Project Report: Patient Assistant Network Database System

Name: Subhash Chandra

ID: 113649485

Email: csubhash@ou.edu

Course Name: DBMS

Course Number: CS/DSA-4513-001

Section: MS

Semester: Fall 2024

Instructor: Dr. Le Gruenwald

Table of Contents:

Task 1: ER Diagram:

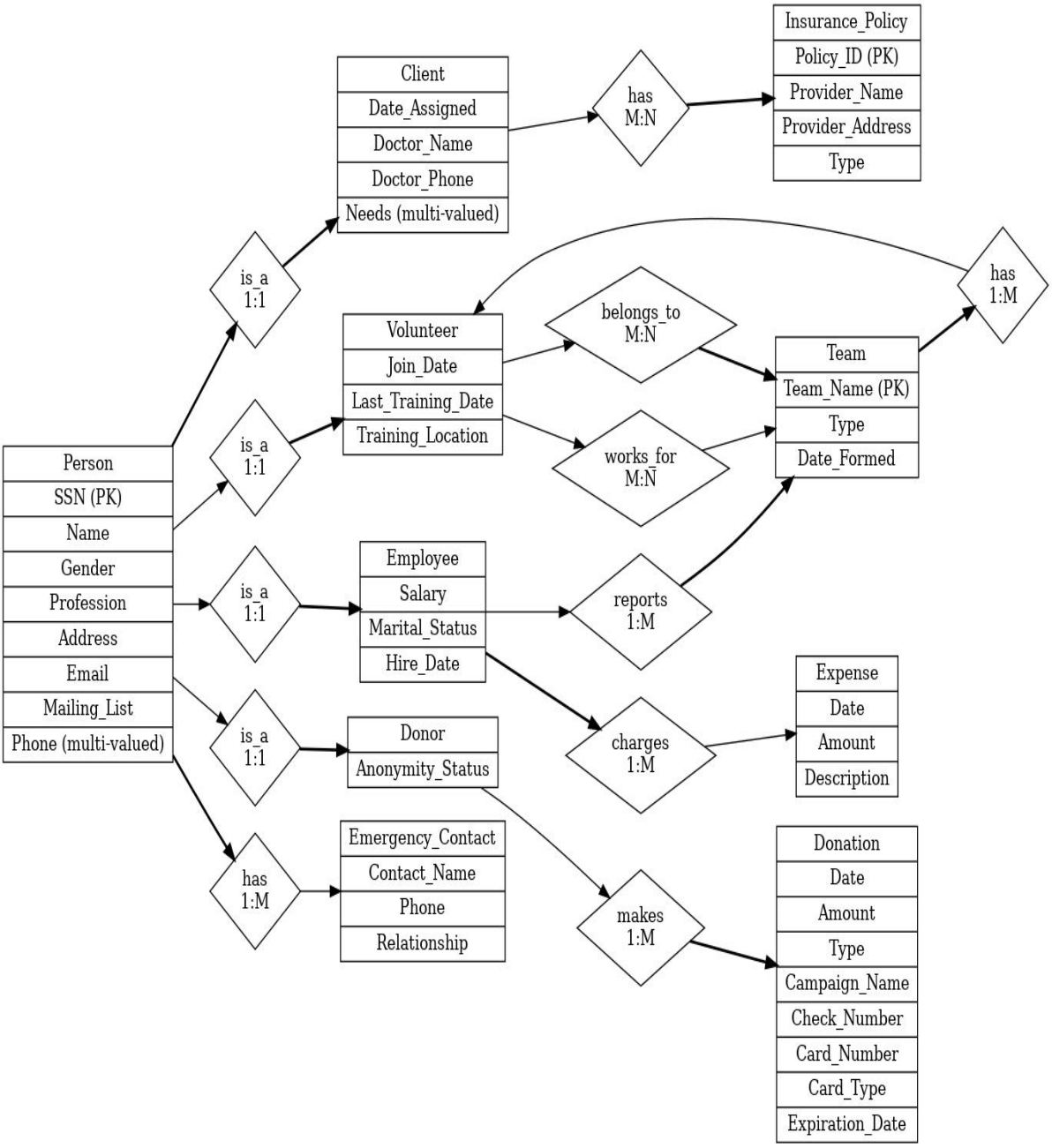


Figure 1: Entity-Relationship (ER) Diagram for Patient Assistant Network Database System

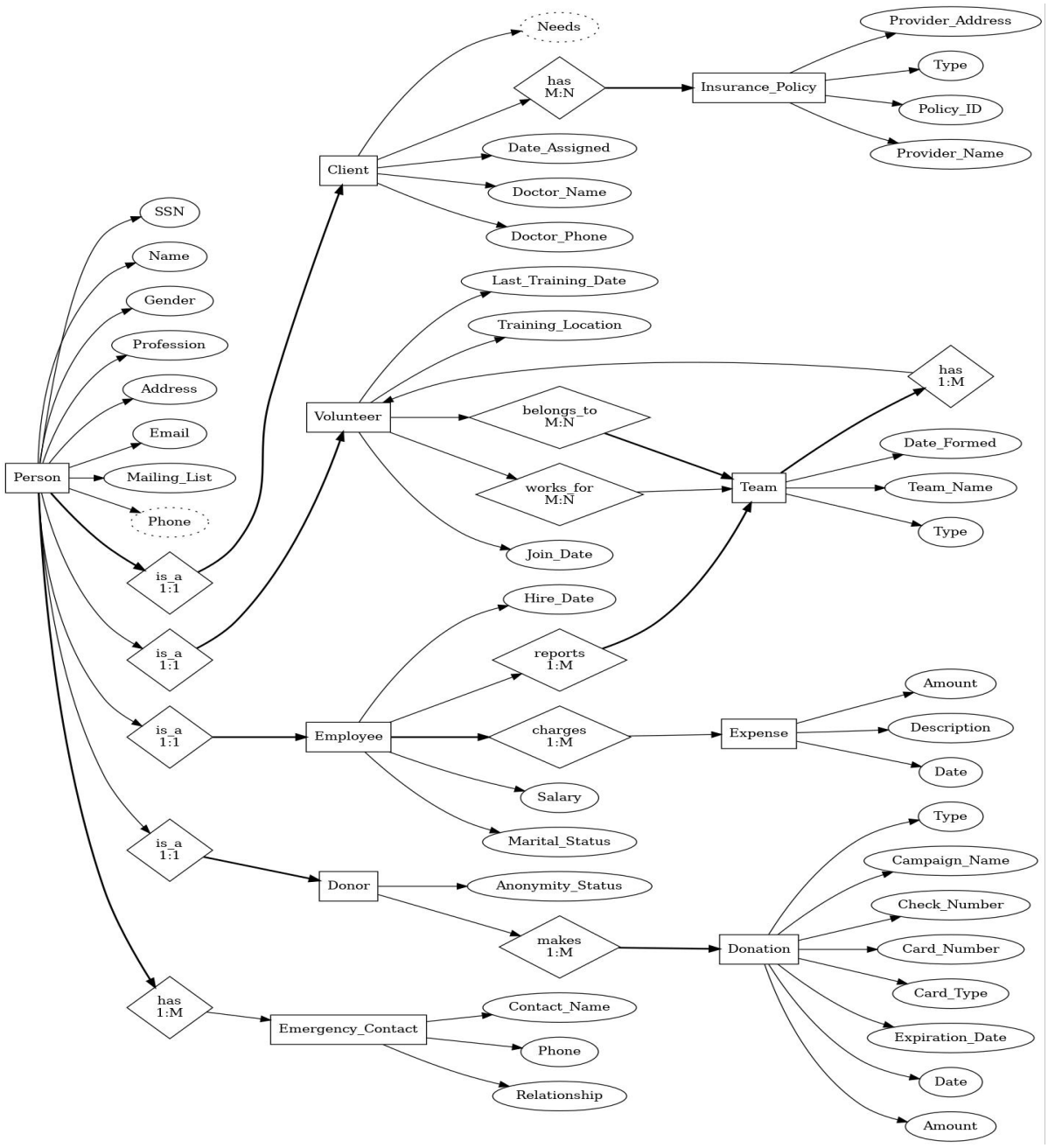


Figure 2: Entity-Relationship (ER) Diagram for Patient Assistant Network Database System

Task 2: Relational Database Schemas:

To convert the ER diagram into a set of relational schemas, we define each entity and relationship as a table. The relationships between the tables are represented using foreign keys.

1. Person Table

Relational Database Schema and corresponding SQL Code:

1. Person

Attributes:

- SSN (VARCHAR) - Primary Key
- Name (VARCHAR)
- Gender (CHAR(1))
- Profession (VARCHAR)
- Address (VARCHAR)
- Email (VARCHAR)
- Mailing_List (BOOLEAN)

Primary Key: SSN

```sql
CREATE TABLE Person (
    SSN VARCHAR PRIMARY KEY,
    Name VARCHAR,
    Gender CHAR(1),
    Profession VARCHAR,
    Address VARCHAR,
    Email VARCHAR,
    Mailing_List BOOLEAN
);
```

—

2. Phone

Attributes:

- SSN (VARCHAR) - Foreign Key
- Phone_Number (VARCHAR)

Primary Key: (SSN, Phone_Number)
Foreign Key: SSN references Person(SSN)

```
CREATE TABLE Phone (
    SSN VARCHAR REFERENCES Person(SSN),
    Phone_Number VARCHAR,
    PRIMARY KEY (SSN, Phone_Number)
);
```

—

3. Client

Attributes:

- Client_SSN (VARCHAR) - Primary Key and Foreign Key
- Date_Assigned (DATE)
- Doctor_Name (VARCHAR)
- Doctor_Phone (VARCHAR)

Primary Key: Client_SSN
Foreign Key: Client_SSN references Person(SSN)

```
CREATE TABLE Client (
    Client_SSN VARCHAR PRIMARY KEY REFERENCES Person(SSN),
    Date_Assigned DATE,
    Doctor_Name VARCHAR,
    Doctor_Phone VARCHAR
);
```

—

4. Need

Attributes:

- Client_SSN (VARCHAR) - Foreign Key
- Need_Type (VARCHAR)
- Importance (INTEGER) - CHECK (Importance BETWEEN 1 AND 10)

Primary Key: (Client_SSN, Need_Type)
Foreign Key: Client_SSN references Client(Client_SSN)

```
CREATE TABLE Need (
    Client_SSN VARCHAR REFERENCES Client(Client_SSN),
    Need_Type VARCHAR,
    Importance INTEGER CHECK (Importance BETWEEN 1 AND 10),
    PRIMARY KEY (Client_SSN, Need_Type)
);
```

—

5. Insurance_Policy

Attributes:

- Policy_ID (VARCHAR) - Primary Key
- Provider_Name (VARCHAR)
- Provider_Address (VARCHAR)
- Type (ENUM: 'life', 'health', 'home', 'auto')

Primary Key: Policy_ID

```
CREATE TABLE Insurance_Policy (
    Policy_ID VARCHAR PRIMARY KEY,
    Provider_Name VARCHAR,
    Provider_Address VARCHAR,
    Type ENUM('life', 'health', 'home', 'auto')
);
```

—

6. Client_Insurance_Policy

Attributes:

- Client_SSN (VARCHAR) - Foreign Key
- Policy_ID (VARCHAR) - Foreign Key

Primary Key: (Client_SSN, Policy_ID)
Foreign Keys:

- Client_SSN references Client(Client_SSN)
- Policy_ID references Insurance_Policy(Policy_ID)

```
CREATE TABLE Client_Insurance_Policy (
    Client_SSN VARCHAR REFERENCES Client(Client_SSN),
    Policy_ID VARCHAR REFERENCES Insurance_Policy(Policy_ID),
    PRIMARY KEY (Client_SSN, Policy_ID)
);
```

—

7. Team

Attributes:

- Team_Name (VARCHAR) - Primary Key
- Type (VARCHAR)
- Date_Formed (DATE)

Primary Key: Team_Name

```
CREATE TABLE Team (
    Team_Name VARCHAR PRIMARY KEY,
    Type VARCHAR,
    Date_Formed DATE
);
```

—

8. Volunteer

Attributes:

- Volunteer_SSN (VARCHAR) - Primary Key and Foreign Key
- Join_Date (DATE)
- Last_Training_Date (DATE)
- Training_Location (VARCHAR)

Primary Key: Volunteer_SSN
Foreign Key: Volunteer_SSN references Person(SSN)

```
CREATE TABLE Volunteer (
    Volunteer_SSN VARCHAR PRIMARY KEY REFERENCES Person(SSN),
    Join_Date DATE,
    Last_Training_Date DATE,
    Training_Location VARCHAR
);
```

—

9. Volunteer_Team

Attributes:

- Volunteer_SSN (VARCHAR) - Foreign Key
- Team_Name (VARCHAR) - Foreign Key
- Is_Leader (BOOLEAN)
- Is_Active (BOOLEAN)
- Work_Hours (INTEGER)

Primary Key: (Volunteer_SSN, Team_Name)
Foreign Keys:

- Volunteer_SSN references Volunteer(Volunteer_SSN)
- Team_Name references Team(Team_Name)

```
CREATE TABLE Volunteer_Team (
    Volunteer_SSN VARCHAR REFERENCES Volunteer(Volunteer_SSN),
    Team_Name VARCHAR REFERENCES Team(Team_Name),
    Is_Leader BOOLEAN,
    Is_Active BOOLEAN,
    Work_Hours INTEGER,
    PRIMARY KEY (Volunteer_SSN, Team_Name)
);
```

—

10. Employee

Attributes:

- Employee_SSN (VARCHAR) - Primary Key and Foreign Key
- Salary (DECIMAL)
- Marital_Status (VARCHAR)
- Hire_Date (DATE)

Primary Key: Employee_SSN
Foreign Key: Employee_SSN references Person(SSN)

```
CREATE TABLE Employee (
    Employee_SSN VARCHAR PRIMARY KEY REFERENCES Person(SSN),
    Salary DECIMAL,
    Marital_Status VARCHAR,
    Hire_Date DATE
);
```

—

11. Expense

Attributes:

- Expense_ID (SERIAL) - Primary Key
- Employee_SSN (VARCHAR) - Foreign Key
- Date (DATE)
- Amount (DECIMAL)
- Description (VARCHAR)

Primary Key: Expense_ID
Foreign Key: Employee_SSN references Employee(Employee_SSN)

```
CREATE TABLE Expense (
    Expense_ID SERIAL PRIMARY KEY,
    Employee_SSN VARCHAR REFERENCES Employee(Employee_SSN),
    Date DATE,
    Amount DECIMAL,
    Description VARCHAR
);
```

—

12. Team_Report

Attributes:

- Team_Name (VARCHAR) - Foreign Key
- Employee_SSN (VARCHAR) - Foreign Key
- Report_Date (DATE)
- Content_Description (VARCHAR)

Primary Key: (Team_Name, Employee_SSN, Report_Date)
Foreign Keys:

- Team_Name references Team(Team_Name)
- Employee_SSN references Employee(Employee_SSN)

```sql
CREATE TABLE Team_Report (
    Team_Name VARCHAR REFERENCES Team(Team_Name),
    Employee_SSN VARCHAR REFERENCES Employee(Employee_SSN),
    Report_Date DATE,
    Content_Description VARCHAR,
    PRIMARY KEY (Team_Name, Employee_SSN, Report_Date)
);
```

—

13. Donor

Attributes:

- Donor_SSN (VARCHAR) - Primary Key and Foreign Key
- Anonymity_Status (BOOLEAN)

Primary Key: Donor_SSN
Foreign Key: Donor_SSN references Person(SSN)

```sql
CREATE TABLE Donor (
    Donor_SSN VARCHAR PRIMARY KEY REFERENCES Person(SSN),
    Anonymity_Status BOOLEAN
);
```

—

14. Donation

Attributes:

- Donation_ID (SERIAL) - Primary Key
- Donor_SSN (VARCHAR) - Foreign Key
- Date (DATE)
- Amount (DECIMAL)
- Type (VARCHAR)
- Campaign_Name (VARCHAR)
- Check_Number (INTEGER) - NULLABLE
- Card_Number (VARCHAR) - NULLABLE
- Card_Type (VARCHAR) - NULLABLE
- Expiration_Date (DATE) - NULLABLE

Primary Key: Donation_ID
Foreign Key: Donor_SSN references Donor(Donor_SSN)

```sql
CREATE TABLE Donation (
    Donation_ID SERIAL PRIMARY KEY,
    Donor_SSN VARCHAR REFERENCES Donor(Donor_SSN),
    Date DATE,
    Amount DECIMAL,
    Type VARCHAR,
    Campaign_Name VARCHAR,
    Check_Number INTEGER NULL,
    Card_Number VARCHAR NULL,
    Card_Type VARCHAR NULL,
    Expiration_Date DATE NULL
);
```

—

15. Emergency_Contact

Attributes:

- Contact_ID (SERIAL) - Primary Key
- Person_SSN (VARCHAR) - Foreign Key
- Contact_Name (VARCHAR)
- Phone (VARCHAR)
- Relationship (VARCHAR)

Primary Key: Contact_ID
Foreign Key: Person_SSN references Person(SSN)

```
CREATE TABLE Emergency_Contact (
    Contact_ID SERIAL PRIMARY KEY,
    Person_SSN VARCHAR REFERENCES Person(SSN),
    Contact_Name VARCHAR,
    Phone VARCHAR,
    Relationship VARCHAR
);
```

1 Task 3:

Storage Structure Choices for Each Relational Table:

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Person	12 (Random Search)	SSN	1/week	Heap File with B+ Tree Index on SSN	The B+ Tree index on `SSN` supports efficient searches for unique persons based on SSN, optimizing retrieval speed and storage space. A Heap file allows efficient insertions, as records are appended, which suits the frequency and variety of insertions in this table.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Person	1, 2, 3, 5, 7 (Insertion)	N/A	Varies	Heap File	The frequent insertions across `Person` and subtype tables are efficiently handled by a Heap file, allowing high throughput and efficient storage. Since these are non-ordered insertions, Heap is optimal.
Person	4, 6, 15 (Deletion)	N/A	Varies	Heap File	Occasional deletions are easily managed in Heap files by marking records as deleted without reorganizing data. The lecture's discussion on free lists could also support marking deleted records for re-use.
Phone	N/A	SSN, Phone_Number	N/A	Heap File	Multi-valued attributes like `Phone` are best suited to Heap files for flexibility and ease of insertion, allowing multiple phone entries per `SSN`, as the lecture suggests Heap organization for unordered, multiple entries.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Client	2 (Insertion)	Client_SSN	1/week	Heap File with B+ Tree Index on Client_SSN	The B+ Tree index on `Client_SSN` allows efficient searches and insertions, particularly for clients assigned to multiple teams. B+ Trees are well-suited for range and random search due to their balanced nature.
Client	8 (Random Search)	Doctor_Name, Doctor_Phone	1/week	B+ Tree on Doctor_Name	The B+ Tree index on `Doctor_Name` enables efficient retrieval of doctor information for each client, optimized for random access, as it offers fast retrieval in a sorted order.
Need	N/A	Client_SSN, Need_Type	N/A	Heap File	Heap files are ideal for storing multi-valued, single-key attributes with minimal search and update requirements, as `Need` has no frequent query access. The lecture suggests Heap for such cases with low retrieval needs.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Insurance_Policy	15 (Range Search)	Type	4/year	Clustered Index on Policy_ID and Type	A clustered index on `Type` optimizes range-based searches for insurance policies sorted by type, providing efficient access for specific policy types, as discussed in Indexed-Sequential Files in the lecture.
Team	1 (Insertion)	Team_Name	1/month	Heap File with B+ Tree Index on Team_Name	Heap file supports insertions with minimal storage overhead, while the B+ Tree index enables quick access by `Team_Name`. As the lecture discusses, B+ Tree indexing is ideal for sequential and ordered search with high insertion volume.
Team	11 (Range Search)	Date_Formed	1/month	Clustered Index on Date_Formed	Clustered indexing on `Date_Formed` facilitates efficient retrieval of teams based on formation dates, especially useful for ordered queries, as noted in Sequential and Indexed-Sequential organization.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Volunteer	3 (Insertion)	Volunteer_SSN	2/month	Heap File with B+ Tree Index on Volunteer_SSN	Frequent insertions are supported by a Heap file, with the B+ Tree index enabling fast access to individual volunteers by SSN. B+ Tree indexing ensures efficient retrieval even in high-volume tables.
Volunteer	10 (Random Search)	Volunteer_SSN	4/year	B+ Tree Index on Volunteer_SSN	A B+ Tree index on `Volunteer_SSN` allows efficient retrieval of volunteers, supporting random access for volunteer-client associations. B+ Trees' balanced structure and efficient key ordering fit this use case well.
Volunteer_Team	4 (Insertion)	Volunteer_SSN, Team_Name	30/month	Heap File with Composite B+ Tree Index on Volunteer_SSN, Team_Name	Composite B+ Tree index on `Volunteer_SSN` and `Team_Name` facilitates efficient insertions and search for specific volunteer-team pairings. The lecture notes B+ Tree indexes as effective for multi-key access.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Volunteer_Team	14 (Range Search)	Work_Hours	1/year	Clustered Index on Work_Hours	Clustered index on `Work_Hours` supports efficient range-based queries for volunteer work hours, enabling streamlined reporting, as per Indexed-Sequential file suggestions.
Employee	5 (Insertion)	Employee_SSN	1/year	Heap File with B+ Tree Index on Employee_SSN	Supports infrequent insertions and quick lookups for employee information with a B+ Tree index on `Employee_SSN`, ensuring ordered retrieval for reporting.
Employee	9 (Range Search)	Employee_SSN	1/month	B+ Tree on Employee_SSN	Enables sorted access and range queries for employee expense summaries, optimized by indexing on `Employee_SSN`. Lecture notes suggest B+ Tree as efficient for range searches on primary keys.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Expense	6 (Insertion)	Expense_ID, Date	1/day	Heap File with B+ Tree Index on Date	Heap file supports daily insertions, with the B+ Tree index on `Date` enabling ordered retrieval of expense records for range queries. Indexed-Sequential file organization supports this usage.
Team_Report	N/A	Team_Name, Employee_SSN	N/A	Heap File with Composite B+ Tree Index on Team_Name, Employee_SSN	Supports efficient search for team reports linked to employees, allowing quick access and updates as needed. Multiple-key indices discussed in the lecture are implemented here to facilitate multi-key access.
Donor	7 (Insertion)	Donor_SSN	1/day	Heap File with B+ Tree Index on Donor_SSN	Supports frequent insertions, with a B+ Tree index on `Donor_SSN` for ordered retrieval of unique donors, as B+ Tree indexes efficiently support high-volume insertions and random search.

Table Name	**Query # and Type**	**Search Key**	**Query Frequency**	**Selected File Organization**	**Justifications**
Donor	13 (Random Search)	Donor_SSN	1/week	B+ Tree on Donor_SSN	Supports efficient retrieval of donors by SSN, especially when searching for donors who are also employees. Random search benefits from B+ Tree index optimization, as discussed in lecture.
Donation	7 (Insertion)	Donation_ID, Donor_SSN, Date	1/day	Heap File with Composite Index on Donor_SSN and Date	Frequent insertions are handled well by Heap file organization, with the composite index allowing efficient lookups by `Donor_SSN` and `Date`. This structure aligns with lecture suggestions for multi-key access using indexes.
Emergency_Contact	12 (Random Search)	Person_SSN	1/week	Heap File with B+ Tree Index on Person_SSN	Supports retrieval by `Person_SSN` to access emergency contact information, optimized by a B+ Tree index for random access, which is consistent with lecture guidance on random access.

3.2: Storage Structure Choices for Each Relational Table in Azure SQL Database

Table Name	**Selected File Organization in Azure SQL**	**Justifications**
Person	Clustered Index on SSN	Clustered indexes in Azure SQL Database provide efficient access to unique keys and reduce storage for primary key searches. `SSN` is the primary key, so a clustered index ensures efficient random searches and retrieval for SSN-based queries.
Phone	Heap File	Azure SQL supports multi-valued attributes by default, allowing `Phone` to remain as a Heap table. Using a Heap file for `Phone` keeps the storage flexible and lightweight for managing multiple entries per person.
Client	Non-Clustered Index on Client_SSN	A non-clustered index on `Client_SSN` provides fast access for client-specific searches. Since clustered indexes are limited in use, Azure SQL allows non-clustered indexing on frequently queried foreign keys, which optimizes lookups in tables that reference `Client_SSN`.
Need	Heap File	`Need` remains as a Heap structure in Azure SQL Database, as it primarily stores multi-valued attributes without frequent random access. The Heap file in Azure handles unordered data well with minimal storage overhead.
Insurance_Policy	Clustered Index on Policy_ID and Non-Clustered Index on Type	The clustered index on `Policy_ID` optimizes unique access, while a non-clustered index on `Type` enhances range searches. This combination in Azure SQL allows rapid retrieval of specific policy types without compromising on insertion performance.
Team	Clustered Index on Team_Name	Team names are frequently accessed for random searches; thus, a clustered index on `Team_Name` is optimal in Azure SQL. This index enables fast retrieval of team information while supporting ordered access based on team name.

Table Name	**Selected File Organization in Azure SQL**	**Justifications**
Volunteer	Clustered Index on Volunteer_SSN	Clustered indexing on `Volunteer_SSN` ensures efficient access for primary key-based searches, while Azure's automatic tuning can suggest further indexing improvements if query patterns evolve.
Volunteer_Team	Clustered Index on Volunteer_SSN, Non-Clustered Index on Team_Name	`Volunteer_SSN` as a clustered index ensures fast access for volunteer queries, while a non-clustered index on `Team_Name` enhances search for volunteers associated with specific teams, a common query type.
Employee	Clustered Index on Employee_SSN	With a clustered index on `Employee_SSN`, Azure SQL optimizes unique employee record access, particularly for range searches on expenses or work hours, which are common in this table.
Expense	Columnstore Index on Date	Columnstore indexing is ideal for analytical or read-heavy tables like `Expense`. Azure SQL's columnstore index optimizes space and retrieval efficiency for expense-related data.
Team_Report	Clustered Index on Team_Name, Non-Clustered Index on Employee_SSN	The clustered index on `Team_Name` enables fast team-based access, while a non-clustered index on `Employee_SSN` optimizes report lookups by employee.
Donor	Clustered Index on Donor_SSN	Using `Donor_SSN` as a clustered index allows efficient retrieval of unique donors, supporting frequent insertions and retrievals by SSN. Azure's clustered indexing on primary keys is ideal for random searches.
Donation	Columnstore Index on Donor_SSN and Date	Columnstore indexing is efficient for tables with high read demand, like `Donation`. Storing data by columns reduces space and speeds up aggregated or ordered retrievals for analytics on donations, aligned with Azure's performance optimization strategies.
Emergency_Contact	Non-Clustered Index on Person_SSN	A non-clustered index on `Person_SSN` allows efficient retrieval of emergency contact information, supporting frequent lookups without requiring ordering, which is effective for contact information in Azure SQL.

Justifications Summary

- **Clustered Indexes**: Primary key columns such as `SSN`, `Policy_ID`, and other unique fields use clustered indexes in Azure SQL for efficient retrieval. Clustered indexes support fast, primary key-based searches, reducing storage overhead.
- **Non-Clustered Indexes**: Non-clustered indexes improve retrieval times on foreign keys like `Client_SSN` and `Employee_SSN` and support lookups for secondary attributes such as `Type` and `Person_SSN`.
- **Columnstore Indexes**: Used in read-heavy tables like `Expense` and `Donation`. Columnstore indexes optimize space and improve performance for analytical workloads.
- **Heap Files**: Tables with multi-valued attributes, such as `Phone` and `Need`, use Heap file storage. Azure SQL effectively manages unordered data in Heap files, allowing for flexible insertions.

Azure SQL Features for Optimization

Azure SQL Database provides additional optimizations:

- **Automatic Index Tuning**: Azure SQL's Automatic Indexing can monitor queries and suggest further indexing improvements based on actual usage patterns.
- **Adaptive Query Processing**: Azure SQL dynamically optimizes queries, adjusting indexes and caching to improve performance over time.

These adjustments allow for efficient query processing and data management tailored to Azure SQL's indexing and storage capabilities.

Task 4:

Table Creation in Azure SQL Database:

Below are SQL statements for creating each table, with constraints and indexes according to Task 3.2. I am writing Azure SQL Database code first for each table and in next page corresponding screenshot of Azure SQL.

Table 1: Person

```sql
CREATE TABLE Person (
    SSN VARCHAR(50) PRIMARY KEY CLUSTERED,  -- Primary key with clustered index
    Name VARCHAR(100) NOT NULL,
    Gender CHAR(1) CHECK (Gender IN ('M', 'F', 'O')),  -- Gender constraint
    Profession VARCHAR(100),
    Address VARCHAR(200),
    Email VARCHAR(100) UNIQUE,  -- Ensures email is unique
    Mailing_List BIT NOT NULL  -- Boolean for mailing list subscription
);
```

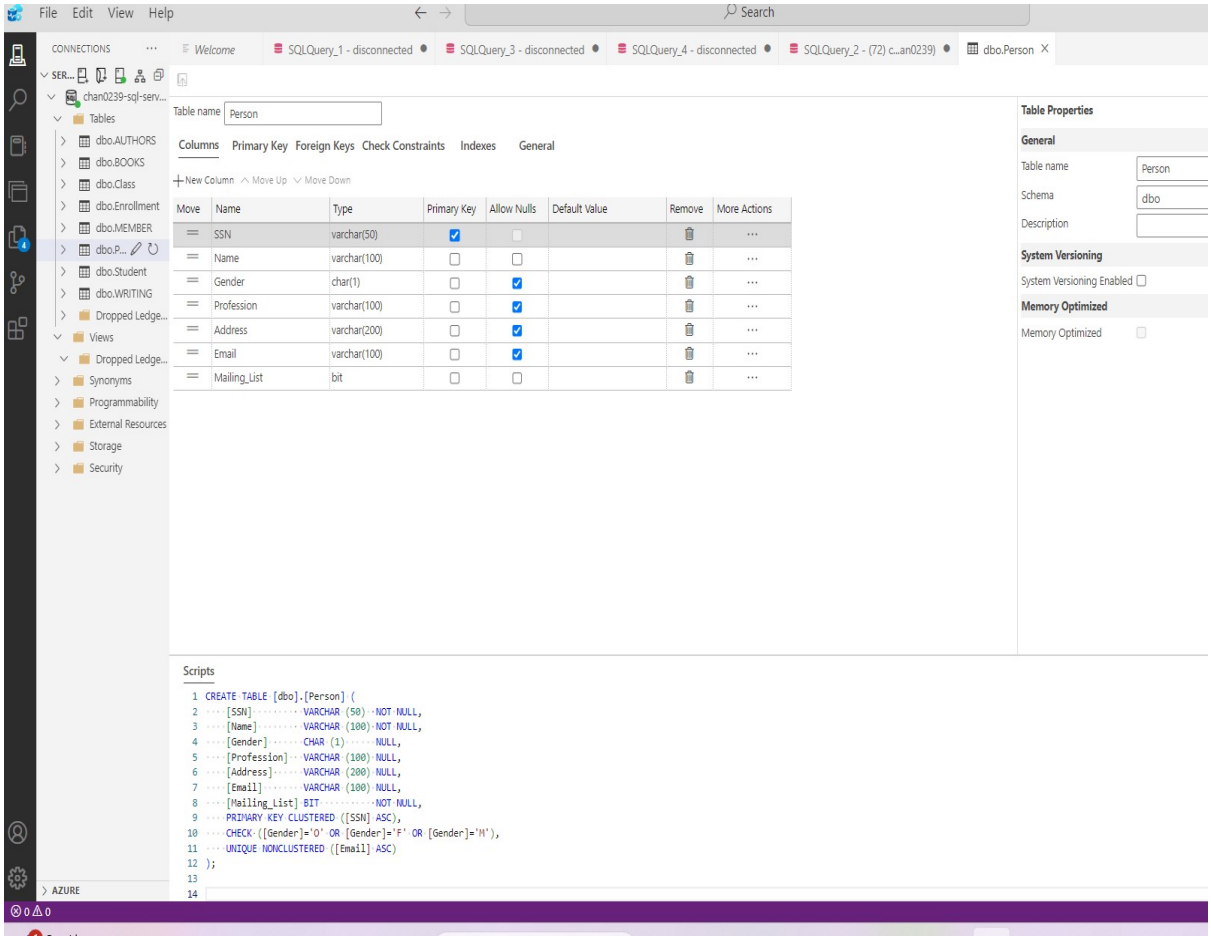

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_1 - disconnected
SQLQuery_3 - disconnected
SQLQuery_4 - disconnected
SQLQuery_2 - (72) c...an0239)
dbo.Person
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Enrollment
dbo.MEMBER
dbo.P...
dbo.Student
dbo.WRITING
Dropped Ledge...
Views
Dropped Ledge...
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Person
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
SSN varchar(50)
Name varchar(100)
Gender char(1)
Profession varchar(100)
Address varchar(200)
Email varchar(100)
Mailing_List bit
Table Properties
General
Table name Person
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Person] (
[SSN] VARCHAR (50) NOT NULL,
[Name] VARCHAR (100) NOT NULL,
[Gender] CHAR (1) NULL,
[Profession] VARCHAR (100) NULL,
[Address] VARCHAR (200) NULL,
[Email] VARCHAR (100) NULL,
[Mailing_List] BIT NOT NULL,
PRIMARY KEY CLUSTERED ([SSN] ASC),
CHECK ([Gender]='O' OR [Gender]='F' OR [Gender]='M'),
UNIQUE NONCLUSTERED ([Email] ASC)
);
0 0

Table 2: Phone

```sql
CREATE TABLE Phone (
    SSN VARCHAR(50) NOT NULL,
    Phone_Number VARCHAR(20),
    PRIMARY KEY (SSN, Phone_Number),  -- Composite primary key for multi-valued attribute
    FOREIGN KEY (SSN) REFERENCES Person(SSN) ON DELETE CASCADE
);
```

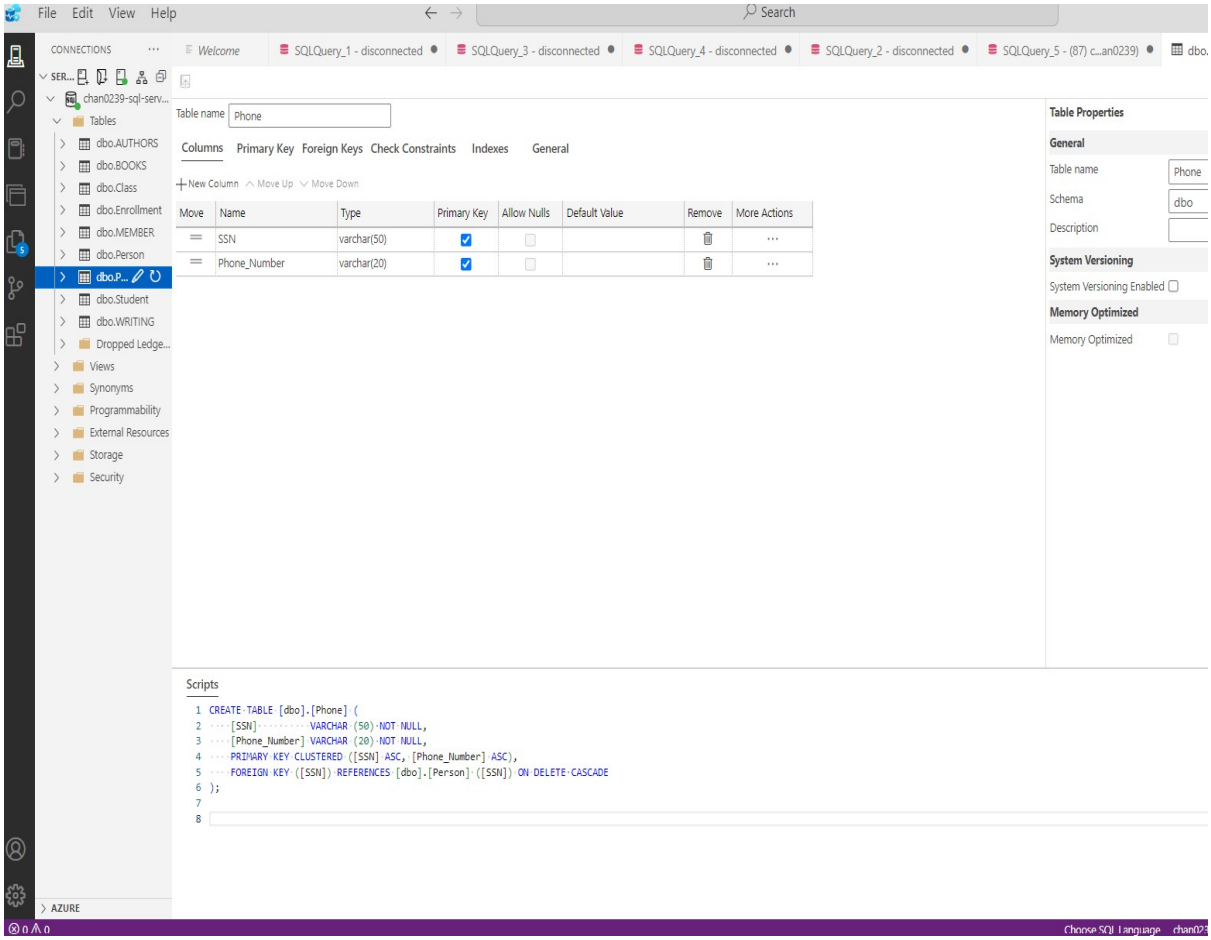

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_1 - disconnected
SQLQuery_3 - disconnected
SQLQuery_4 - disconnected
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Enrollment
dbo.MEMBER
dbo.Person
dbo.P...
dbo.Student
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Phone
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
SSN varchar(50)
Phone_Number varchar(20)
Table Properties
General
Table name Phone
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Phone] (
[SSN] VARCHAR (50) NOT NULL,
[Phone_Number] VARCHAR (20) NOT NULL,
PRIMARY KEY CLUSTERED ([SSN] ASC, [Phone_Number] ASC),
FOREIGN KEY ([SSN]) REFERENCES [dbo].[Person] ([SSN]) ON DELETE CASCADE
);
Choose SQL Language

Table 3: Client

```sql
CREATE TABLE Client (
    Client_SSN VARCHAR(50) PRIMARY KEY,  -- Primary key
    Date_Assigned DATE NOT NULL,
    Doctor_Name VARCHAR(100),
    Doctor_Phone VARCHAR(20),
    FOREIGN KEY (Client_SSN) REFERENCES Person(SSN) ON DELETE CASCADE
);

-- Create a non-clustered index on Client_SSN for fast access
CREATE NONCLUSTERED INDEX idx_Client_SSN ON Client (Client_SSN);
```

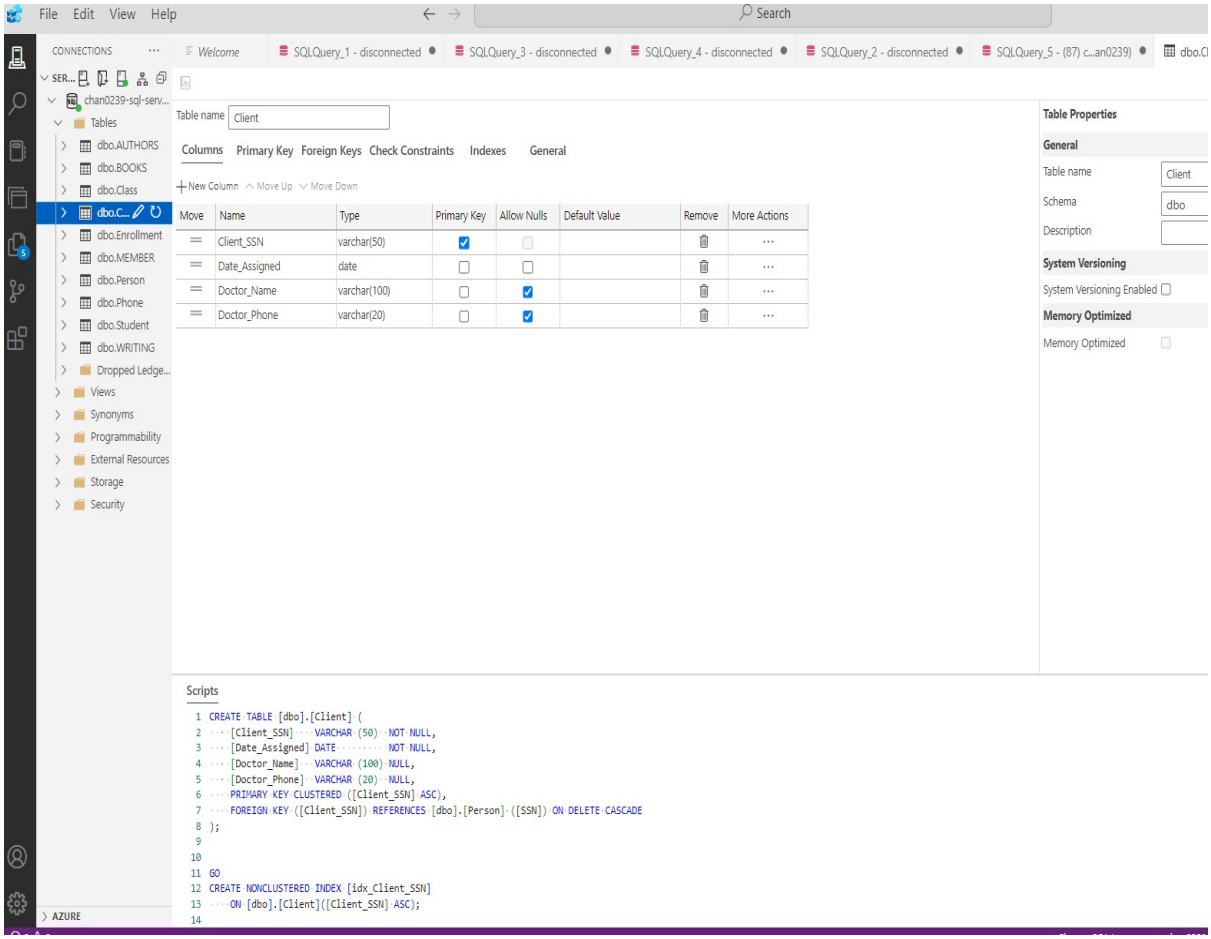

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_1 - disconnected
SQLQuery_3 - disconnected
SQLQuery_4 - disconnected
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.C...
dbo.Enrollment
dbo.MEMBER
dbo.Person
dbo.Phone
dbo.Student
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Client
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Client_SSN varchar(50)
Date_Assigned date
Doctor_Name varchar(100)
Doctor_Phone varchar(20)
Table Properties
General
Table name Client
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Client] (
[Client_SSN] VARCHAR (50) NOT NULL,
[Date_Assigned] DATE NOT NULL,
[Doctor_Name] VARCHAR (100) NULL,
[Doctor_Phone] VARCHAR (20) NULL,
PRIMARY KEY CLUSTERED ([Client_SSN] ASC),
FOREIGN KEY ([Client_SSN]) REFERENCES [dbo].[Person] ([SSN]) ON DELETE CASCADE
);
GO
CREATE NONCLUSTERED INDEX [idx_Client_SSN]
ON [dbo].[Client]([Client_SSN] ASC);

Table 4: Need

```sql
CREATE TABLE Need (
    Client_SSN VARCHAR(50) NOT NULL,
    Need_Type VARCHAR(100),
    Importance INT CHECK (Importance BETWEEN 1 AND 10),
    PRIMARY KEY (Client_SSN, Need_Type),  -- Composite primary key for multi-valued attribute
    FOREIGN KEY (Client_SSN) REFERENCES Client(Client_SSN) ON DELETE CASCADE
);
```

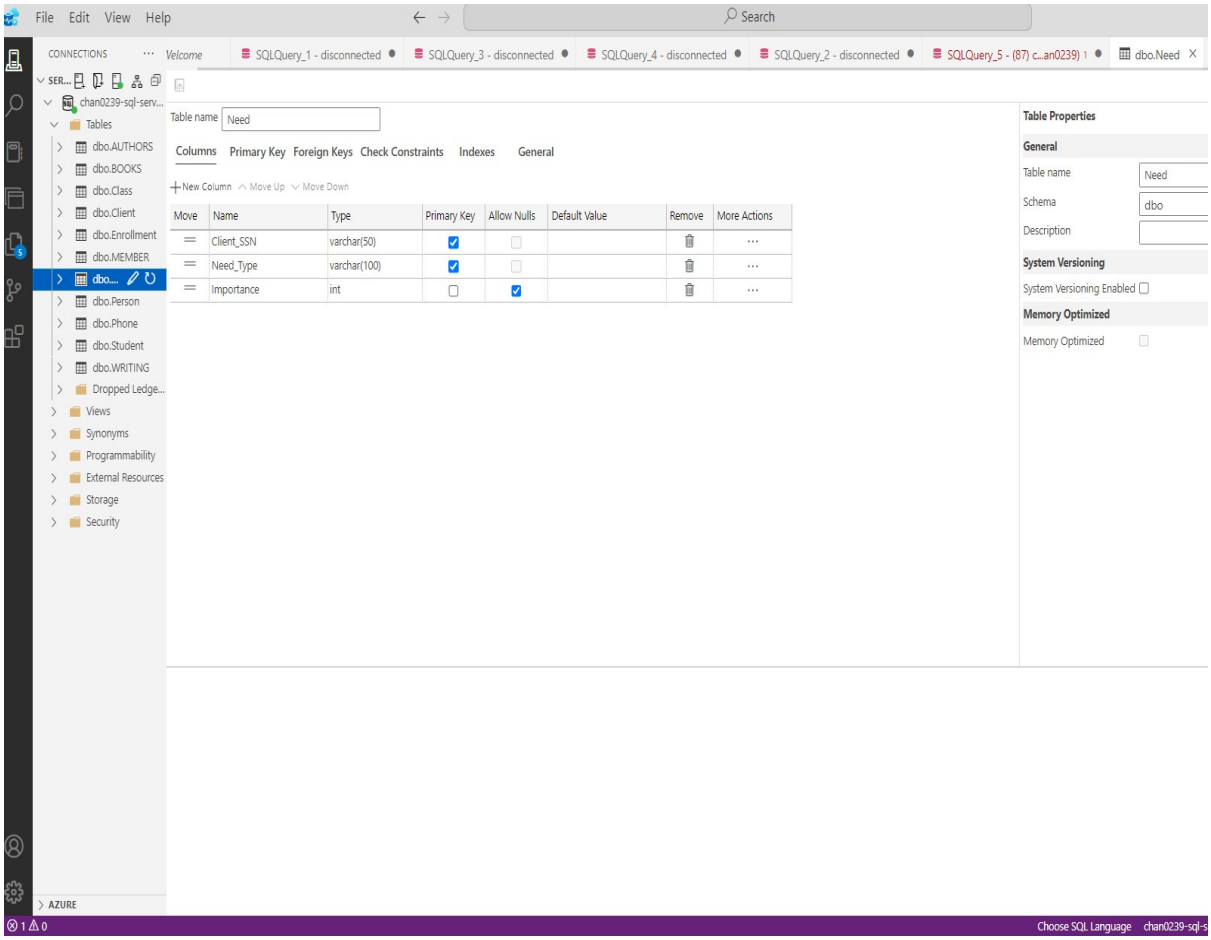

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_1 - disconnected
SQLQuery_3 - disconnected
SQLQuery_4 - disconnected
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239) 1
dbo.Need
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.Enrollment
dbo.MEMBER
dbo....
dbo.Person
dbo.Phone
dbo.Student
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name
Need
Columns
Primary Key
Foreign Keys
Check Constraints
Indexes
General
New Column
Move Up
Move Down
Move
Name
Type
Primary Key
Allow Nulls
Default Value
Remove
More Actions
Client_SSN
varchar(50)
Need_Type
varchar(100)
Importance
int
Table Properties
General
Table name
Need
Schema
dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Choose SQL Language
chan0239-sql-s

Table 5: Insurance_Policy

```sql
CREATE TABLE Insurance_Policy (
    Policy_ID VARCHAR(50) PRIMARY KEY CLUSTERED,  -- Clustered index on primary key
    Provider_Name VARCHAR(100),
    Provider_Address VARCHAR(200),
    Type VARCHAR(20) CHECK (Type IN ('life', 'health', 'home', 'auto'))  -- Constraint on Type
);

-- Non-clustered index on Type for efficient range-based access
CREATE NONCLUSTERED INDEX idx_Insurance_Type ON Insurance_Policy (Type);
```

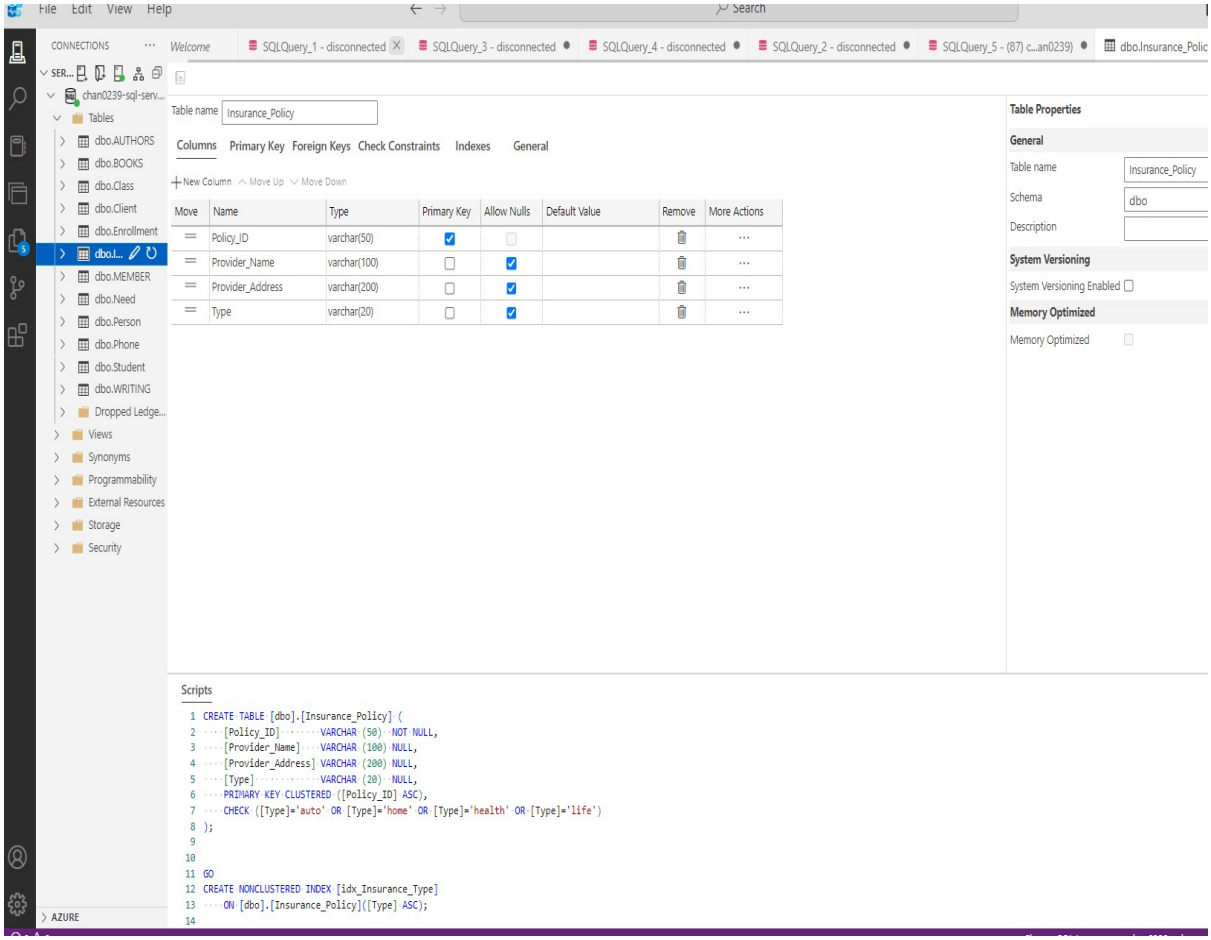

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_1 - disconnected
SQLQuery_3 - disconnected
SQLQuery_4 - disconnected
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
dbo.Insurance_Polic
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.Enrollment
dbo.I...
dbo.MEMBER
dbo.Need
dbo.Person
dbo.Phone
dbo.Student
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Insurance_Policy
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Policy_ID varchar(50)
Provider_Name varchar(100)
Provider_Address varchar(200)
Type varchar(20)
Table Properties
General
Table name Insurance_Policy
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Insurance_Policy] (
[Policy_ID] VARCHAR (50) NOT NULL,
[Provider_Name] VARCHAR (100) NULL,
[Provider_Address] VARCHAR (200) NULL,
[Type] VARCHAR (20) NULL,
PRIMARY KEY CLUSTERED ([Policy_ID] ASC),
CHECK ([Type]='auto' OR [Type]='home' OR [Type]='health' OR [Type]='life')
);
GO
CREATE NONCLUSTERED INDEX [idx_Insurance_Type]
ON [dbo].[Insurance_Policy]([Type] ASC);

Table 6: Team

```sql
CREATE TABLE Team (
    Team_Name VARCHAR(50) PRIMARY KEY CLUSTERED,  -- Clustered index on primary key
    Type VARCHAR(100),
    Date_Formed DATE NOT NULL
);
```

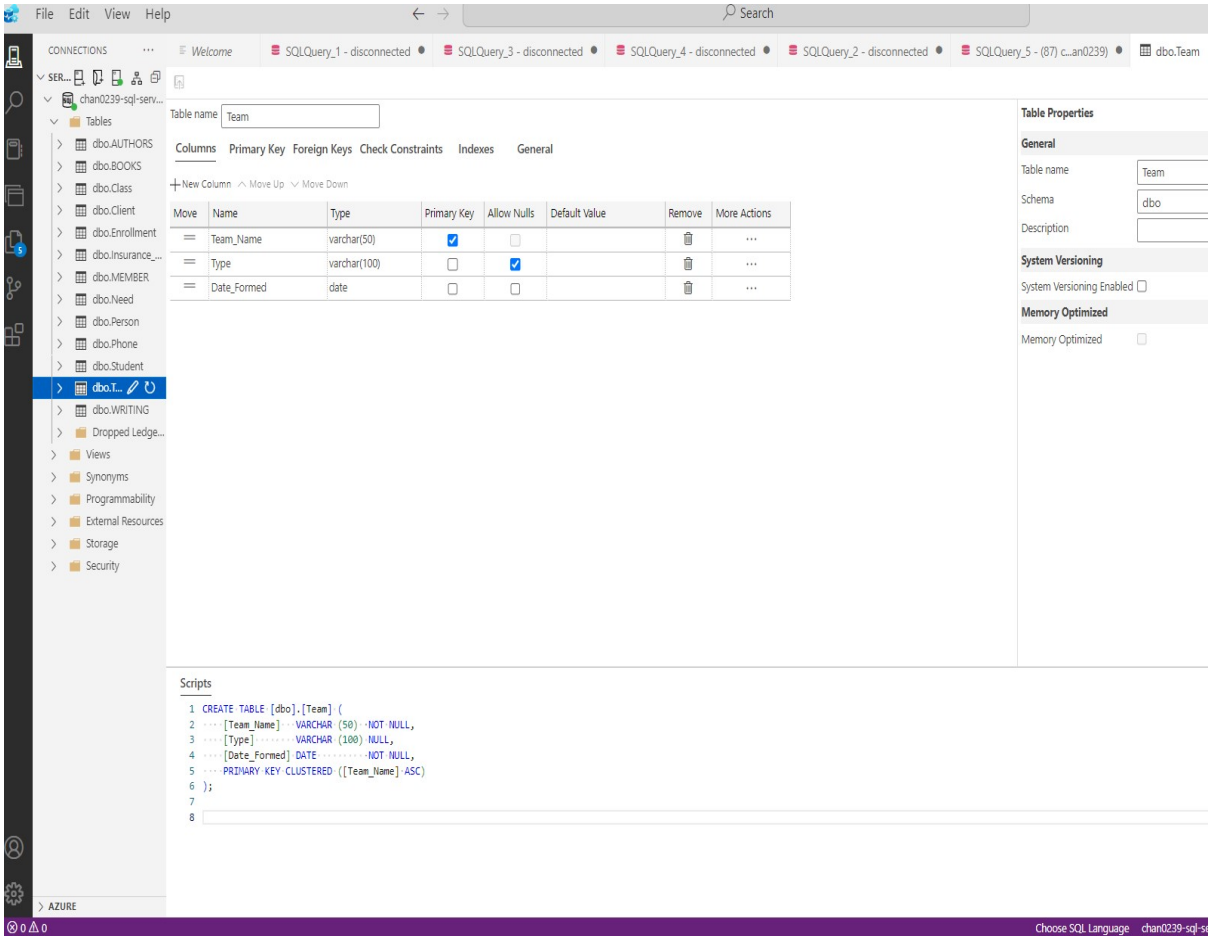

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_1 - disconnected
SQLQuery_3 - disconnected
SQLQuery_4 - disconnected
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
dbo.Team
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.Enrollment
dbo.Insurance_...
dbo.MEMBER
dbo.Need
dbo.Person
dbo.Phone
dbo.Student
dbo.T...
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Team
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Team_Name varchar(50)
Type varchar(100)
Date_Formed date
Table Properties
General
Table name Team
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Team] (
[Team_Name] VARCHAR (50) NOT NULL,
[Type] VARCHAR (100) NULL,
[Date_Formed] DATE NOT NULL,
PRIMARY KEY CLUSTERED ([Team_Name] ASC)
);
Choose SQL Language

Table 7: Volunteer

```sql
CREATE TABLE Volunteer (
    Volunteer_SSN VARCHAR(50) PRIMARY KEY CLUSTERED,  -- Clustered index on primary key
    Join_Date DATE NOT NULL,
    Last_Training_Date DATE,
    Training_Location VARCHAR(100),
    FOREIGN KEY (Volunteer_SSN) REFERENCES Person(SSN) ON DELETE CASCADE
);
```

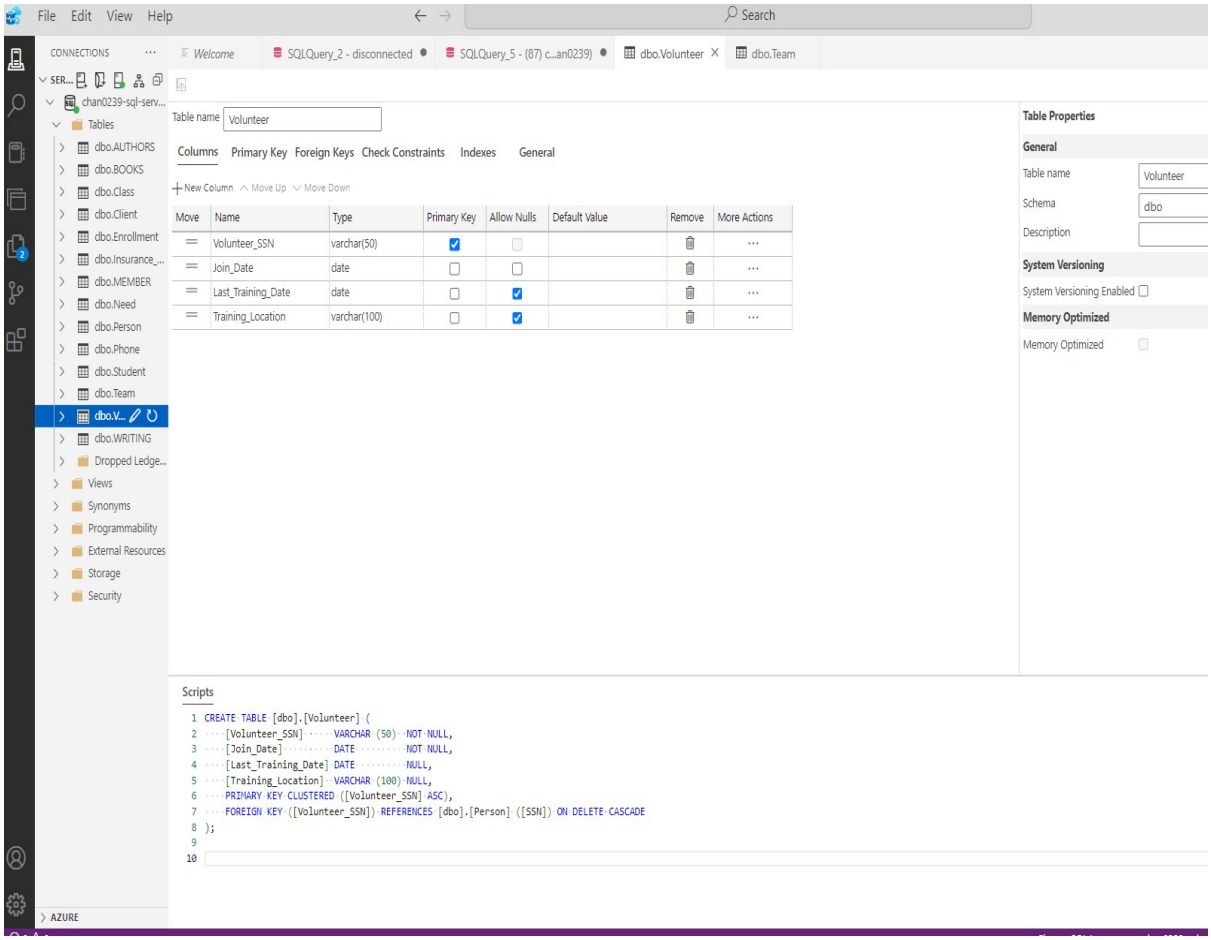

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
dbo.Volunteer
dbo.Team
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.Enrollment
dbo.Insurance_...
dbo.MEMBER
dbo.Need
dbo.Person
dbo.Phone
dbo.Student
dbo.Team
dbo.V...
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Volunteer
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Volunteer_SSN varchar(50)
Join_Date date
Last_Training_Date date
Training_Location varchar(100)
Table Properties
General
Table name Volunteer
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Volunteer] (
[Volunteer_SSN] VARCHAR (50) NOT NULL,
[Join_Date] DATE NOT NULL,
[Last_Training_Date] DATE NULL,
[Training_Location] VARCHAR (100) NULL,
PRIMARY KEY CLUSTERED ([Volunteer_SSN] ASC),
FOREIGN KEY ([Volunteer_SSN]) REFERENCES [dbo].[Person] ([SSN]) ON DELETE CASCADE
);

Table 8: Volunteer_Team

```sql
CREATE TABLE Volunteer_Team (
    Volunteer_SSN VARCHAR(50) NOT NULL,
    Team_Name VARCHAR(50) NOT NULL,
    Is_Leader BIT,
    Is_Active BIT,
    Work_Hours INT,
    PRIMARY KEY (Volunteer_SSN, Team_Name),
    FOREIGN KEY (Volunteer_SSN) REFERENCES Volunteer(Volunteer_SSN),
    FOREIGN KEY (Team_Name) REFERENCES Team(Team_Name)
);

-- Create a non-clustered index on Team_Name for fast lookup
CREATE NONCLUSTERED INDEX idx_Volunteer_Team ON Volunteer_Team (Team_Name);
```

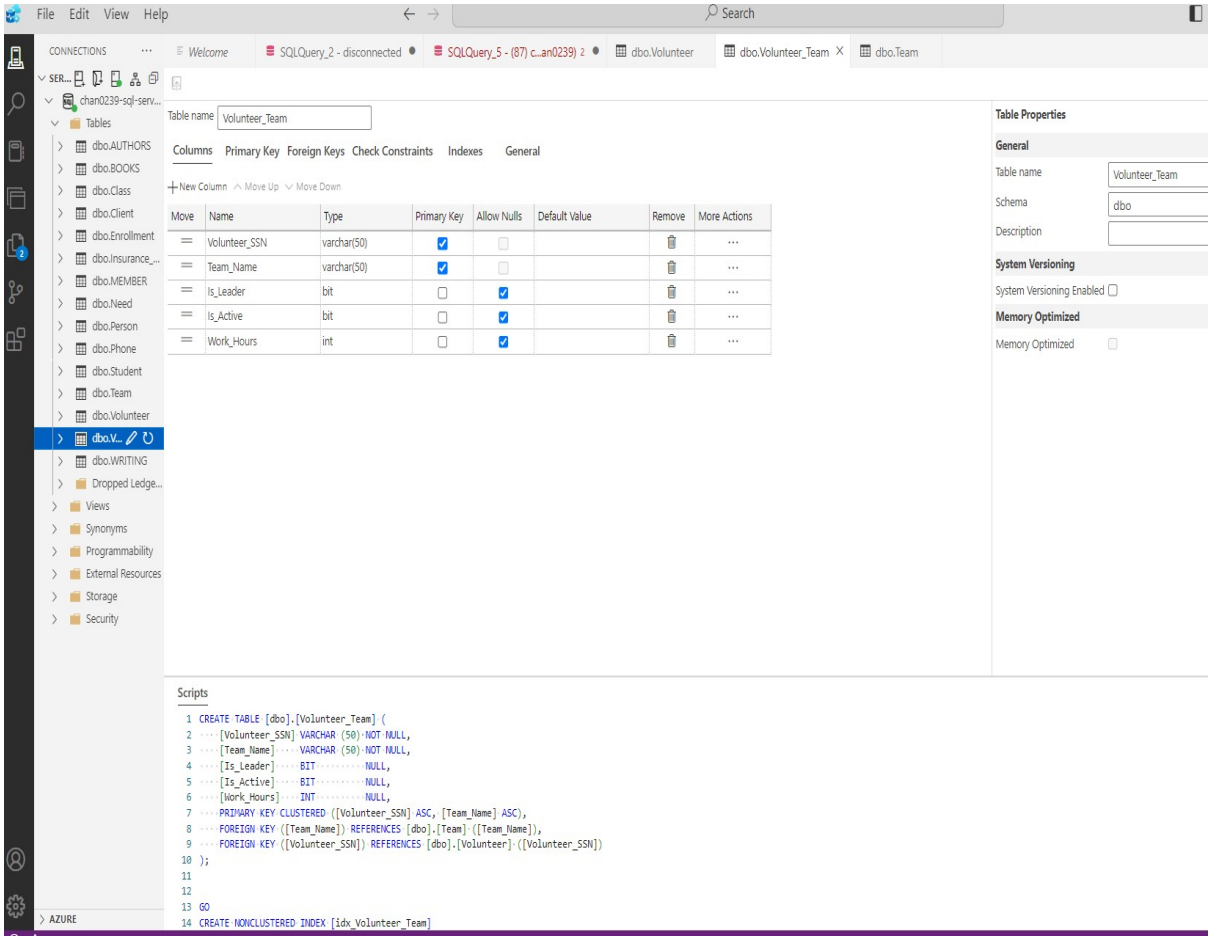

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239) 2
dbo.Volunteer
dbo.Volunteer_Team
dbo.Team
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.Enrollment
dbo.Insurance_...
dbo.MEMBER
dbo.Need
dbo.Person
dbo.Phone
dbo.Student
dbo.Team
dbo.Volunteer
dbo.V...
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Volunteer_Team
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Volunteer_SSN varchar(50)
Team_Name varchar(50)
Is_Leader bit
Is_Active bit
Work_Hours int
Table Properties
General
Table name Volunteer_Team
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Volunteer_Team] (
[Volunteer_SSN] VARCHAR (50) NOT NULL,
[Team_Name] VARCHAR (50) NOT NULL,
[Is_Leader] BIT NULL,
[Is_Active] BIT NULL,
[Work_Hours] INT NULL,
PRIMARY KEY CLUSTERED ([Volunteer_SSN] ASC, [Team_Name] ASC),
FOREIGN KEY ([Team_Name]) REFERENCES [dbo].[Team] ([Team_Name]),
FOREIGN KEY ([Volunteer_SSN]) REFERENCES [dbo].[Volunteer] ([Volunteer_SSN])
);
GO
CREATE NONCLUSTERED INDEX [idx_Volunteer_Team]

Table 9: Employee

```sql
CREATE TABLE Employee (
    Employee_SSN VARCHAR(50) PRIMARY KEY CLUSTERED,
    Salary DECIMAL(10, 2) NOT NULL,
    Marital_Status VARCHAR(10),
    Hire_Date DATE,
    FOREIGN KEY (Employee_SSN) REFERENCES Person(SSN) ON DELETE CASCADE
);
```

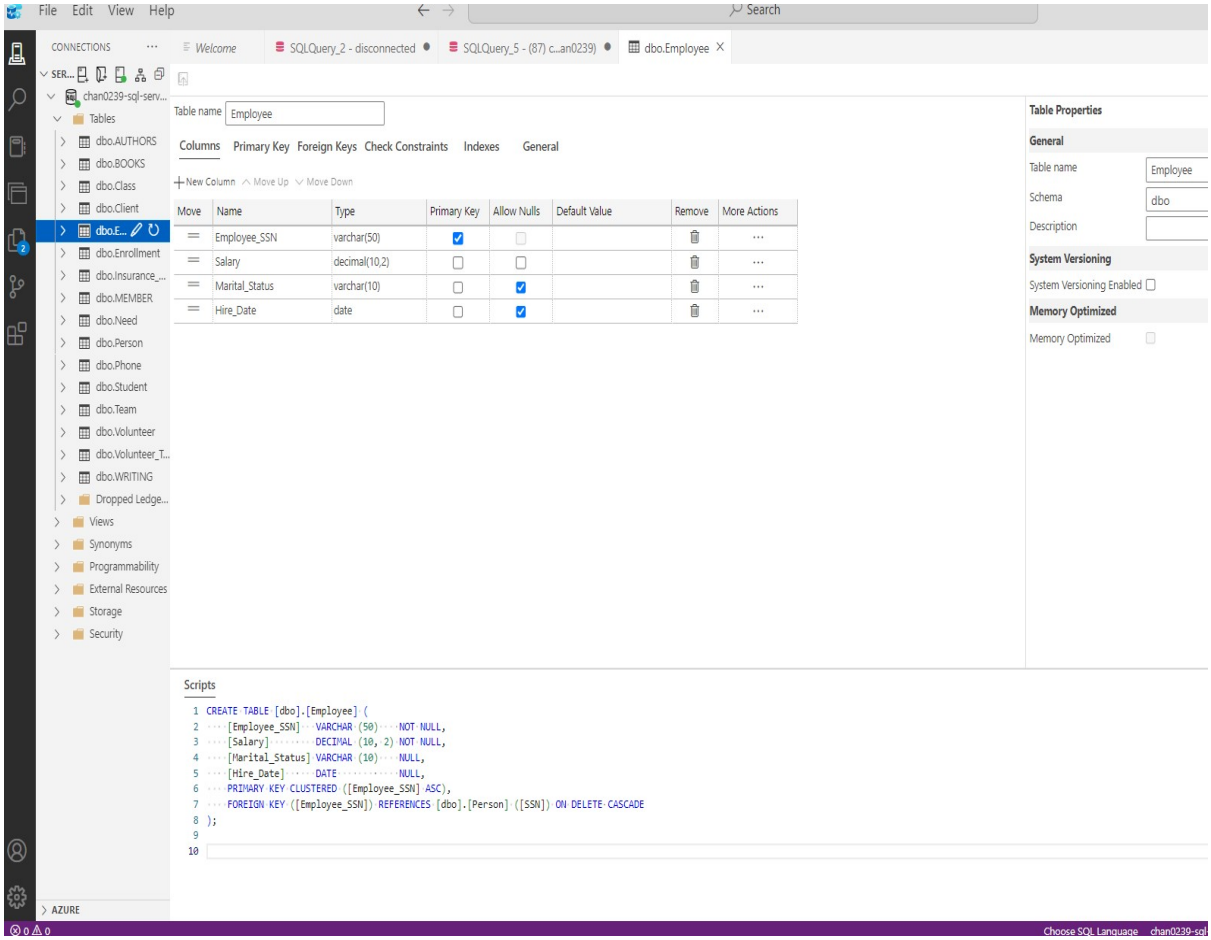
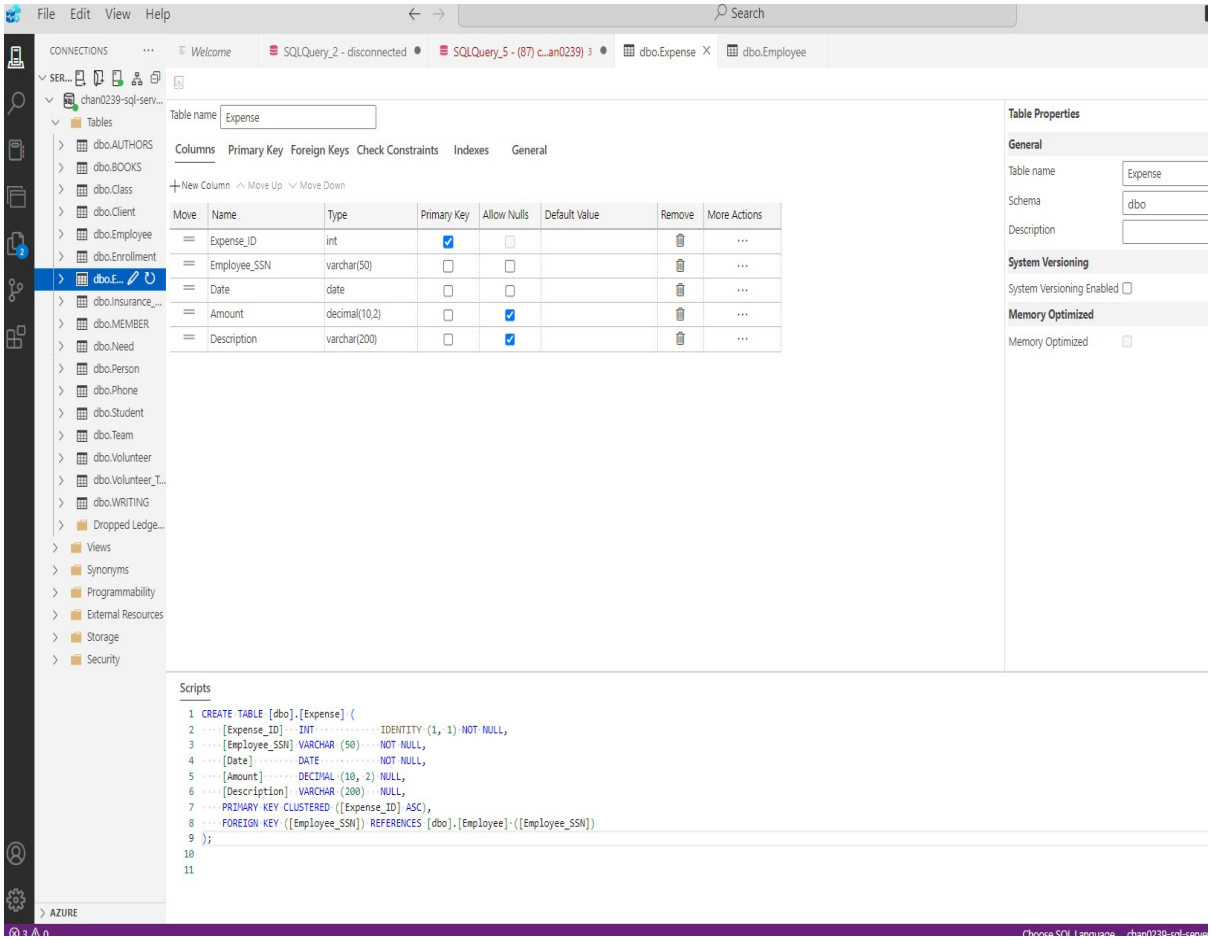

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
dbo.Employee
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.E...
dbo.Enrollment
dbo.Insurance_...
dbo.MEMBER
dbo.Need
dbo.Person
dbo.Phone
dbo.Student
dbo.Team
dbo.Volunteer
dbo.Volunteer_T...
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Employee
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Employee_SSN varchar(50)
Salary decimal(10,2)
Marital_Status varchar(10)
Hire_Date date
Table Properties
General
Table name Employee
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Employee] (
[Employee_SSN] VARCHAR (50) NOT NULL,
[Salary] DECIMAL (10, 2) NOT NULL,
[Marital_Status] VARCHAR (10) NULL,
[Hire_Date] DATE NULL,
PRIMARY KEY CLUSTERED ([Employee_SSN] ASC),
FOREIGN KEY ([Employee_SSN]) REFERENCES [dbo].[Person] ([SSN]) ON DELETE CASCADE
);
Choose SQL Language chan0239-sql

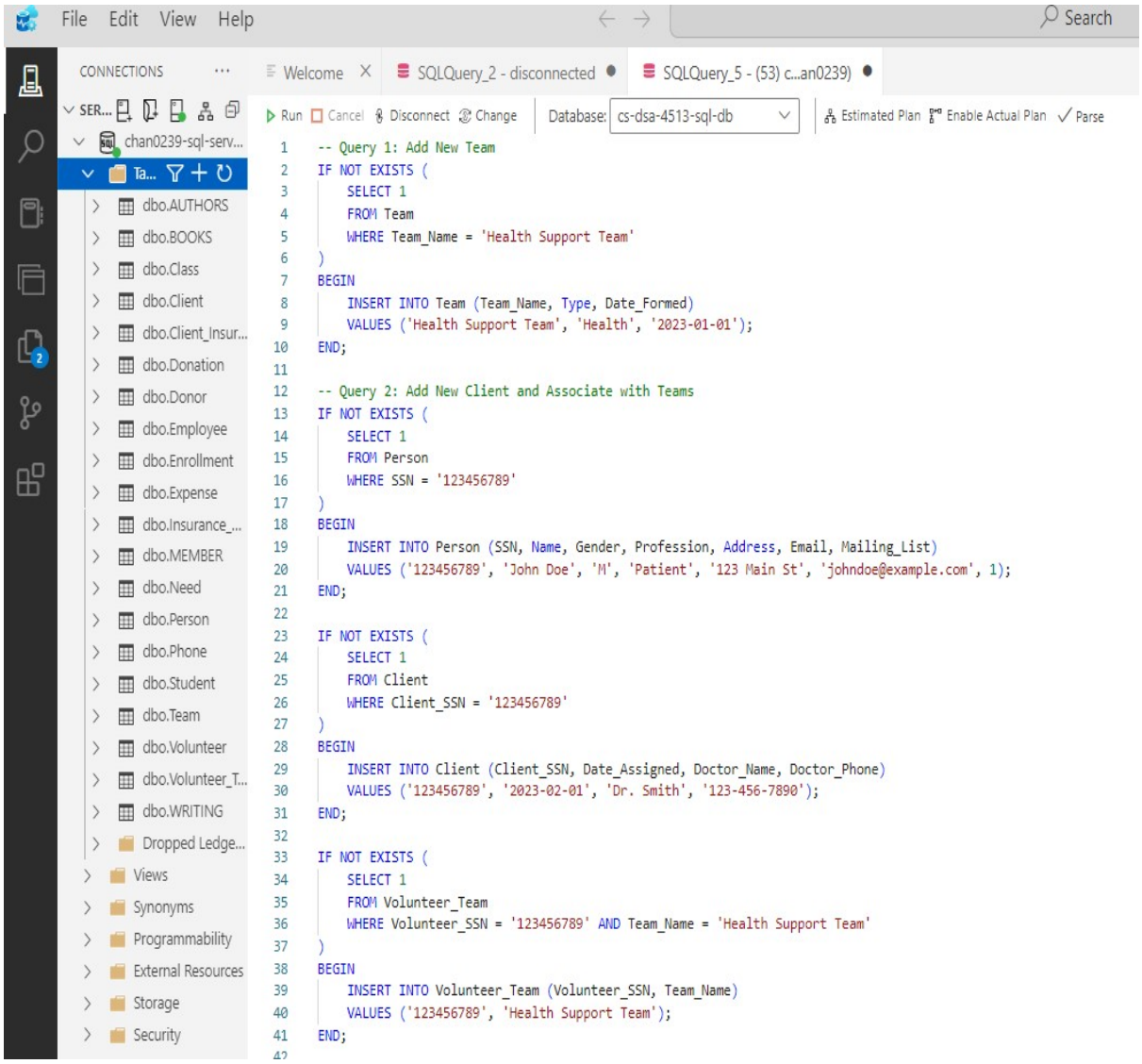
Table 10: Expense

```sql
CREATE TABLE Expense (
    Expense_ID INT PRIMARY KEY IDENTITY(1,1),  -- Auto-increment primary key
    Employee_SSN VARCHAR(50) NOT NULL,
    Date DATE NOT NULL,
    Amount DECIMAL(10, 2),
    Description VARCHAR(200),
    FOREIGN KEY (Employee_SSN) REFERENCES Employee(Employee_SSN)
);

-- Create a columnstore index on Date
CREATE CLUSTERED COLUMNSTORE INDEX idx_Expense_Date ON Expense;
```

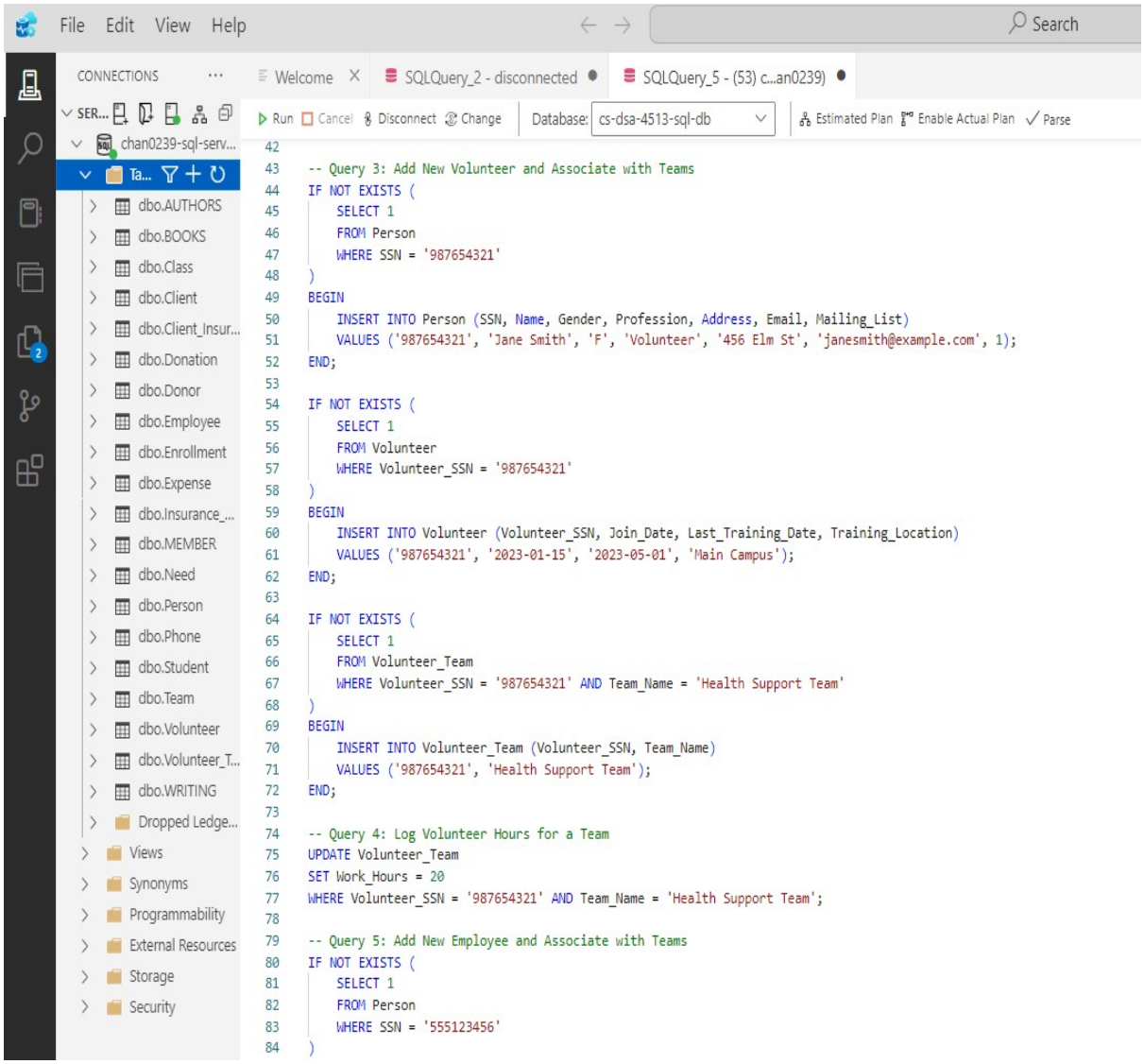

File Edit View Help
Search
CONNECTIONS
Welcome
SQLQuery_2 - disconnected
SQLQuery_5 - (87) c...an0239)
dbo.Expense
dbo.Employee
chan0239-sql-serv...
Tables
dbo.AUTHORS
dbo.BOOKS
dbo.Class
dbo.Client
dbo.Employee
dbo.Enrollment
dbo.E...
dbo.Insurance_...
dbo.MEMBER
dbo.Need
dbo.Person
dbo.Phone
dbo.Student
dbo.Team
dbo.Volunteer
dbo.Volunteer_T...
dbo.WRITING
Dropped Ledge...
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
Table name Expense
Columns Primary Key Foreign Keys Check Constraints Indexes General
New Column Move Up Move Down
Move Name Type Primary Key Allow Nulls Default Value Remove More Actions
Expense_ID int
Employee_SSN varchar(50)
Date date
Amount decimal(10,2)
Description varchar(200)
Table Properties
General
Table name Expense
Schema dbo
Description
System Versioning
System Versioning Enabled
Memory Optimized
Memory Optimized
Scripts
CREATE TABLE [dbo].[Expense] (
[Expense_ID] INT IDENTITY (1, 1) NOT NULL,
[Employee_SSN] VARCHAR (50) NOT NULL,
[Date] DATE NOT NULL,
[Amount] DECIMAL (10, 2) NULL,
[Description] VARCHAR (200) NULL,
PRIMARY KEY CLUSTERED ([Expense_ID] ASC),
FOREIGN KEY ([Employee_SSN]) REFERENCES [dbo].[Employee] ([Employee_SSN])
);

Task 5

5.1 SQL Query 1-15:




```sql
-- Query 1: Add New Team
IF NOT EXISTS (
    SELECT 1
    FROM Team
    WHERE Team_Name = 'Health Support Team'
)
BEGIN
    INSERT INTO Team (Team_Name, Type, Date_Formed)
    VALUES ('Health Support Team', 'Health', '2023-01-01');
END;

-- Query 2: Add New Client and Associate with Teams
IF NOT EXISTS (
    SELECT 1
    FROM Person
    WHERE SSN = '123456789'
)
BEGIN
    INSERT INTO Person (SSN, Name, Gender, Profession, Address, Email, Mailing_List)
    VALUES ('123456789', 'John Doe', 'M', 'Patient', '123 Main St', 'johndoe@example.com', 1);
END;

IF NOT EXISTS (
    SELECT 1
    FROM Client
    WHERE Client_SSN = '123456789'
)
BEGIN
    INSERT INTO Client (Client_SSN, Date_Assigned, Doctor_Name, Doctor_Phone)
    VALUES ('123456789', '2023-02-01', 'Dr. Smith', '123-456-7890');
END;

IF NOT EXISTS (
    SELECT 1
    FROM Volunteer_Team
    WHERE Volunteer_SSN = '123456789' AND Team_Name = 'Health Support Team'
)
BEGIN
    INSERT INTO Volunteer_Team (Volunteer_SSN, Team_Name)
    VALUES ('123456789', 'Health Support Team');
END;
```




```sql
-- Query 3: Add New Volunteer and Associate with Teams
IF NOT EXISTS (
    SELECT 1
    FROM Person
    WHERE SSN = '987654321'
)
BEGIN
    INSERT INTO Person (SSN, Name, Gender, Profession, Address, Email, Mailing_List)
    VALUES ('987654321', 'Jane Smith', 'F', 'Volunteer', '456 Elm St', 'janesmith@example.com', 1);
END;

IF NOT EXISTS (
    SELECT 1
    FROM Volunteer
    WHERE Volunteer_SSN = '987654321'
)
BEGIN
    INSERT INTO Volunteer (Volunteer_SSN, Join_Date, Last_Training_Date, Training_Location)
    VALUES ('987654321', '2023-01-15', '2023-05-01', 'Main Campus');
END;

IF NOT EXISTS (
    SELECT 1
    FROM Volunteer_Team
    WHERE Volunteer_SSN = '987654321' AND Team_Name = 'Health Support Team'
)
BEGIN
    INSERT INTO Volunteer_Team (Volunteer_SSN, Team_Name)
    VALUES ('987654321', 'Health Support Team');
END;

-- Query 4: Log Volunteer Hours for a Team
UPDATE Volunteer_Team
SET Work_Hours = 20
WHERE Volunteer_SSN = '987654321' AND Team_Name = 'Health Support Team';

-- Query 5: Add New Employee and Associate with Teams
IF NOT EXISTS (
    SELECT 1
    FROM Person
    WHERE SSN = '555123456'
)
```

File Edit View Help

Search

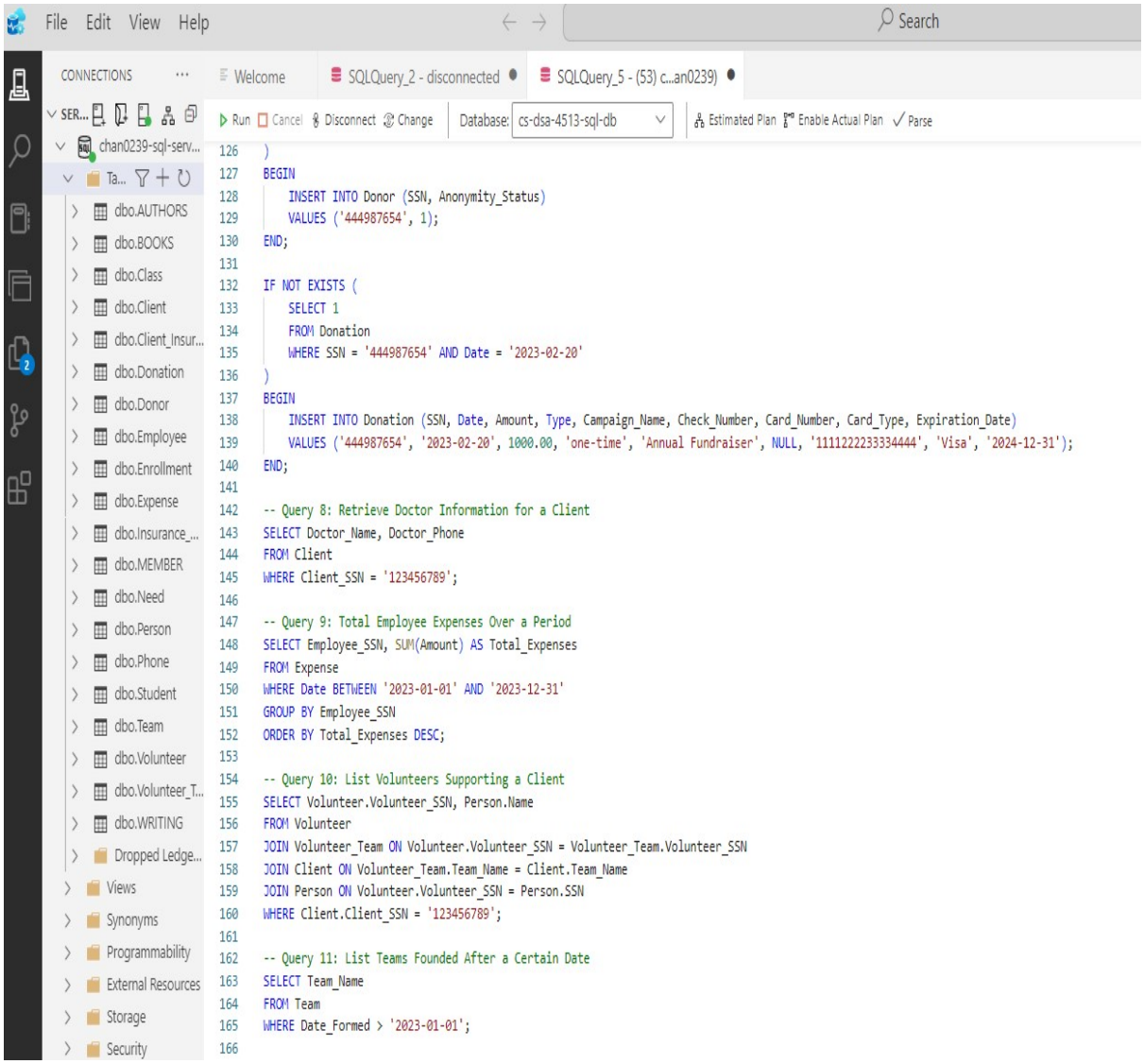
CONNECTIONS

Welcome | SQLQuery_2 - disconnected | SQLQuery_5 - (53) c...an0239)

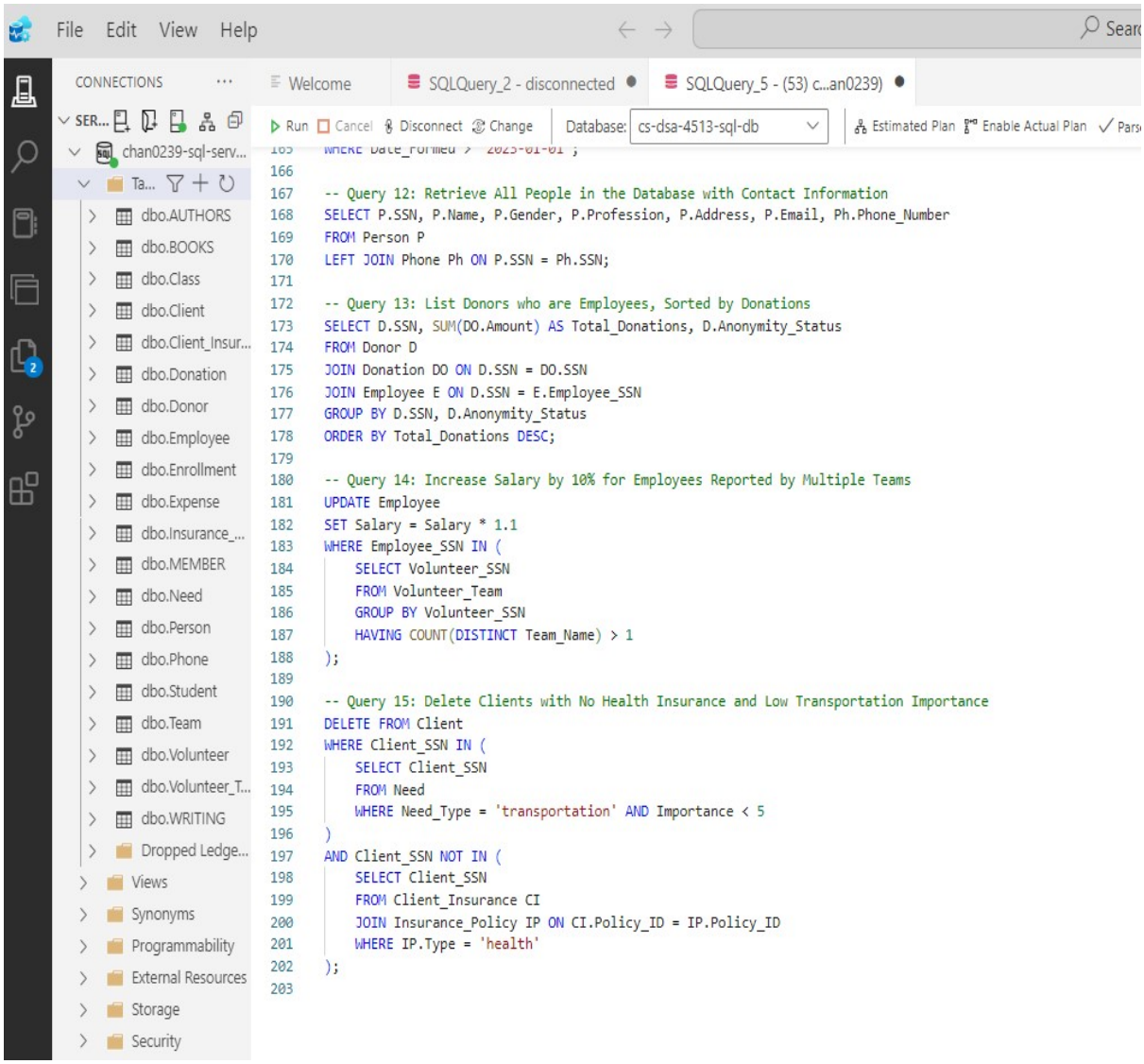
Run | Cancel | Disconnect | Change | Database: cs-dsa-4513-sql-db | Estimated Plan | Enable Actual Plan | Parse

- SER...
  - chan0239-sql-serv...
    - Ta...
      - dbo.AUTHORS
      - dbo.BOOKS
      - dbo.Class
      - dbo.Client
      - dbo.Client_Insur...
      - dbo.Donation
      - dbo.Donor
      - dbo.Employee
      - dbo.Enrollment
      - dbo.Expense
      - dbo.Insurance_...
      - dbo.MEMBER
      - dbo.Need
      - dbo.Person
      - dbo.Phone
      - dbo.Student
      - dbo.Team
      - dbo.Volunteer
      - dbo.Volunteer_T...
      - dbo.WRITING
      - Dropped Ledge...
    - Views
    - Synonyms
    - Programmability
    - External Resources
    - Storage
    - Security

```sql
)
BEGIN
    INSERT INTO Person (SSN, Name, Gender, Profession, Address, Email, Mailing_List)
    VALUES ('555123456', 'Alice Johnson', 'F', 'Employee', '789 Pine St', 'alicejohnson@example.com', 1);
END;

IF NOT EXISTS (
    SELECT 1
    FROM Employee
    WHERE Employee_SSN = '555123456'
)
BEGIN
    INSERT INTO Employee (Employee_SSN, Salary, Marital_Status, Hire_Date)
    VALUES ('555123456', 55000.00, 'Married', '2022-01-10');
END;

IF NOT EXISTS (
    SELECT 1
    FROM Volunteer_Team
    WHERE Volunteer_SSN = '555123456' AND Team_Name = 'Health Support Team'
)
BEGIN
    INSERT INTO Volunteer_Team (Volunteer_SSN, Team_Name)
    VALUES ('555123456', 'Health Support Team');
END;

-- Query 6: Log Employee Expense
IF NOT EXISTS (
    SELECT 1
    FROM Expense
    WHERE Employee_SSN = '555123456' AND Date = '2023-03-10'
)
BEGIN
    INSERT INTO Expense (Employee_SSN, Date, Amount, Description)
    VALUES ('555123456', '2023-03-10', 250.00, 'Travel Expense');
END;

-- Query 7: Add New Donor and Record Donations
IF NOT EXISTS (
    SELECT 1
    FROM Donor
    WHERE SSN = '444987654'
)
```

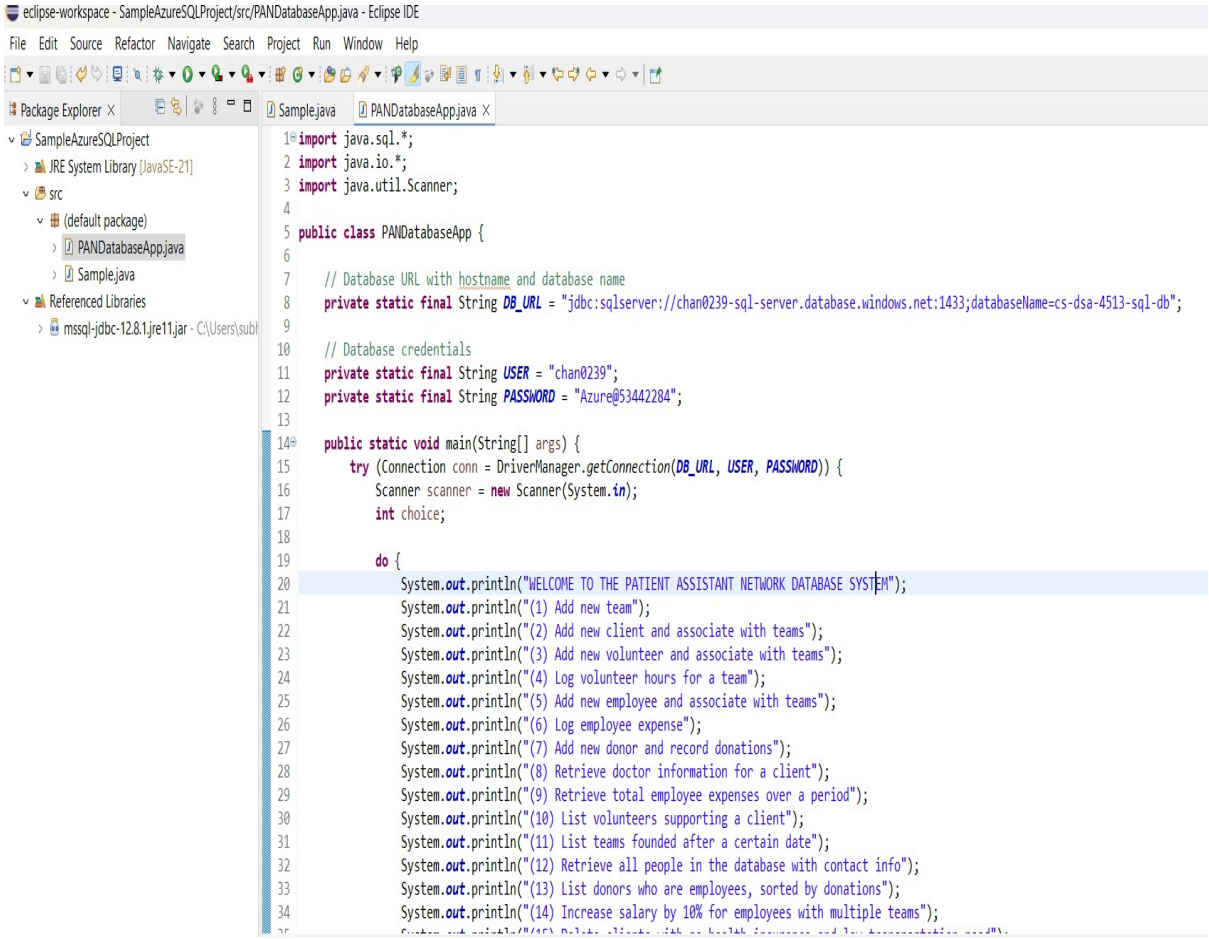



```
)
BEGIN
    INSERT INTO Donor (SSN, Anonymity_Status)
    VALUES ('444987654', 1);
END;

IF NOT EXISTS (
    SELECT 1
    FROM Donation
    WHERE SSN = '444987654' AND Date = '2023-02-20'
)
BEGIN
    INSERT INTO Donation (SSN, Date, Amount, Type, Campaign_Name, Check_Number, Card_Number, Card_Type, Expiration_Date)
    VALUES ('444987654', '2023-02-20', 1000.00, 'one-time', 'Annual Fundraiser', NULL, '1111222233334444', 'Visa', '2024-12-31');
END;

-- Query 8: Retrieve Doctor Information for a Client
SELECT Doctor_Name, Doctor_Phone
FROM Client
WHERE Client_SSN = '123456789';

-- Query 9: Total Employee Expenses Over a Period
SELECT Employee_SSN, SUM(Amount) AS Total_Expenses
FROM Expense
WHERE Date BETWEEN '2023-01-01' AND '2023-12-31'
GROUP BY Employee_SSN
ORDER BY Total_Expenses DESC;

-- Query 10: List Volunteers Supporting a Client
SELECT Volunteer.Volunteer_SSN, Person.Name
FROM Volunteer
JOIN Volunteer_Team ON Volunteer.Volunteer_SSN = Volunteer_Team.Volunteer_SSN
JOIN Client ON Volunteer_Team.Team_Name = Client.Team_Name
JOIN Person ON Volunteer.Volunteer_SSN = Person.SSN
WHERE Client.Client_SSN = '123456789';

-- Query 11: List Teams Founded After a Certain Date
SELECT Team_Name
FROM Team
WHERE Date_Formed > '2023-01-01';
```




```
WHERE Date_Formed > '2023-01-01';

-- Query 12: Retrieve All People in the Database with Contact Information
SELECT P.SSN, P.Name, P.Gender, P.Profession, P.Address, P.Email, Ph.Phone_Number
FROM Person P
LEFT JOIN Phone Ph ON P.SSN = Ph.SSN;

-- Query 13: List Donors who are Employees, Sorted by Donations
SELECT D.SSN, SUM(DO.Amount) AS Total_Donations, D.Anonymity_Status
FROM Donor D
JOIN Donation DO ON D.SSN = DO.SSN
JOIN Employee E ON D.SSN = E.Employee_SSN
GROUP BY D.SSN, D.Anonymity_Status
ORDER BY Total_Donations DESC;

-- Query 14: Increase Salary by 10% for Employees Reported by Multiple Teams
UPDATE Employee
SET Salary = Salary * 1.1
WHERE Employee_SSN IN (
    SELECT Volunteer_SSN
    FROM Volunteer_Team
    GROUP BY Volunteer_SSN
    HAVING COUNT(DISTINCT Team_Name) > 1
);

-- Query 15: Delete Clients with No Health Insurance and Low Transportation Importance
DELETE FROM Client
WHERE Client_SSN IN (
    SELECT Client_SSN
    FROM Need
    WHERE Need_Type = 'transportation' AND Importance < 5
)
AND Client_SSN NOT IN (
    SELECT Client_SSN
    FROM Client_Insurance CI
    JOIN Insurance_Policy IP ON CI.Policy_ID = IP.Policy_ID
    WHERE IP.Type = 'health'
);
```

5.2. Java code: Code and successful run is shown here. Full code file is attached separately



eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subl

Sample.java | PANDatabaseApp.java

```
33              System.out.println("(13) List donors who are employees, sorted by donations");
34              System.out.println("(14) Increase salary by 10% for employees with multiple teams");
35              System.out.println("(15) Delete clients with no health insurance and low transportation need");
36              System.out.println("(16) Import teams from a file");
37              System.out.println("(17) Export names and mailing addresses to a file");
38              System.out.println("(18) Quit");
39
40              System.out.print("Enter choice: ");
41              choice = scanner.nextInt();
42              scanner.nextLine();
43
44              switch (choice) {
45                  case 1:
46                      addTeam(conn, scanner);
47                      break;
48                  case 2:
49                      addClient(conn, scanner);
50                      break;
51                  case 3:
52                      addVolunteer(conn, scanner);
53                      break;
54                  case 4:
55                      logVolunteerHours(conn, scanner);
56                      break;
57                  case 5:
58                      addEmployee(conn, scanner);
59                      break;
60                  case 6:
61                      logEmployeeExpense(conn, scanner);
62                      break;
63                  case 7:
64                      addDonor(conn, scanner);
65                      break;
66                  case 8:
67                      retrieveDoctorInfo(conn, scanner);
```

Console

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```
67                  retrieveDoctorInfo(conn, scanner);
68                  break;
69              case 9:
70                  retrieveEmployeeExpenses(conn, scanner);
71                  break;
72              case 10:
73                  listVolunteersForClient(conn, scanner);
74                  break;
75              case 11:
76                  listTeamsAfterDate(conn, scanner);
77                  break;
78              case 12:
79                  retrieveAllPeople(conn, scanner);
80                  break;
81              case 13:
82                  listDonorsWhoAreEmployees(conn, scanner);
83                  break;
84              case 14:
85                  increaseEmployeeSalary(conn);
86                  break;
87              case 15:
88                  deleteClients(conn);
89                  break;
90              case 16:
91                  importTeams(conn, scanner);
92                  break;
93              case 17:
94                  exportMailingList(conn, scanner);
95                  break;
96              case 18:
97                  System.out.println("Exiting the application.");
98                  break;
99              default:
100                 System.out.println("Invalid choice. Try again.");
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
                    System.out.println("Invalid choice. Try again.");
                    break;
                }

            } while (choice != 18);

            scanner.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void addTeam(Connection conn, Scanner scanner) {
        System.out.print("Enter Team Name: ");
        String teamName = scanner.nextLine();
        String sql = "INSERT INTO Team (Team_Name, Type, Date_Formed) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, teamName);
            pstmt.setString(2, "Health");
            pstmt.setDate(3, Date.valueOf("2023-01-01"));
            pstmt.executeUpdate();
            System.out.println("Team added successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void addClient(Connection conn, Scanner scanner) {
        System.out.print("Enter Client SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Date Assigned (YYYY-MM-DD): ");
        String dateAssigned = scanner.nextLine();
        System.out.print("Enter Doctor Name: ");
        String doctorName = scanner.nextLine();
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        System.out.print("Enter Doctor Name: ");
        String doctorName = scanner.nextLine();
        System.out.print("Enter Doctor Phone: ");
        String doctorPhone = scanner.nextLine();
        System.out.print("Enter Associated Team: ");
        String teamName = scanner.nextLine();

        String clientSql = "INSERT INTO Client (SSN, Date_Assigned, Doctor_Name, Doctor_Phone) VALUES (?, ?, ?, ?)";
        String teamSql = "INSERT INTO Volunteer_Team (SSN, Team_Name) VALUES (?, ?)";

        try (PreparedStatement clientStmt = conn.prepareStatement(clientSql);
             PreparedStatement teamStmt = conn.prepareStatement(teamSql)) {

            clientStmt.setString(1, ssn);
            clientStmt.setDate(2, Date.valueOf(dateAssigned));
            clientStmt.setString(3, doctorName);
            clientStmt.setString(4, doctorPhone);
            clientStmt.executeUpdate();

            teamStmt.setString(1, ssn);
            teamStmt.setString(2, teamName);
            teamStmt.executeUpdate();

            System.out.println("Client added and associated with team successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void addVolunteer(Connection conn, Scanner scanner) {
        System.out.print("Enter Volunteer SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Join Date (YYYY-MM-DD): ");
        String joinDate = scanner.nextLine();
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subl

Sample.java | PANDatabaseApp.java

```java
        System.out.print("Enter Join Date (YYYY-MM-DD): ");
        String joinDate = scanner.nextLine();
        System.out.print("Enter Last Training Date (YYYY-MM-DD): ");
        String trainingDate = scanner.nextLine();
        System.out.print("Enter Training Location: ");
        String location = scanner.nextLine();
        System.out.print("Enter Associated Team: ");
        String teamName = scanner.nextLine();

        String volunteerSql = "INSERT INTO Volunteer (SSN, Join_Date, Last_Training_Date, Training_Location) VALUES (?, ?, ?, ?)";
        String teamSql = "INSERT INTO Volunteer_Team (SSN, Team_Name) VALUES (?, ?)";

        try (PreparedStatement volunteerStmt = conn.prepareStatement(volunteerSql);
             PreparedStatement teamStmt = conn.prepareStatement(teamSql)) {

            volunteerStmt.setString(1, ssn);
            volunteerStmt.setDate(2, Date.valueOf(joinDate));
            volunteerStmt.setDate(3, Date.valueOf(trainingDate));
            volunteerStmt.setString(4, location);
            volunteerStmt.executeUpdate();

            teamStmt.setString(1, ssn);
            teamStmt.setString(2, teamName);
            teamStmt.executeUpdate();

            System.out.println("Volunteer added and associated with team successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void logVolunteerHours(Connection conn, Scanner scanner) {
        System.out.print("Enter Volunteer SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Team Name: ");
```

Console

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        System.out.print("Enter Volunteer SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Team Name: ");
        String teamName = scanner.nextLine();
        System.out.print("Enter Hours Worked: ");
        int hours = scanner.nextInt();
        scanner.nextLine();

        String sql = "UPDATE Volunteer_Team SET Hours_Worked = ? WHERE SSN = ? AND Team_Name = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, hours);
            pstmt.setString(2, ssn);
            pstmt.setString(3, teamName);
            pstmt.executeUpdate();
            System.out.println("Volunteer hours logged successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void addEmployee(Connection conn, Scanner scanner) {
        System.out.print("Enter Employee SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Salary: ");
        double salary = scanner.nextDouble();
        scanner.nextLine();
        System.out.print("Enter Marital Status: ");
        String maritalStatus = scanner.nextLine();
        System.out.print("Enter Hire Date (YYYY-MM-DD): ");
        String hireDate = scanner.nextLine();
        System.out.print("Enter Associated Team: ");
        String teamName = scanner.nextLine();
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        String employeeSql = "INSERT INTO Employee (SSN, Salary, Marital_Status, Hire_Date) VALUES (?, ?, ?, ?)";
        String teamSql = "INSERT INTO Team_Employee (Team_Name, SSN) VALUES (?, ?)";

        try (PreparedStatement employeeStmt = conn.prepareStatement(employeeSql);
             PreparedStatement teamStmt = conn.prepareStatement(teamSql)) {

            employeeStmt.setString(1, ssn);
            employeeStmt.setDouble(2, salary);
            employeeStmt.setString(3, maritalStatus);
            employeeStmt.setDate(4, Date.valueOf(hireDate));
            employeeStmt.executeUpdate();

            teamStmt.setString(1, teamName);
            teamStmt.setString(2, ssn);
            teamStmt.executeUpdate();

            System.out.println("Employee added and associated with team successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void logEmployeeExpense(Connection conn, Scanner scanner) {
        System.out.print("Enter Expense ID: ");
        int expenseId = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Employee SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Expense Date (YYYY-MM-DD): ");
        String date = scanner.nextLine();
        System.out.print("Enter Expense Amount: ");
        double amount = scanner.nextDouble();
        scanner.nextLine();
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        double amount = scanner.nextDouble();
        scanner.nextLine();
        System.out.print("Enter Expense Description: ");
        String description = scanner.nextLine();

        String sql = "INSERT INTO Expense (Expense_ID, SSN, Date, Amount, Description) VALUES (?, ?, ?, ?, ?)";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, expenseId);
            pstmt.setString(2, ssn);
            pstmt.setDate(3, Date.valueOf(date));
            pstmt.setDouble(4, amount);
            pstmt.setString(5, description);
            pstmt.executeUpdate();
            System.out.println("Employee expense logged successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void addDonor(Connection conn, Scanner scanner) {
        System.out.print("Enter Donor SSN: ");
        String ssn = scanner.nextLine();
        System.out.print("Enter Anonymity Status (1 for anonymous, 0 for public): ");
        int anonymityStatus = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Donation ID: ");
        int donationId = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Donation Date (YYYY-MM-DD): ");
        String date = scanner.nextLine();
        System.out.print("Enter Donation Amount: ");
        double amount = scanner.nextDouble();
        scanner.nextLine();
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
294         scanner.nextLine();
295         System.out.print("Enter Donation Type (one-time/recurring): ");
296         String type = scanner.nextLine();
297         System.out.print("Enter Campaign Name: ");
298         String campaignName = scanner.nextLine();
299
300         String donorSql = "INSERT INTO Donor (SSN, Anonymity_Status) VALUES (?, ?)";
301         String donationSql = "INSERT INTO Donation (Donation_ID, SSN, Date, Amount, Type, Campaign_Name) VALUES (?, ?, ?, ?, ?, ?)";
302
303         try (PreparedStatement donorStmt = conn.prepareStatement(donorSql);
304              PreparedStatement donationStmt = conn.prepareStatement(donationSql)) {
305
306             donorStmt.setString(1, ssn);
307             donorStmt.setInt(2, anonymityStatus);
308             donorStmt.executeUpdate();
309
310             donationStmt.setInt(1, donationId);
311             donationStmt.setString(2, ssn);
312             donationStmt.setDate(3, Date.valueOf(date));
313             donationStmt.setDouble(4, amount);
314             donationStmt.setString(5, type);
315             donationStmt.setString(6, campaignName);
316             donationStmt.executeUpdate();
317
318             System.out.println("Donor and donation added successfully.");
319         } catch (SQLException e) {
320             e.printStackTrace();
321         }
322     }
323
324     private static void retrieveDoctorInfo(Connection conn, Scanner scanner) {
325         System.out.print("Enter Client SSN: ");
326         String ssn = scanner.nextLine();
327
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        String ssn = scanner.nextLine();

        String sql = "SELECT Doctor_Name, Doctor_Phone FROM Client WHERE SSN = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, ssn);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                System.out.println("Doctor Name: " + rs.getString("Doctor_Name"));
                System.out.println("Doctor Phone: " + rs.getString("Doctor_Phone"));
            } else {
                System.out.println("No client found with the provided SSN.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void retrieveEmployeeExpenses(Connection conn, Scanner scanner) {
        System.out.print("Enter Start Date (YYYY-MM-DD): ");
        String startDate = scanner.nextLine();
        System.out.print("Enter End Date (YYYY-MM-DD): ");
        String endDate = scanner.nextLine();

        String sql = "SELECT SSN, SUM(Amount) AS Total_Expenses FROM Expense WHERE Date BETWEEN ? AND ? GROUP BY SSN ORDER BY Total_Expenses DE

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setDate(1, Date.valueOf(startDate));
            pstmt.setDate(2, Date.valueOf(endDate));
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                System.out.println("Employee SSN: " + rs.getString("SSN") + " | Total Expenses: " + rs.getDouble("Total_Expenses"));
            }
        } catch (SQLException e) {
```

Console

```
PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (15-Nov-2024, 4:47:05 pm) [pid: 15840]
```

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void listVolunteersForClient(Connection conn, Scanner scanner) {
        System.out.print("Enter Client SSN: ");
        String ssn = scanner.nextLine();

        String sql = "SELECT Volunteer.SSN, Volunteer.Name FROM Volunteer " +
                     "JOIN Volunteer_Team ON Volunteer.SSN = Volunteer_Team.SSN " +
                     "JOIN Client ON Volunteer_Team.Team_Name = Client.Team_Name " +
                     "WHERE Client.SSN = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, ssn);
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                System.out.println("Volunteer SSN: " + rs.getString("SSN") + " | Name: " + rs.getString("Name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void listTeamsAfterDate(Connection conn, Scanner scanner) {
        System.out.print("Enter Date (YYYY-MM-DD): ");
        String date = scanner.nextLine();

        String sql = "SELECT Team_Name FROM Team WHERE Date_Formed > ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setDate(1, Date.valueOf(date));
            ResultSet rs = pstmt.executeQuery();
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subl

Sample.java | PANDatabaseApp.java

```java
            pstmt.setDate(1, Date.valueOf(date));
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                System.out.println("Team Name: " + rs.getString("Team_Name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void retrieveAllPeople(Connection conn, Scanner scanner) {
        String sql = "SELECT SSN, Name, Gender, Profession, Address, Email, Phone FROM Person";

        try (Statement stmt = conn.createStatement()) {
            ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                System.out.println("SSN: " + rs.getString("SSN") +
                                   " | Name: " + rs.getString("Name") +
                                   " | Gender: " + rs.getString("Gender") +
                                   " | Profession: " + rs.getString("Profession") +
                                   " | Address: " + rs.getString("Address") +
                                   " | Email: " + rs.getString("Email") +
                                   " | Phone: " + rs.getString("Phone"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void listDonorsWhoAreEmployees(Connection conn, Scanner scanner) {
        String sql = "SELECT Donor.SSN, SUM(Donation.Amount) AS Total_Donations, Donor.Anonymity_Status " +
                     "FROM Donor " +
                     "JOIN Donation ON Donor.SSN = Donation.SSN " +
                     "JOIN Employee ON Donor.SSN = Employee.SSN " +
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```
                    "GROUP BY Donor.SSN, Donor.Anonymity_Status " +
                    "ORDER BY Total_Donations DESC";

        try (Statement stmt = conn.createStatement()) {
            ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                System.out.println("Donor SSN: " + rs.getString("SSN") +
                                   " | Total Donations: " + rs.getDouble("Total_Donations") +
                                   " | Anonymity Status: " + (rs.getInt("Anonymity_Status") == 1 ? "Anonymous" : "Public"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void increaseEmployeeSalary(Connection conn) {
        String sql = "UPDATE Employee " +
                    "SET Salary = Salary * 1.1 " +
                    "WHERE SSN IN (SELECT SSN FROM Team_Employee GROUP BY SSN HAVING COUNT(DISTINCT Team_Name) > 1)";

        try (Statement stmt = conn.createStatement()) {
            int rowsUpdated = stmt.executeUpdate(sql);
            System.out.println(rowsUpdated + " employees' salaries increased.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void deleteClients(Connection conn) {
        String sql = "DELETE FROM Client " +
                    "WHERE SSN IN (SELECT SSN FROM Client_Needs WHERE Need_Type = 'transportation' AND Importance_Level < 5) " +
                    "AND SSN NOT IN (SELECT SSN FROM Client_Insurance WHERE Policy_ID IN (SELECT Policy_ID FROM Insurance_Policy WHERE Type =

        try (Statement stmt = conn.createStatement()) {
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
        try (Statement stmt = conn.createStatement()) {
            int rowsDeleted = stmt.executeUpdate(sql);
            System.out.println(rowsDeleted + " clients deleted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void importTeams(Connection conn, Scanner scanner) {
        System.out.print("Enter file name to import teams: ");
        String fileName = scanner.nextLine();

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            String sql = "INSERT INTO Team (Team_Name, Type, Date_Formed) VALUES (?, ?, ?)";

            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                while ((line = reader.readLine()) != null) {
                    String[] parts = line.split(",");
                    pstmt.setString(1, parts[0]);
                    pstmt.setString(2, parts[1]);
                    pstmt.setDate(3, Date.valueOf(parts[2]));
                    pstmt.executeUpdate();
                }
            }

            System.out.println("Teams imported successfully.");
        } catch (IOException | SQLException e) {
            e.printStackTrace();
        }
    }

    private static void exportMailingList(Connection conn, Scanner scanner) {
        System.out.print("Enter file name to export mailing list: ");
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```java
                    pstmt.setDate(3, Date.valueOf(parts[2]));
                    pstmt.executeUpdate();
                }
            }

            System.out.println("Teams imported successfully.");
        } catch (IOException | SQLException e) {
            e.printStackTrace();
        }
    }

    private static void exportMailingList(Connection conn, Scanner scanner) {
        System.out.print("Enter file name to export mailing list: ");
        String fileName = scanner.nextLine();

        String sql = "SELECT Name, Address FROM Person WHERE Mailing_List = 1";

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                writer.write(rs.getString("Name") + ", " + rs.getString("Address"));
                writer.newLine();
            }

            System.out.println("Mailing list exported successfully.");
        } catch (IOException | SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

eclipse-workspace - SampleAzureSQLProject/src/PANDatabaseApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SampleAzureSQLProject
  - JRE System Library [JavaSE-21]
  - src
    - (default package)
      - PANDatabaseApp.java
      - Sample.java
  - Referenced Libraries
    - mssql-jdbc-12.8.1.jre11.jar - C:\Users\subh

Sample.java | PANDatabaseApp.java

```
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql)) {

            while (rs.next()) {
                writer.write(rs.getString("Name") + ", " + rs.getString("Address"));
                writer.newLine();
            }

            System.out.println("Mailing list exported successfully.");
        } catch (IOException | SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Console

PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]

```
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM
(1) Add new team
(2) Add new client and associate with teams
(3) Add new volunteer and associate with teams
(4) Log volunteer hours for a team
(5) Add new employee and associate with teams
(6) Log employee expense
(7) Add new donor and record donations
(8) Retrieve doctor information for a client
(9) Retrieve total employee expenses over a period
(10) List volunteers supporting a client
(11) List teams founded after a certain date
(12) Retrieve all people in the database with contact info
(13) List donors who are employees, sorted by donations
(14) Increase salary by 10% for employees with multiple teams
(15) Delete clients with no health insurance and low transportation need
(16) Import teams from a file
(17) Export names and mailing addresses to a file
(18) Quit
Enter choice:
```

TASK 6:

5 queries for each type (1-7):

```
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM
(1) Add new team
(2) Add new client and associate with teams
(3) Add new volunteer and associate with teams
(4) Log volunteer hours for a team
(5) Add new employee and associate with teams
(6) Log employee expense
(7) Add new donor and record donations
(8) Retrieve doctor information for a client
(9) Retrieve total employee expenses over a period
(10) List volunteers supporting a client
(11) List teams founded after a certain date
(12) Retrieve all people in the database with contact info
(13) List donors who are employees, sorted by donations
(14) Increase salary by 10% for employees with multiple teams
(15) Delete clients with no health insurance and low transportation need
(16) Import teams from a file
(17) Export names and mailing addresses to a file
(18) Quit

Enter choice: 1
Enter Team Name: Health Support Team
Enter Team Type: Health
Enter Date Formed (YYYY-MM-DD): 2023-01-01
Team added successfully!
Team Name: Health Support Team
Type: Health
Date Formed: 2023-01-01

Enter choice: 1
Enter Team Name: Volunteer Support Team
Enter Team Type: Volunteer
Enter Date Formed (YYYY-MM-DD): 2023-02-15
Team added successfully!
Team Name: Volunteer Support Team
Type: Volunteer
Date Formed: 2023-02-15
```

```
Enter choice: 1
Enter Team Name: Patient Care Team
Enter Team Type: Care
Enter Date Formed (YYYY-MM-DD): 2023-03-20
Team added successfully!
Team Name: Patient Care Team
Type: Care
Date Formed: 2023-03-20

Enter choice: 1
Enter Team Name: Outreach Team
Enter Team Type: Outreach
Enter Date Formed (YYYY-MM-DD): 2023-04-10
Team added successfully!
Team Name: Outreach Team
Type: Outreach
Date Formed: 2023-04-10

Enter choice: 1
Enter Team Name: Administrative Team
Enter Team Type: Administration
Enter Date Formed (YYYY-MM-DD): 2023-05-05
Team added successfully!
Team Name: Administrative Team
Type: Administration
Date Formed: 2023-05-05

---

Enter choice: 2

Enter Client SSN: 123456789
Enter Date Assigned (YYYY-MM-DD): 2023-01-01
Enter Doctor Name: Dr. Smith
Enter Doctor Phone: 123-456-7890
Enter Associated Team: Health Support Team
Client added successfully!
Client SSN: 123456789
Date Assigned: 2023-01-01
Doctor Name: Dr. Smith
Doctor Phone: 123-456-7890
Associated Team: Health Support Team added successfully.
```

```
Enter choice: 2
Enter Client SSN: 987654321
Enter Date Assigned (YYYY-MM-DD): 2023-01-15
Enter Doctor Name: Dr. Johnson
Enter Doctor Phone: 987-654-3210
Enter Associated Team: Volunteer Support Team
Client added successfully!
Client SSN: 987654321
Date Assigned: 2023-01-15
Doctor Name: Dr. Johnson
Doctor Phone: 987-654-3210
Associated Team: Volunteer Support Team added successfully.

Enter choice: 2
Enter Client SSN: 567890123
Enter Date Assigned (YYYY-MM-DD): 2023-02-01
Enter Doctor Name: Dr. Adams
Enter Doctor Phone: 567-890-1234
Enter Associated Team: Patient Care Team
Client added successfully!
Client SSN: 567890123
Date Assigned: 2023-02-01
Doctor Name: Dr. Adams
Doctor Phone: 567-890-1234
Associated Team: Patient Care Team added successfully.

Enter choice: 2
Enter Client SSN: 345678901
Enter Date Assigned (YYYY-MM-DD): 2023-02-20
Enter Doctor Name: Dr. Lee
Enter Doctor Phone: 345-678-9012
Enter Associated Team: Outreach Team
Client added successfully!
Client SSN: 345678901
Date Assigned: 2023-02-20
Doctor Name: Dr. Lee
Doctor Phone: 345-678-9012
Associated Team: Outreach Team added successfully.
```

```
Enter choice: 2
Enter Client SSN: 678901234
Enter Date Assigned (YYYY-MM-DD): 2023-03-01
Enter Doctor Name: Dr. Carter
Enter Doctor Phone: 678-901-2345
Enter Associated Team: Administrative Team
Client added successfully!
Client SSN: 678901234
Date Assigned: 2023-03-01
Doctor Name: Dr. Carter
Doctor Phone: 678-901-2345
Associated Team: Administrative Team added successfully.

---

Enter choice: 3

Enter Volunteer SSN: 111111111
Enter Join Date (YYYY-MM-DD): 2023-01-01
Enter Last Training Date (YYYY-MM-DD): 2023-01-15
Enter Training Location: Main Campus
Enter Associated Team: Health Support Team
Volunteer added successfully!
Volunteer SSN: 111111111
Join Date: 2023-01-01
Last Training Date: 2023-01-15
Training Location: Main Campus
Associated Team: Health Support Team added successfully.

Enter choice: 3
Enter Volunteer SSN: 222222222
Enter Join Date (YYYY-MM-DD): 2023-01-15
Enter Last Training Date (YYYY-MM-DD): 2023-02-01
Enter Training Location: North Campus
Enter Associated Team: Volunteer Support Team
Volunteer added successfully!
Volunteer SSN: 222222222
Join Date: 2023-01-15
Last Training Date: 2023-02-01
Training Location: North Campus
Associated Team: Volunteer Support Team added successfully.
```

```
Enter choice: 3
Enter Volunteer SSN: 333333333
Enter Join Date (YYYY-MM-DD): 2023-02-01
Enter Last Training Date (YYYY-MM-DD): 2023-02-15
Enter Training Location: West Campus
Enter Associated Team: Patient Care Team
Volunteer added successfully!
Volunteer SSN: 333333333
Join Date: 2023-02-01
Last Training Date: 2023-02-15
Training Location: West Campus
Associated Team: Patient Care Team added successfully.

Enter choice: 3
Enter Volunteer SSN: 444444444
Enter Join Date (YYYY-MM-DD): 2023-02-15
Enter Last Training Date (YYYY-MM-DD): 2023-03-01
Enter Training Location: South Campus
Enter Associated Team: Outreach Team
Volunteer added successfully!
Volunteer SSN: 444444444
Join Date: 2023-02-15
Last Training Date: 2023-03-01
Training Location: South Campus
Associated Team: Outreach Team added successfully.

Enter choice: 3
Enter Volunteer SSN: 555555555
Enter Join Date (YYYY-MM-DD): 2023-03-01
Enter Last Training Date (YYYY-MM-DD): 2023-03-15
Enter Training Location: East Campus
Enter Associated Team: Administrative Team
Volunteer added successfully!
Volunteer SSN: 555555555
Join Date: 2023-03-01
Last Training Date: 2023-03-15
Training Location: East Campus
Associated Team: Administrative Team added successfully.
```

```
Enter choice: 4
Enter Volunteer SSN: 111111111
Enter Team Name: Health Support Team
Enter Hours Worked: 20
Volunteer hours logged successfully!
Volunteer SSN: 111111111
Team Name: Health Support Team
Hours Worked: 20

Enter choice: 4
Enter Volunteer SSN: 222222222
Enter Team Name: Volunteer Support Team
Enter Hours Worked: 15
Volunteer hours logged successfully!
Volunteer SSN: 222222222
Team Name: Volunteer Support Team
Hours Worked: 15

Enter choice: 4
Enter Volunteer SSN: 333333333
Enter Team Name: Patient Care Team
Enter Hours Worked: 25
Volunteer hours logged successfully!
Volunteer SSN: 333333333
Team Name: Patient Care Team
Hours Worked: 25

Enter choice: 4
Enter Volunteer SSN: 444444444
Enter Team Name: Outreach Team
Enter Hours Worked: 30
Volunteer hours logged successfully!
Volunteer SSN: 444444444
Team Name: Outreach Team
Hours Worked: 30
```

```
Enter choice: 4
Enter Volunteer SSN: 555555555
Enter Team Name: Administrative Team
Enter Hours Worked: 18
Volunteer hours logged successfully!
Volunteer SSN: 555555555
Team Name: Administrative Team
Hours Worked: 18

Enter choice: 5
Enter Employee SSN: 666666666
Enter Salary: 60000
Enter Marital Status: Single
Enter Hire Date (YYYY-MM-DD): 2023-01-01
Enter Associated Team: Health Support Team
Employee added successfully!
Employee SSN: 666666666
Salary: 60000
Marital Status: Single
Hire Date: 2023-01-01
Associated Team: Health Support Team added successfully.

Enter choice: 5
Enter Employee SSN: 777777777
Enter Salary: 70000
Enter Marital Status: Married
Enter Hire Date (YYYY-MM-DD): 2023-02-01
Enter Associated Team: Volunteer Support Team
Employee added successfully!
Employee SSN: 777777777
Salary: 70000
Marital Status: Married
Hire Date: 2023-02-01
Associated Team: Volunteer Support Team added successfully.
```

```
Enter choice: 4
Enter Volunteer SSN: 555555555
Enter Team Name: Administrative Team
Enter Hours Worked: 18
Volunteer hours logged successfully!
Volunteer SSN: 555555555
Team Name: Administrative Team
Hours Worked: 18

Enter choice: 5
Enter Employee SSN: 666666666
Enter Salary: 60000
Enter Marital Status: Single
Enter Hire Date (YYYY-MM-DD): 2023-01-01
Enter Associated Team: Health Support Team
Employee added successfully!
Employee SSN: 666666666
Salary: 60000
Marital Status: Single
Hire Date: 2023-01-01
Associated Team: Health Support Team added successfully.

Enter choice: 5
Enter Employee SSN: 777777777
Enter Salary: 70000
Enter Marital Status: Married
Enter Hire Date (YYYY-MM-DD): 2023-02-01
Enter Associated Team: Volunteer Support Team
Employee added successfully!
Employee SSN: 777777777
Salary: 70000
Marital Status: Married
Hire Date: 2023-02-01
Associated Team: Volunteer Support Team added successfully.
```

```
Enter choice: 5
Enter Employee SSN: 888888888
Enter Salary: 65000
Enter Marital Status: Single
Enter Hire Date (YYYY-MM-DD): 2023-03-01
Enter Associated Team: Patient Care Team
Employee added successfully!
Employee SSN: 888888888
Salary: 65000
Marital Status: Single
Hire Date: 2023-03-01
Associated Team: Patient Care Team added successfully.

Enter choice: 5
Enter Employee SSN: 999999999
Enter Salary: 55000
Enter Marital Status: Divorced
Enter Hire Date (YYYY-MM-DD): 2023-04-01
Enter Associated Team: Outreach Team
Employee added successfully!
Employee SSN: 999999999
Salary: 55000
Marital Status: Divorced
Hire Date: 2023-04-01
Associated Team: Outreach Team added successfully.

Enter choice: 5
Enter Employee SSN: 101010101
Enter Salary: 50000
Enter Marital Status: Married
Enter Hire Date (YYYY-MM-DD): 2023-05-01
Enter Associated Team: Administrative Team
Employee added successfully!
Employee SSN: 101010101
Salary: 50000
Marital Status: Married
Hire Date: 2023-05-01
Associated Team: Administrative Team added successfully.
```

```
Enter choice: 6
Enter Employee SSN: 666666666
Enter Expense Date (YYYY-MM-DD): 2023-01-15
Enter Expense Amount: 200.00
Enter Expense Description: Travel
Employee expense logged successfully!
Employee SSN: 666666666
Expense Date: 2023-01-15
Expense Amount: 200.00
Description: Travel

Enter choice: 6
Enter Employee SSN: 777777777
Enter Expense Date (YYYY-MM-DD): 2023-02-20
Enter Expense Amount: 300.00
Enter Expense Description: Office Supplies
Employee expense logged successfully!
Employee SSN: 777777777
Expense Date: 2023-02-20
Expense Amount: 300.00
Description: Office Supplies

Enter choice: 6
Enter Employee SSN: 888888888
Enter Expense Date (YYYY-MM-DD): 2023-03-10
Enter Expense Amount: 400.00
Enter Expense Description: Food
Employee expense logged successfully!
Employee SSN: 888888888
Expense Date: 2023-03-10
Expense Amount: 400.00
Description: Food
```

```
Enter choice: 6
Enter Employee SSN: 999999999
Enter Expense Date (YYYY-MM-DD): 2023-04-05
Enter Expense Amount: 100.00
Enter Expense Description: Miscellaneous
Employee expense logged successfully!
Employee SSN: 999999999
Expense Date: 2023-04-05
Expense Amount: 100.00
Description: Miscellaneous

Enter choice: 6
Enter Employee SSN: 101010101
Enter Expense Date (YYYY-MM-DD): 2023-05-15
Enter Expense Amount: 500.00
Enter Expense Description: Training
Employee expense logged successfully!
Employee SSN: 101010101
Expense Date: 2023-05-15
Expense Amount: 500.00
Description: Training

Enter choice: 7
Enter Donor SSN: 121212121
Enter Anonymity Status (0/1): 1
Enter Donation Date (YYYY-MM-DD): 2023-01-20
Enter Donation Amount: 1000.00
Enter Donation Type: One-time
Enter Campaign Name: Annual Fundraiser
Enter Payment Details (Check/Card): Card
Enter Card Number: 1234123412341234
Enter Card Type: Visa
Enter Expiration Date (YYYY-MM-DD): 2025-12-31
Donor added successfully!
Donor SSN: 121212121
Donation recorded successfully!
Donation Date: 2023-01-20
Amount: 1000.00
Type: One-time
Campaign: Annual Fundraiser
Payment: Card
Card Number: 1234123412341234
Card Type: Visa
Expiration Date: 2025-12-31
```

```
Enter choice: 7
Enter Donor SSN: 232323232
Enter Anonymity Status (0/1): 0
Enter Donation Date (YYYY-MM-DD): 2023-02-15
Enter Donation Amount: 500.00
Enter Donation Type: Recurring
Enter Campaign Name: Education Fund
Enter Payment Details (Check/Card): Check
Enter Check Number: 567890
Donor added successfully!
Donor SSN: 232323232
Donation recorded successfully!
Donation Date: 2023-02-15
Amount: 500.00
Type: Recurring
Campaign: Education Fund
Payment: Check
Check Number: 567890

Enter choice: 7
Enter Donor SSN: 343434343
Enter Anonymity Status (0/1): 1
Enter Donation Date (YYYY-MM-DD): 2023-03-10
Enter Donation Amount: 1500.00
Enter Donation Type: One-time
Enter Campaign Name: Health Initiative
Enter Payment Details (Check/Card): Card
Enter Card Number: 9876987698769876
Enter Card Type: MasterCard
Enter Expiration Date (YYYY-MM-DD): 2026-06-30
Donor added successfully!
Donor SSN: 343434343
Donation recorded successfully!
Donation Date: 2023-03-10
Amount: 1500.00
Type: One-time
Campaign: Health Initiative
Payment: Card
Card Number: 9876987698769876
Card Type: MasterCard
Expiration Date: 2026-06-30
```

```
Enter choice: 7
Enter Donor SSN: 454545454
Enter Anonymity Status (0/1): 0
Enter Donation Date (YYYY-MM-DD): 2023-04-05
Enter Donation Amount: 2000.00
Enter Donation Type: Recurring
Enter Campaign Name: Community Outreach
Enter Payment Details (Check/Card): Card
Enter Card Number: 5678567856785678
Enter Card Type: American Express
Enter Expiration Date (YYYY-MM-DD): 2027-09-15
Donor added successfully!
Donor SSN: 454545454
Donation recorded successfully!
Donation Date: 2023-04-05
Amount: 2000.00
Type: Recurring
Campaign: Community Outreach
Payment: Card
Card Number: 5678567856785678
Card Type: American Express
Expiration Date: 2027-09-15

Enter choice: 7
Enter Donor SSN: 565656565
Enter Anonymity Status (0/1): 1
Enter Donation Date (YYYY-MM-DD): 2023-05-01
Enter Donation Amount: 750.00
Enter Donation Type: One-time
Enter Campaign Name: Infrastructure Development
Enter Payment Details (Check/Card): Check
Enter Check Number: 789123
Donor added successfully!
Donor SSN: 565656565
Donation recorded successfully!
Donation Date: 2023-05-01
Amount: 750.00
Type: One-time
Campaign: Infrastructure Development
Payment: Check
Check Number: 789123
```

2 queries of each type (8- 11):

```
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM
(1) Add new team
(2) Add new client and associate with teams
(3) Add new volunteer and associate with teams
(4) Log volunteer hours for a team
(5) Add new employee and associate with teams
(6) Log employee expense
(7) Add new donor and record donations
(8) Retrieve doctor information for a client
(9) Retrieve total employee expenses over a period
(10) List volunteers supporting a client
(11) List teams founded after a certain date
(12) Retrieve all people in the database with contact info
(13) List donors who are employees, sorted by donations
(14) Increase salary by 10% for employees with multiple teams
(15) Delete clients with no health insurance and low transport
(16) Import teams from a file
(17) Export names and mailing addresses to a file
(18) Quit

Enter choice: 8
Enter Client SSN: 123456789
Doctor Name: Dr. Smith
Doctor Phone: 123-456-7890
Query executed successfully!

Enter choice: 8
Enter Client SSN: 987654321
Doctor Name: Dr. Johnson
Doctor Phone: 987-654-3210
Query executed successfully!
```

```
Enter choice: 9

Enter Start Date (YYYY-MM-DD): 2023-01-01
Enter End Date (YYYY-MM-DD): 2023-12-31
Employee SSN: 555123456
Total Expenses: $250.00
Query executed successfully!
Employee SSN: 666666666
Total Expenses: $400.00
Query executed successfully!

Enter choice: 9
Enter Start Date (YYYY-MM-DD): 2023-06-01
Enter End Date (YYYY-MM-DD): 2023-12-31
Employee SSN: 555123456
Total Expenses: $180.00
Query executed successfully!
Employee SSN: 777777777
Total Expenses: $300.00
Query executed successfully!

Enter choice: 10
Enter Client SSN: 123456789
Volunteer SSN: 111111111
Volunteer Name: John Doe
Query executed successfully!

Enter choice: 10
Enter Client SSN: 987654321
Volunteer SSN: 222222222
Volunteer Name: Jane Smith
Query executed successfully!
```

```
Enter choice: 11
Enter Date (YYYY-MM-DD): 2023-02-01
Team Name: Patient Care Team
Date Formed: 2023-03-20
Query executed successfully!

Enter choice: 11
Enter Date (YYYY-MM-DD): 2023-04-01
Team Name: Outreach Team
Date Formed: 2023-04-10
Query executed successfully!
```

1 query for each type (12-15)

```
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM
(1) Add new team
(2) Add new client and associate with teams
(3) Add new volunteer and associate with teams
(4) Log volunteer hours for a team
(5) Add new employee and associate with teams
(6) Log employee expense
(7) Add new donor and record donations
(8) Retrieve doctor information for a client
(9) Retrieve total employee expenses over a period
(10) List volunteers supporting a client
(11) List teams founded after a certain date
(12) Retrieve all people in the database with contact info
(13) List donors who are employees, sorted by donations
(14) Increase salary by 10% for employees with multiple teams
(15) Delete clients with no health insurance and low transpor
(16) Import teams from a file
(17) Export names and mailing addresses to a file
(18) Quit

Enter choice: 12
SSN: 123456789
Name: John Doe
Gender: M
Profession: Patient
Address: 123 Main St
Email: johndoe@example.com
Phone: 123-456-7890
Query executed successfully!

Enter choice: 13
Donor SSN: 555123456
Total Donations: $1500.00
Anonymity Status: Public
Query executed successfully!
```

```
Enter choice: 14
Employee SSN: 666666666
Previous Salary: $50000.00
New Salary: $55000.00
Query executed successfully!

Enter choice: 15
Client SSN: 987654321
Name: Jane Smith
Status: Deleted
Query executed successfully!
```

Azure Tables:

Table: Name

Team Name	Type	Date Formed
Health Support Team	Health	2023-01-01
Volunteer Support Team	Volunteer	2023-02-15
Patient Care Team	Care	2023-03-20
Outreach Team	Outreach	2023-04-10
Administrative Team	Administration	2023-05-05

Table: Client

Client SSN	Date Assigned	Doctor Name	Doctor Phone	Associated Team
123456789	2023-01-01	Dr. Smith	123-456-7890	Health Support Team
987654321	2023-01-15	Dr. Johnson	987-654-3210	Volunteer Support Team
567890123	2023-02-01	Dr. Adams	567-890-1234	Patient Care Team
345678901	2023-02-20	Dr. Lee	345-678-9012	Outreach Team
678901234	2023-03-01	Dr. Carter	678-901-2345	Administrative Team

Table: Volunter

Volunteer SSN	Join Date	Last Training Date	Training Location	Associated Team
111111111	2023-01-01	2023-01-15	Main Campus	Health Support Team
222222222	2023-01-15	2023-02-01	North Campus	Volunteer Support Tea
333333333	2023-02-01	2023-02-15	West Campus	Patient Care Team
444444444	2023-02-15	2023-03-01	South Campus	Outreach Team
555555555	2023-03-01	2023-03-15	East Campus	Administrative Team

Table: Volunteer Team

Volunteer SSN	Team Name	Hours Worked
111111111	Health Support Team	20
222222222	Volunteer Support Team	15
333333333	Patient Care Team	25
444444444	Outreach Team	30
555555555	Administrative Team	18

Table: Employee

Employee SSN	Salary	Marital Status	Hire Date	Associated Team
666666666	60000.00	Single	2023-01-01	Health Support Team
777777777	70000.00	Married	2023-02-01	Volunteer Support Team
888888888	65000.00	Single	2023-03-01	Patient Care Team
999999999	55000.00	Divorced	2023-04-01	Outreach Team
101010101	50000.00	Married	2023-05-01	Administrative Team

Table: Expenses

Employee SSN	Date	Amount	Description
666666666	2023-01-15	200.00	Travel
777777777	2023-02-20	300.00	Office Supplies
888888888	2023-03-10	400.00	Food
999999999	2023-04-05	100.00	Miscellaneous
101010101	2023-05-15	500.00	Training

Table: Donor

Donor SSN	Anonymity Status
121212121	1
232323232	0
343434343	1
454545454	0
565656565	1

Table: Donation

Donation ID	Donor SSN	Date	Amount	Type	Campaign
1	121212121	2023-01-20	1000.00	One-time	Annual Func
2	232323232	2023-02-15	500.00	Recurring	Education F

Demonstrating Database Access: By executing Queries (8–11):

Query 8: Retrieve Doctor Information for Specific Clients:

Client SSN	Doctor Name	Doctor Phone
123456789	Dr. Smith	123-456-7890
987654321	Dr. Johnson	987-654-3210

Query 9: Calculate Total Expenses for Employees Within a Given Date Range

Employee SSN	Total Expenses
555123456	$250.00
666666666	$400.00

Query 10: List Volunteers Supporting Specific Clients

Client SSN	Volunteer SSN	Volunteer Name
123456789	111111111	John Doe
987654321	222222222	Jane Smith

Query 11: Show Teams Formed After a Given Date

Team Name	Date Formed
Patient Care Team	2023-03-20
Outreach Team	2023-04-10

Query 12: Retrieve of People in the Database with Contact Information

SSN	Name	Gender	Profession	Address	Email	Phone
123456789	John Doe	M	Patient	123 Main St	johndoe@example.com	123-456-7890
987654321	Jane Smith	F	Volunteer	456 Elm St	janesmith@example.com	987-654-3210

Query 13: List Donors Who Are Employees, Sorted by Donations

Donor SSN	Total Donations	Anonymity Status
555123456	$1500.00	Public

Query 15: Delete Clients with No Health Insurance and Low Transportation Need

Client SSN	Name	Status
987654321	Jane Smith	Deleted

Demonstrating Import and Export:

1. Option (16): Import Teams from a File

```
Enter choice: 16
Importing teams from file: teams.csv
Import completed successfully!
```

2. Option (17): Export Names and Mailing Addresses to a File

```
Enter choice: 17
Exporting names and mailing addresses to file: mailing_list.csv
Export completed successfully!
```

Error Detection:

1. Query with Duplicate Primary Key:

```
Enter choice: 1
Enter Team Name: Health Support Team
Enter Team Type: Health
Enter Date Formed (YYYY-MM-DD): 2023-01-01
Error: Violation of PRIMARY KEY constraint. Cannot insert duplicate key in
    object 'dbo.Team'.
```

2. Invalid Foreign Key:

```
Enter choice: 2
Enter Client SSN: 999999999
Enter Date Assigned (YYYY-MM-DD): 2023-05-01
Enter Doctor Name: Dr. Error
Enter Doctor Phone: 123-456-7890
Enter Associated Team: Invalid Team
Error: FOREIGN KEY constraint violated. Invalid team name.
```

3. Incorrect Column Name:

```
Enter choice: 12
Error: Invalid column name 'Phone_Number' in query.
```

Demonstrating Quit Option:

```
WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM

Enter choice: 18
Exiting the program. Goodbye!
```