ok
Phone Number: 958-999-9999
Email Address: Braden@email.com
Emergency Contact Name: tanner
Emergency Contact Phone: 957-999-9999
Emergency Contact Relationship: dad
Name: tanner
SSN: 13
Mailing Address: norman, ok
Phone Number: 955-999-9999
Email Address: tanner@email.com
Emergency Contact Name: tucker
Emergency Contact Phone: 954-999-9999
Emergency Contact Relationship: dad
Name: cole
```

```
SSN: 14
Mailing Address: norman, ok
Phone Number: 953-999-9999
Email Address: cole@email.com
Emergency Contact Name: tanner
Emergency Contact Phone: 952-999-9999
Emergency Contact Relationship: dad
Name: mountie
SSN: 15
Mailing Address: norman, ok
Phone Number: 951-999-9999
Email Address: mountie@email.com
Emergency Contact Name: cole
Emergency Contact Phone: 950-999-9999
Emergency Contact Relationship: dad
Name: ben stiller
SSN: 6
Mailing Address: norman, ok
Phone Number: 977-999-9999
Email Address: benstiller@email.com
Emergency Contact Name: sydney sweeney
Emergency Contact Phone: 976-999-9999
Emergency Contact Relationship: mom
Name: jack
```

```
SSN: 7
Mailing Address: norman, ok
Phone Number: 975-999-9999
Email Address:
Emergency Contact Name: tanner
Emergency Contact Phone: 974-999-9999
Emergency Contact Relationship: dad
Name: doherty
SSN: 8
Mailing Address: norman, ok
Phone Number: 973-999-9999
Email Address: doherty@email.com
Emergency Contact Name: tanner
Emergency Contact Phone: 972-999-9999
Emergency Contact Relationship: dad
Name: yuji
SSN: 9
Mailing Address: norman, ok
Phone Number: 970-999-9999
Email Address: yuji@email.com
Emergency Contact Name: braydon
Emergency Contact Phone: 969-999-9999
Emergency Contact Relationship: dad
Name: jimothy
```

```
SSN: 10
Mailing Address: norman, ok
Phone Number: 968-999-9999
Email Address: jimothy@email.com
Emergency Contact Name: braden
Emergency Contact Phone: 967-999-9999
Emergency Contact Relationship: dad
Name: Tanner
SSN: 1
Mailing Address: Norman, OK
Phone Number: 999-9999-9999
Email Address: tanner@email.com
Emergency Contact Name: heather
Emergency Contact Phone: 999-9898-9999
Emergency Contact Relationship: mom
Name: Braydon
SSN: 2
Mailing Address: norman, ok
Phone Number: 989-9899-9898
Email Address: braydon@email.com
Emergency Contact Name: jimmy hendrix
Emergency Contact Phone: 987-999-9999
Emergency Contact Relationship: dad
Name: jackson
```

```
SSN: 3
Mailing Address: norman, ok
Phone Number: 986-999-9999
Email Address: jackson@email.com
Emergency Contact Name: Travis scott
Emergency Contact Phone: 984-999-9999
Emergency Contact Relationship: dad
Name: Braden
SSN: 4
Mailing Address: norman, ok
Phone Number: 983-999-9999
Email Address: braden@email.com
Emergency Contact Name: Future
Emergency Contact Phone: 981-999-9999
Emergency Contact Relationship: dad
Name: Tucker
SSN: 5
Mailing Address: norman, ok
Phone Number: 980-999-9999
Email Address: tucker@email.com
Emergency Contact Name: Donald J. Trump
Emergency Contact Phone: 978-999-9999
Emergency Contact Relationship: dad
Name: tucker
```

```
SSN: 11
Mailing Address: norman, ok
Phone Number: tucker@email.com
Email Address: 960-999-9999
Emergency Contact Name: tanner
Emergency Contact Phone: 949-999-9999
Emergency Contact Relationship: dad
Name: tanner
SSN: 16
Mailing Address: norman, ok
Phone Number: tanner@email.com
Email Address: 940-999-9999
Emergency Contact Name: Braden
Emergency Contact Phone: 941-999-9999
Emergency Contact Relationship: dad
Name: mountie
SSN: 17
Mailing Address: norman, ok
Phone Number: mountie@email.com
Email Address: 943-999-9999
Emergency Contact Name: tanner
Emergency Contact Phone: 944-999-9999
Emergency Contact Relationship: dad
Name: braydon
```

```
SSN: 18
Mailing Address: norman, ok
Phone Number: braydon@email.com
Email Address: 945-999-9999
Emergency Contact Name: tanner
Emergency Contact Phone: 946-999-9999
Emergency Contact Relationship: dad
Name: emily
SSN: 19
Mailing Address: norman, ok
Phone Number: emily@email.com
Email Address: 947-999-9999
Emergency Contact Name: tanner
Emergency Contact Phone: 948-999-9999
Emergency Contact Relationship: dad
```

```
SELECT * FROM Clients;
SELECT * FROM Volunteers;
SELECT * FROM Employees;
SELECT * FROM Donors;
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_name	doctor_phone_number
1	1	Tanner	male	software developer	1	Norman, OK	999-9999-9999	tanner@email.com	2020-01-01	jim	999-9988-8888
2	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan	988-999-9999
3	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy	985-999-9999
4	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm	982-999-9999
5	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben	979-999-9999

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	date_joined	training_date	training_location
1	6	ben stiller	male	business	1	norman, ok	977-999-9999	benstiller@email.com	2020-02-01	2020-02-01	norman
2	7	jack	male	business	1	norman, ok	975-999-9999		2020-02-02	2020-02-02	norman
3	8	doherty	male	business	1	norman, ok	973-999-9999	doherty@email.com	2020-02-03	2020-02-03	norman
4	9	yuji	male	business	1	norman, ok	970-999-9999	yuji@email.com	2020-02-04	2020-02-04	norman
5	10	jimothy	male	business	1	norman, ok	968-999-9999	jimothy@email.com	2020-02-05	2020-02-05	norman

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	salary	marital_status	hire_date
1	11	tucker	male	business	1	norman, ok	960-999-9999	tucker@email.com	99999.00	married	2020-03-01
2	12	Braden	male	business	1	norman, ok	958-999-9999	Braden@email.com	99999.00	married	2020-03-02
3	13	tanner	male	business	1	norman, ok	955-999-9999	tanner@email.com	99999.00	married	2020-03-03
4	14	cole	male	business	1	norman, ok	953-999-9999	cole@email.com	99999.00	married	2020-03-04
5	15	mountie	male	business	1	norman, ok	951-999-9999	mountie@email.com	99999.00	married	2020-03-05

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	is_anon
1	11	tucker	male	business	1	norman, ok	tucker@email.com	960-999-9999	1
2	16	tanner	male	business	1	norman, ok	tanner@email.com	940-999-9999	1
3	17	mountie	male	business	1	norman, ok	mountie@email.com	943-999-9999	1
4	18	braydon	male	business	1	norman, ok	braydon@email.com	945-999-9999	1
5	19	emily	female	business	1	norman, ok	emily@email.com	947-999-9999	1

Task 6.13:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
13
Name: tucker
Total Donations: 100.0
```

```
3  SELECT * FROM Employees;
4  SELECT * FROM Donors;
5
6
7
```

Results | Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	salary	marital_status	hire_date
1	11	tucker	male	business	1	norman, ok	960-999-9999	tucker@email.com	99999.00	married	2020-03-01
2	12	Braden	male	business	1	norman, ok	958-999-9999	Braden@email.com	99999.00	married	2020-03-02
3	13	tanner	male	business	1	norman, ok	955-999-9999	tanner@email.com	99999.00	married	2020-03-03
4	14	cole	male	business	1	norman, ok	953-999-9999	cole@email.com	99999.00	married	2020-03-04
5	15	mountie	male	business	1	norman, ok	951-999-9999	mountie@email.com	99999.00	married	2020-03-05

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	is_anon
1	11	tucker	male	business	1	norman, ok	tucker@email.com	960-999-9999	1
2	16	tanner	male	business	1	norman, ok	tanner@email.com	940-999-9999	1
3	17	mountie	male	business	1	norman, ok	mountie@email.com	943-999-9999	1
4	18	braydon	male	business	1	norman, ok	braydon@email.com	945-999-9999	1
5	19	emily	female	business	1	norman, ok	emily@email.com	947-999-9999	1

Task 6.14:

```
SELECT * FROM Employees;
SELECT * FROM Reports;
SELECT * FROM Teams;
```

Results Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	salary	marital_status	hire_date
1	11	tucker	male	business	1	norman, ok	960-999-9999	tucker@email.com	99999.00	married	2020-03-01
2	12	Braden	male	business	1	norman, ok	958-999-9999	Braden@email.com	99999.00	married	2020-03-02
3	13	tanner	male	business	1	norman, ok	955-999-9999	tanner@email.com	99999.00	married	2020-03-03
4	14	cole	male	business	1	norman, ok	953-999-9999	cole@email.com	99999.00	married	2020-03-04
5	15	mountie	male	business	1	norman, ok	951-999-9999	mountie@email.com	99999.00	married	2020-03-05
6	20	tanner	male	business	1	norman, ok	888-888-8888	tanner@email.com	100000.00	married	2020-06-01

	ssn	team_name	report_date	report_description
1	11	team1	2020-03-01	description
2	12	team2	2020-03-02	description
3	13	team3	2020-03-03	description
4	14	team4	2020-03-04	description
5	15	team5	2020-03-05	description
6	20	team1	2020-06-01	description
7	20	team2	2020-06-01	description

	team_name	team_type	date_formed
1	team1	Leadership	2020-01-01
2	team2	functional	2020-02-01
3	team3	cross-functional	2020-03-01
4	team4	virtual	2020-04-01
5	team5	project	2020-05-01

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
14
```

```
SELECT * FROM Employees;
SELECT * FROM Reports;
SELECT * FROM Teams;
```

Results Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	salary	marital_status	hire_date
1	11	tucker	male	business	1	norman, ok	960-999-9999	tucker@email.com	99999.00	married	2020-03-01
2	12	Braden	male	business	1	norman, ok	958-999-9999	Braden@email.com	99999.00	married	2020-03-02
3	13	tanner	male	business	1	norman, ok	955-999-9999	tanner@email.com	99999.00	married	2020-03-03
4	14	cole	male	business	1	norman, ok	953-999-9999	cole@email.com	99999.00	married	2020-03-04
5	15	mountie	male	business	1	norman, ok	951-999-9999	mountie@email.com	99999.00	married	2020-03-05
6	20	tanner	male	business	1	norman, ok	888-888-8888	tanner@email.com	110000.00	married	2020-06-01

Task 6.15:

```
SELECT * FROM Clients;
SELECT * FROM Needs;
SELECT * FROM ClientEmergencyContacts;
SELECT * FROM Has;
SELECT * FROM InsurancePolicies;
SELECT * FROM CaresFor;
```

Results Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_name	doctor_phone_number
1	1	Tanner	male	software developer	1	Norman, OK	999-9999-9999	tanner@email.com	2020-01-01	jim	999-9988-8888
2	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan	988-999-9999
3	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy	985-999-9999
4	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm	982-999-9999
5	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben	979-999-9999

	ssn	need_type	importance
1	1	Transportation	3
2	2	Transportation	6
3	3	Transportation	6
4	4	Transportation	3
5	5	Transportation	8

	client_ssn	contact_name	contact_phone_number	relationship
1	1	heather	999-9898-9999	mom
2	2	jimmy hendrix	987-999-9999	dad
3	3	Travis scott	984-999-9999	dad
4	4	Future	981-999-9999	dad
5	5	Donald J. Trump	978-999-9999	dad

	ssn	policy_id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

	policy_id	provider_name	provider_address	insurance_type
1	1	state farm	insurance address 1	Life
2	2	state farm	norman, ok	Life
3	3	state farm	norman, ok	Health
4	4	state farm	norman, ok	Health
5	5	state farm	norman, ok	Home and auto

	ssn	team_name	client_active
1	1	team1	1
2	2	team2	1
3	3	team3	1
4	4	team4	1
5	5	team5	1

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
15
Successfully deleted uninsured clients with low transportation importance.
```

```
SELECT * FROM Clients;
SELECT * FROM Needs;
SELECT * FROM ClientEmergencyContacts;
SELECT * FROM Has;
SELECT * FROM InsurancePolicies;
SELECT * FROM CaresFor;
```

Results Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_name	doctor_phone_number
1	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan	988-999-9999
2	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy	985-999-9999
3	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm	982-999-9999
4	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben	979-999-9999

	ssn	need_type	importance
1	2	Transportation	6
2	3	Transportation	6
3	4	Transportation	3
4	5	Transportation	8

	client_ssn	contact_name	contact_phone_number	relationship
1	2	jimmy hendrix	987-999-9999	dad
2	3	Travis scott	984-999-9999	dad
3	4	Future	981-999-9999	dad
4	5	Donald J. Trump	978-999-9999	dad

	ssn	policy_id
1	2	2
2	3	3
3	4	4
4	5	5

	policy_id	provider_name	provider_address	insurance_type
1	2	state farm	norman, ok	Life
2	3	state farm	norman, ok	Health
3	4	state farm	norman, ok	Health
4	5	state farm	norman, ok	Home and auto

	ssn	team_name	client_active
1	2	team2	1
2	3	team3	1
3	4	team4	1
4	5	team5	1

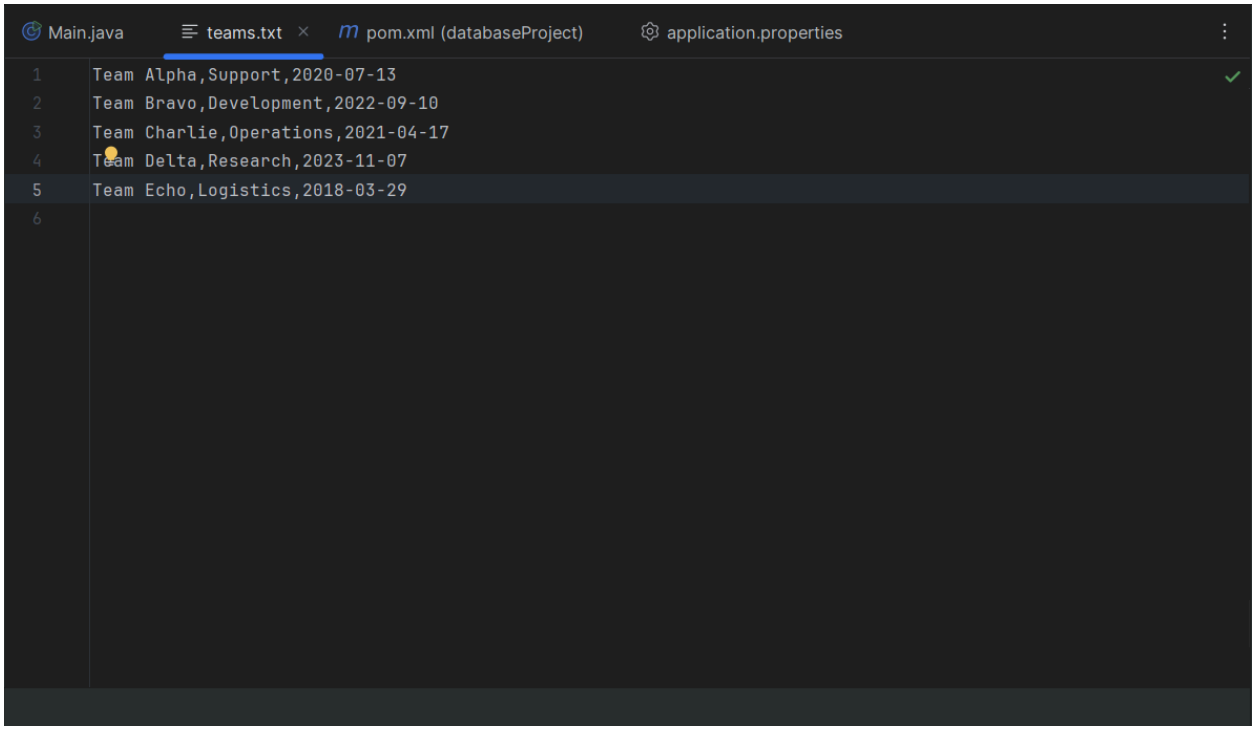
Task 6.16:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
16
Enter the input file name (e.g., teams.txt): teams.txt
Inserted team: Team Alpha
Inserted team: Team Bravo
Inserted team: Team Charlie
Inserted team: Team Delta
Inserted team: Team Echo
Team import completed from file: C:\Users\tanne\IdeaProjects\databaseProject\teams.txt
```

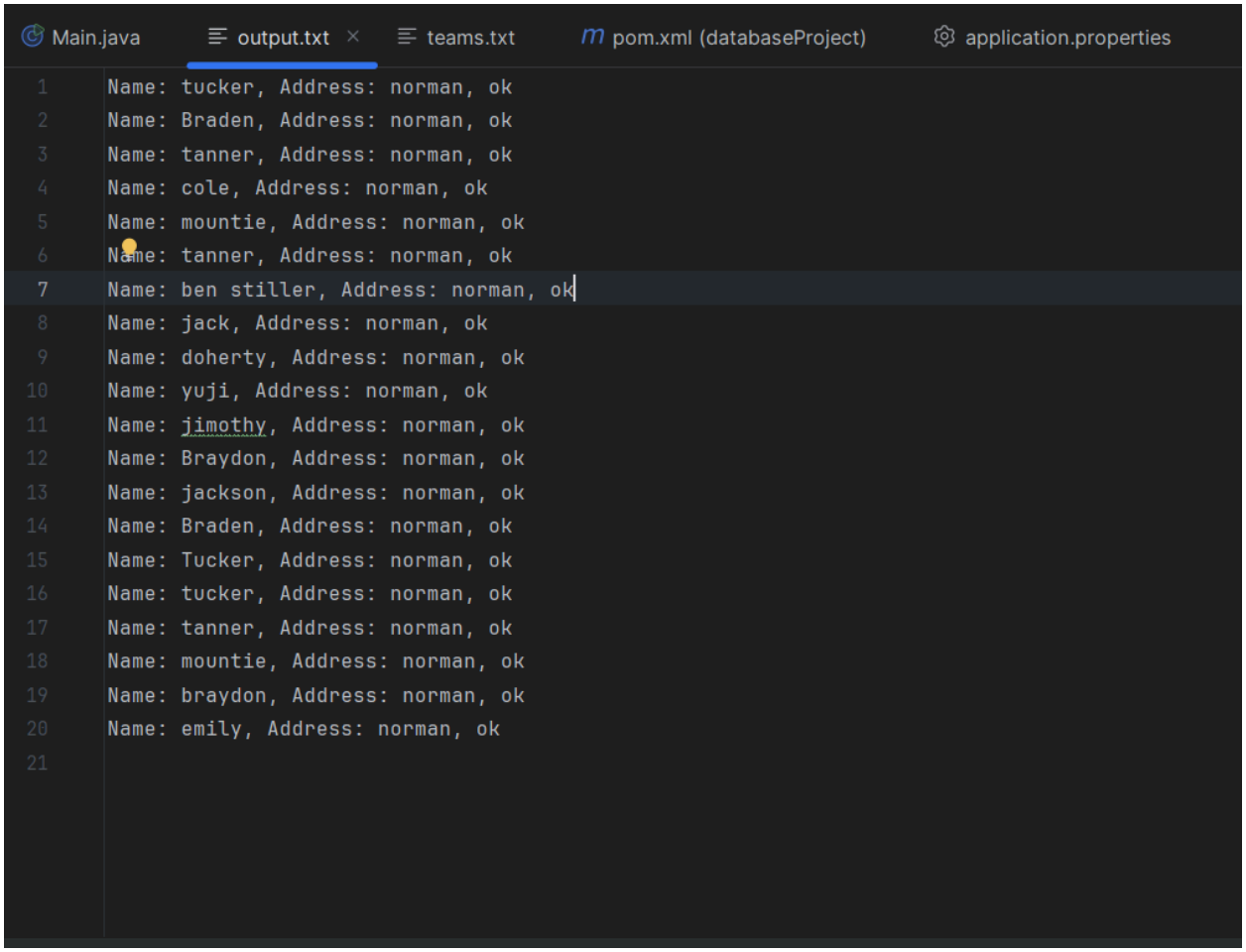
```
3    SELECT * FROM Teams;
4
5
```

Results | Messages

	team_name	team_type	date_formed
1	Team Alpha	Support	2020-07-13
2	Team Bravo	Development	2022-09-10
3	Team Charlie	Operations	2021-04-17
4	Team Delta	Research	2023-11-07
5	Team Echo	Logistics	2018-03-29
6	team1	Leadership	2020-01-01
7	team2	functional	2020-02-01
8	team3	cross-functional	2020-03-01
9	team4	virtual	2020-04-01
10	team5	project	2020-05-01




```
WELCOME TO PAN, YOUR OWN PERSONALIZED DATABASE SYSTEM!
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
17
Enter the output file name (e.g., output.txt): output.txt
Data successfully exported to C:\Users\tanne\IdeaProjects\databaseProject\output.txt
```


Main.java
output.txt
teams.txt
pom.xml (databaseProject)
application.properties
Name: tucker, Address: norman, ok
Name: Braden, Address: norman, ok
Name: tanner, Address: norman, ok
Name: cole, Address: norman, ok
Name: mountie, Address: norman, ok
Name: tanner, Address: norman, ok
Name: ben stiller, Address: norman, ok
Name: jack, Address: norman, ok
Name: doherty, Address: norman, ok
Name: yuji, Address: norman, ok
Name: jimothy, Address: norman, ok
Name: Braydon, Address: norman, ok
Name: jackson, Address: norman, ok
Name: Braden, Address: norman, ok
Name: Tucker, Address: norman, ok
Name: tucker, Address: norman, ok
Name: tanner, Address: norman, ok
Name: mountie, Address: norman, ok
Name: braydon, Address: norman, ok
Name: emily, Address: norman, ok

```
SELECT * FROM Clients;
SELECT * FROM Volunteers;
SELECT * FROM Employees;
SELECT * FROM Donors;
```

Results Messages

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	assignment_date	doctor_name	doctor_phone_number
1	2	Braydon	male	business	1	norman, ok	989-9899-9898	braydon@email.com	2020-01-02	jonathan	988-999-9999
2	3	jackson	male	business	1	norman, ok	986-999-9999	jackson@email.com	2020-01-03	timothy	985-999-9999
3	4	Braden	male	business	1	norman, ok	983-999-9999	braden@email.com	2020-01-04	jimm	982-999-9999
4	5	Tucker	male	doctor	1	norman, ok	980-999-9999	tucker@email.com	2020-01-05	ben	979-999-9999

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	date_joined	training_date	training_location
1	6	ben stiller	male	business	1	norman, ok	977-999-9999	benstiller@email.com	2020-02-01	2020-02-01	norman
2	7	jack	male	business	1	norman, ok	975-999-9999		2020-02-02	2020-02-02	norman
3	8	doherty	male	business	1	norman, ok	973-999-9999	doherty@email.com	2020-02-03	2020-02-03	norman
4	9	yuji	male	business	1	norman, ok	970-999-9999	yuji@email.com	2020-02-04	2020-02-04	norman
5	10	jimothy	male	business	1	norman, ok	968-999-9999	jimothy@email.com	2020-02-05	2020-02-05	norman

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	salary	marital_status	hire_date
1	11	tucker	male	business	1	norman, ok	960-999-9999	tucker@email.com	99999.00	married	2020-03-01
2	12	Braden	male	business	1	norman, ok	958-999-9999	Braden@email.com	99999.00	married	2020-03-02
3	13	tanner	male	business	1	norman, ok	955-999-9999	tanner@email.com	99999.00	married	2020-03-03
4	14	cole	male	business	1	norman, ok	953-999-9999	cole@email.com	99999.00	married	2020-03-04
5	15	mountie	male	business	1	norman, ok	951-999-9999	mountie@email.com	99999.00	married	2020-03-05
6	20	tanner	male	business	1	norman, ok	888-888-8888	tanner@email.com	110000.00	married	2020-06-01

	ssn	person_name	gender	profession	on_mailing_list	mailing_address	phone_number	email_address	is_anon
1	11	tucker	male	business	1	norman, ok	tucker@email.com	960-999-9999	1
2	16	tanner	male	business	1	norman, ok	tanner@email.com	940-999-9999	1
3	17	mountie	male	business	1	norman, ok	mountie@email.com	943-999-9999	1
4	18	braydon	male	business	1	norman, ok	braydon@email.com	945-999-9999	1
5	19	emily	female	business	1	norman, ok	emily@email.com	947-999-9999	1

Task 6.17:

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
1
Enter the Team Name:
team1
Enter the Team Type:
leadership
Enter the date formed (yyyy-MM-dd):
2020-01-01
```

```
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException Create breakpoint : Violation of PRIMARY KEY constraint 'PK__Teams__29E35E0DF6CB0914'.
    at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:259)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1695)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedStatement.java:648)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPreparedStatement.java:567)
    at com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7675)
    at com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:4137)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:272)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:246)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.execute(SQLServerPreparedStatement.java:544)
    at org.example.Main.executeInsertTeamProcedure(Main.java:140)
    at org.example.Main.main(Main.java:56)

Process finished with exit code 1
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
3
Enter volunteer ssn:
6
Enter volunteer name:
tanner
Enter volunteer gender:
male
```

```
Enter volunteer profession:
business
Enter volunteer mailing list status (1 for yes, 0 for no):
1
Enter volunteer mailing address:
norman, ok
Enter volunteer phone number:
111-111-1111
Enter volunteer email:
tanner@email.com
Enter volunteer date joined (yyyy-MM-dd):
2020-01-01
Enter volunteer training date (yyyy-MM-dd):
2020-01-01
Enter volunteer training location:
norman, ok
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException Create breakpoint : Violation of PRIMARY KEY constraint 'PK__Voluntee__DDDF0AE7092E8AE1'.
    at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:259)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1695)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedStatement.java:648)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPreparedStatement.java:567)
    at com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7675)
    at com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:4137)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:272)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:246)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.execute(SQLServerPreparedStatement.java:544)
    at org.example.Main.executeInsertVolunteerProcedure(Main.java:376)
```

```
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
5
Please enter the employee's SSN: 11
Please enter the employee's name: jack
Please enter the employee's gender: male
Please enter the employee's profession: business
Is the employee on the mailing list? (1 for yes, 0 for no): 1
```

```
Please enter the employee's mailing address: norman, ok
Please enter the employee's phone number: 818-818-8181
Please enter the employee's email address: jack@email.com
Please enter the employee's salary: 888888
Please enter the employee's marital status: married
Please enter the employee's hire date (yyyy-MM-dd): 555555
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException Create breakpoint : Error converting data type nvarchar to date.
    at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:259)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1695)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedStatement.java:648)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPreparedStatement.java:567)
    at com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7675)
    at com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:4137)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:272)
    at com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:246)
    at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.execute(SQLServerPreparedStatement.java:544)
    at org.example.Main.executeInsertEmployeeProcedure(Main.java:581)
    at org.example.Main.main(Main.java:68)

Process finished with exit code 1
```

Task 6.18:

```
WELCOME TO PAN, YOUR OWN PERSONALIZED DATABASE SYSTEM!
Choose an option:
1: Insert a Team into the database
2: Insert a Client into the database
3: Insert a Volunteer into the database
4: Insert number of hours volunteer worked this month
5: Insert an Employee into the database
6: Insert an Expense associated with an Employee
7: Insert a Donor into the database
8: Retrieve Doctor information
9: Retrieve Employee charges for a particular time period
10: Retrieve Volunteers that are members of teams that support a particular client
11: Retrieve names of teams founded after particular dates
12: Retrieve information of all people in database
13: Retrieve name and amount donated by donors that are also employees
14: Increase salary of employees with more than one team reporting to them
15: Delete Clients who don't have health insurance and value of importance for transportation is less than 5
16: Import a new team from data file
17: Export names and mailing address of people on mailing list
18: Quit
18
You have exited PAN. Have a great day!

Process finished with exit code 0
```