

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	Description	Page Number(s)
Task 1	ER Diagram	2-3
Task 2	Relational Database Schemas	4-14
Task 3	Main Task 3 Content	15-22
	Discussion of storage structures for tables	23-24
	Discussion of storage structures for tables (Azure SQL Database)	25
Task 4	SQL statements and screenshots showing the creation of tables in Azure SQL Database	26-45
Task 5	SQL statements (and Transact SQL stored procedures, if any) Implementing all queries (1-15 and error checking)	46-50
	The Java source program and screenshots showing its successful compilation	51-67
Task 6	Java program Execution	68-91
	Screenshots showing the testing of query 1	68-69
	Screenshots showing the testing of query 2	69-71
	:	:
	Screenshots showing the testing of query 15	88
	Screenshots showing the testing of the import and export options	89
	Screenshots showing the testing of three types of errors	90-91
	Screenshots showing the testing of the quit option	92

Task 1: ER Diagram:

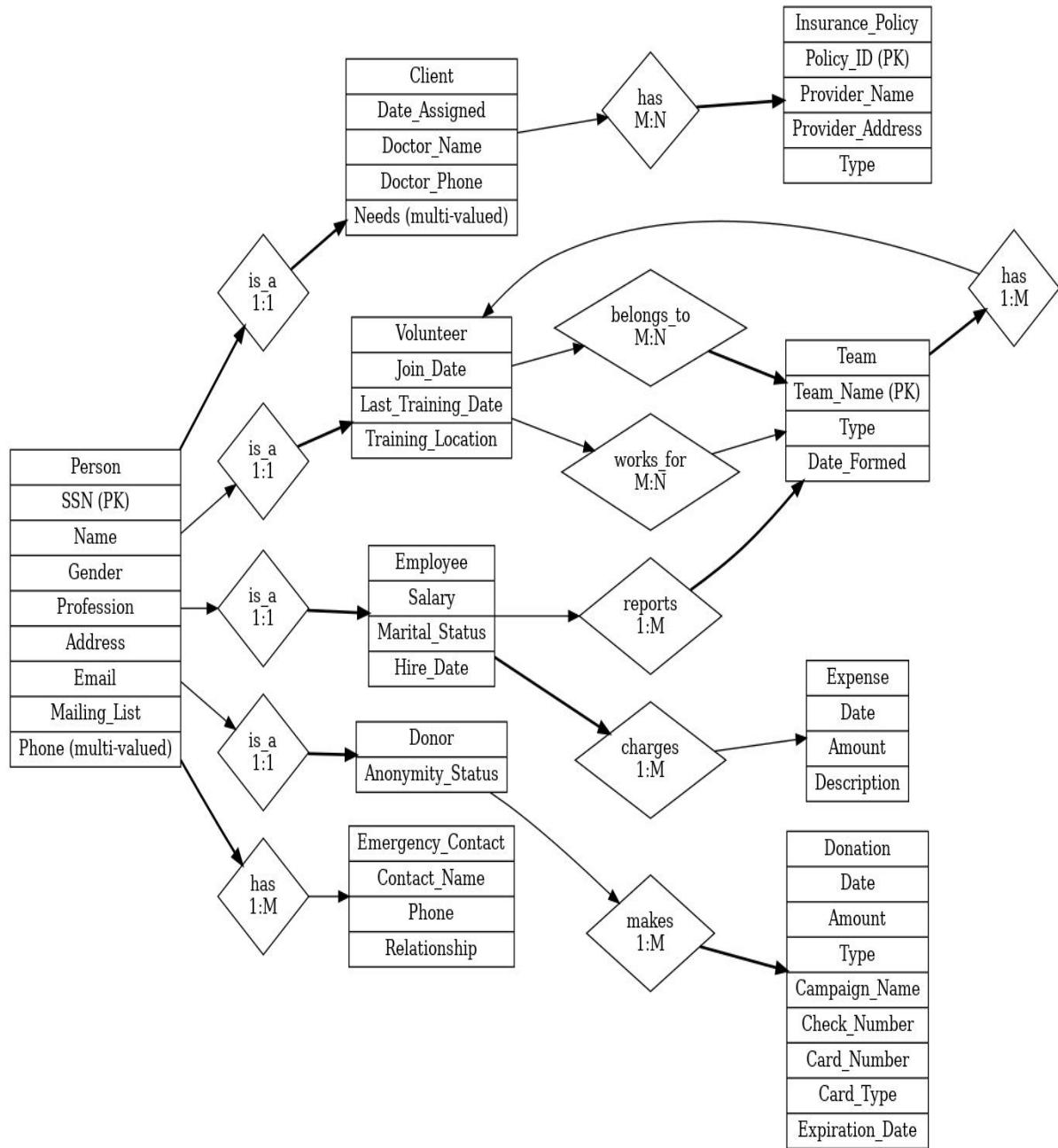


Figure 1: Entity-Relationship (ER) Diagram for Patient Assistant Network Database System

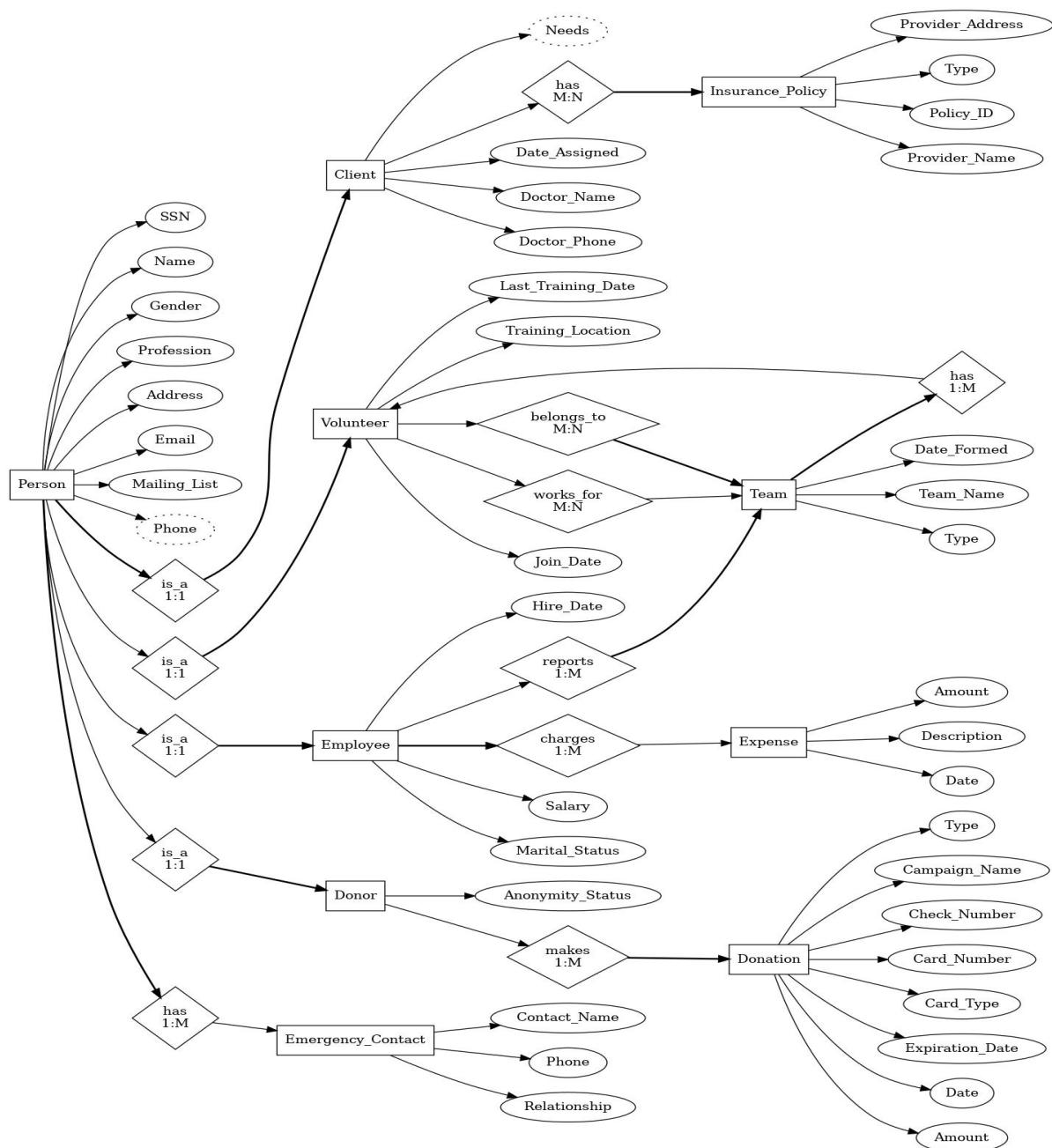


Figure 2: Entity-Relationship (ER) Diagram for Patient Assistant Network Database System

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

```
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)

```
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)

```
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)

```
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

```
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
);
```

Screenshot of SQL Server Management Studio (SSMS) showing the 'Person' table definition.

Table Properties

- General**
 - Table name: Person
 - Schema: dbo
 - Description:
- System Versioning**: System Versioning Enabled (checkbox checked)
- Memory Optimized**: Memory Optimized (checkbox unchecked)

Columns

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
1	SSN	varchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
2	Name	varchar(100)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
3	Gender	char(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
4	Profession	varchar(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
5	Address	varchar(200)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
6	Email	varchar(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
7	Mailing_List	bit	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...

Scripts

```

1 CREATE TABLE [dbo].[Person] (
2     [SSN] ..... VARCHAR (50) NOT NULL,
3     [Name] ..... VARCHAR (100) NOT NULL,
4     [Gender] ..... CHAR (1) ..... NULL,
5     [Profession] ..... VARCHAR (100) NULL,
6     [Address] ..... VARCHAR (200) NULL,
7     [Email] ..... VARCHAR (100) NULL,
8     [Mailing_List] BIT ..... NOT NULL,
9     PRIMARY KEY CLUSTERED ([SSN] ASC),
10    CHECK ([Gender]='M' OR [Gender]='F' OR [Gender]='I'),
11    UNIQUE NONCLUSTERED ([Email] ASC)
12 );
13
14

```

Table 2: Phone

```
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
);
```

The screenshot shows the SQL Server Management Studio (SSMS) interface. On the left is the Object Explorer tree, which includes nodes for CONNECTIONS, SER... (selected), chan0239-sql-serv..., Tables, Views, Synonyms, Programmability, External Resources, Storage, and Security. In the center, the 'Tables' node is expanded, and a new table named 'Phone' is being created. The 'Table Properties' pane on the right shows the table name as 'Phone', schema as 'dbo', and no description. Under 'System Versioning', 'System Versioning Enabled' is unchecked. Under 'Memory Optimized', 'Memory Optimized' is also unchecked. The 'Columns' tab of the 'Phone' table properties is selected, displaying two columns: 'SSN' (Primary Key, varchar(50)) and 'Phone_Number' (varchar(20)). The 'Script' tab at the bottom contains the T-SQL code for creating the table:

```
1 CREATE TABLE [dbo].[Phone] (
2     [SSN] VARCHAR (50) NOT NULL,
3     [Phone_Number] VARCHAR (20) NOT NULL,
4     PRIMARY KEY CLUSTERED ([SSN] ASC, [Phone_Number] ASC),
5     FOREIGN KEY ([SSN]) REFERENCES [dbo].[Person]([SSN]) ON DELETE CASCADE
6 );
7
8 
```

Table 3: Client

```
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);
```

Screenshot of SQL Server Management Studio (SSMS) showing the Object Explorer, Table Designer, and Scripts pane.

Object Explorer:

- Connections: Welcome, SQLQuery_1 - disconnected, SQLQuery_3 - disconnected, SQLQuery_4 - disconnected, SQLQuery_2 - disconnected, SQLQuery_5 - (87) c.an0239, dbo.Client
- SER... (selected)
- Tables: dbo.AUTHORS, dbo.BOOKS, dbo.Class, dbo.Client (selected), dbo.ENROLLMENT, dbo.MEMBER, dbo.Person, dbo.Phone, dbo.Student, dbo.WRITING, Dropped Ledger...
- Views, Synonyms, Programmability, External Resources, Storage, Security

Table Designer (dbo.Client):

Table Properties:

- General: Table name Client, Schema dbo
- System Versioning: System Versioning Enabled (unchecked)
- Memory Optimized: Memory Optimized (unchecked)

Columns:

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
==	Client_SSN	varchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>	...
==	Date_Assigned	date	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>	...
==	Doctor_Name	varchar(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="button" value="Delete"/>	...
==	Doctor_Phone	varchar(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="button" value="Delete"/>	...

Scripts:

```

1 CREATE TABLE [dbo].[Client] (
2     [Client_SSN] VARCHAR (50) NOT NULL,
3     [Date_Assigned] DATE NOT NULL,
4     [Doctor_Name] VARCHAR (100) NULL,
5     [Doctor_Phone] VARCHAR (20) NULL,
6     PRIMARY KEY CLUSTERED ([Client_SSN] ASC),
7     FOREIGN KEY ([Client_SSN]) REFERENCES [dbo].[Person]([SSN]) ON DELETE CASCADE
8 );
9
10
11 GO
12 CREATE NONCLUSTERED INDEX [idx_Client_SSN]
13     ON [dbo].[Client]([Client_SSN] ASC);
14

```

Table 4: Need

```
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
);
```

Table Properties

General

Table name	Need
Schema	dbo
Description	

System Versioning

System Versioning Enabled	<input type="checkbox"/>
---------------------------	--------------------------

Memory Optimized

Memory Optimized	<input type="checkbox"/>
------------------	--------------------------

Choose SQL Language: chan0239-sql-s

Table 5: Insurance_Policy

```
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);
```

The screenshot shows the SQL Server Management Studio (SSMS) interface. On the left is the Object Explorer tree, which includes a connection to 'chan0239-sql-serv...' and various database objects like 'dbo.AUTHORS', 'dbo.BOOKS', 'dbo.Class', 'dbo.Client', 'dbo.Enrollment', 'dbo.Member', 'dbo.Need', 'dbo.Person', 'dbo.Phone', 'dbo.Student', 'dbo.WRITING', and several system-related nodes. The main workspace displays the 'dbo.Insurance_Policy' table definition. The table has four columns: 'Policy_ID' (Primary Key, VARCHAR(50)), 'Provider_Name' (VARCHAR(100)), 'Provider_Address' (VARCHAR(200)), and 'Type' (VARCHAR(20)). The 'Table Properties' pane on the right shows the table name as 'Insurance_Policy', schema as 'dbo', and system versioning as disabled. Below the table definition is a 'Scripts' section containing the T-SQL code used to create the table:

```
1 CREATE TABLE [dbo].[Insurance_Policy] (
2     [Policy_ID] VARCHAR (50) NOT NULL,
3     [Provider_Name] VARCHAR (100) NULL,
4     [Provider_Address] VARCHAR (200) NULL,
5     [Type] VARCHAR (20) NULL,
6     PRIMARY KEY CLUSTERED ([Policy_ID] ASC),
7     CHECK ([Type]='auto' OR [Type]='home' OR [Type]='health' OR [Type]='life')
8 );
9
10
11 GO
12 CREATE NONCLUSTERED INDEX [idx_Insurance_Type]
13     ON [dbo].[Insurance_Policy]([Type] ASC);
14
```

Table 6: Team

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

The screenshot shows the SQL Server Management Studio (SSMS) interface. The left pane displays the Object Explorer with a tree view of database objects. The 'Tables' node under 'dbo' is expanded, showing various tables like AUTHORS, BOOKS, CLASS, Client, Enrollment, MEMBER, Need, Person, Phone, and Team. The 'Team' table is selected and highlighted in blue. The right pane shows the 'Table Properties' dialog for the 'Team' table. The 'General' tab is selected, displaying the table name 'Team', schema 'dbo', and a description field. The 'System Versioning' tab has an unchecked checkbox for 'System Versioning Enabled'. The 'Memory Optimized' tab has an unchecked checkbox for 'Memory Optimized'. Below the properties, there is a 'Scripts' section containing the CREATE TABLE script for the 'Team' table:

```
1 CREATE TABLE [dbo].[Team] (
2     [Team_Name] VARCHAR (50) NOT NULL,
3     [Type] VARCHAR (100) NULL,
4     [Date_Formed] DATE NOT NULL,
5     PRIMARY KEY CLUSTERED ([Team_Name] ASC)
6 );
7
8
```

Table 7: Volunteer

```
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
);
```

Screenshot of SQL Server Management Studio (SSMS) showing the creation of a table named 'Volunteer'.

Table Properties

- General**: Table name: Volunteer, Schema: dbo
- System Versioning**: System Versioning Enabled:
- Memory Optimized**: Memory Optimized:

Table Structure

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
==	Volunteer_SSN	varchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
==	Join_Date	date	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
==	Last_Training_Date	date	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
==	Training_Location	varchar(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...

Scripts

```

1 CREATE TABLE [dbo].[Volunteer] (
2     [Volunteer_SSN] ..... VARCHAR (50) NOT NULL,
3     [Join_Date] ..... DATE ..... NOT NULL,
4     [Last_Training_Date] DATE ..... NULL,
5     [Training_Location] VARCHAR (100) NULL,
6     PRIMARY KEY CLUSTERED ([Volunteer_SSN] ASC),
7     FOREIGN KEY ([Volunteer_SSN]) REFERENCES [dbo].[Person] ([[SSN]) ON DELETE CASCADE
8 );
9
10

```

Table 8: Volunteer_Team

```
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);
```

Screenshot of SQL Server Management Studio (SSMS) showing the creation of a table named 'Volunteer_Team'.

Table Properties

- General**: Table name is 'Volunteer_Team', Schema is 'dbo'.
- System Versioning**: System Versioning Enabled is checked.
- Memory Optimized**: Memory Optimized is checked.

Columns (Table 'Volunteer_Team')

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
==	Volunteer_SSN	varchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
==	Team_Name	varchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
==	Is_Leader	bit	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
==	Is_Active	bit	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
==	Work_Hours	int	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...

Scripts

```

1 CREATE TABLE [dbo].[Volunteer_Team] (
2     [Volunteer_SSN] VARCHAR (50) NOT NULL,
3     [Team_Name] VARCHAR (50) NOT NULL,
4     [Is_Leader] BIT NULL,
5     [Is_Active] BIT NULL,
6     [Work_Hours] INT NULL,
7     PRIMARY KEY CLUSTERED ([Volunteer_SSN] ASC, [Team_Name] ASC),
8     FOREIGN KEY ([Team_Name]) REFERENCES [dbo].[Team]([Team_Name]),
9     FOREIGN KEY ([Volunteer_SSN]) REFERENCES [dbo].[Volunteer]([Volunteer_SSN])
10 );
11
12
13 GO
14 CREATE NONCLUSTERED INDEX [idx_Volunteer_Team]

```

Table 9: Employee

```
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
);
```

Screenshot of SQL Server Management Studio (SSMS) showing the creation of a new table named 'Employee'.

Table Properties

- General**: Table name: Employee, Schema: dbo
- System Versioning**: System Versioning Enabled:
- Memory Optimized**: Memory Optimized:

Columns

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
1	Employee_SSN	varchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
2	Salary	decimal(10,2)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
3	Marital_Status	varchar(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...
4	Hire_Date	date	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	...

Scripts

```

1 CREATE TABLE [dbo].[Employee] (
2     [Employee_SSN] VARCHAR (50) NOT NULL,
3     [Salary] DECIMAL (10, 2) NOT NULL,
4     [Marital_Status] VARCHAR (10) NULL,
5     [Hire_Date] DATE NULL,
6     PRIMARY KEY CLUSTERED ([Employee_SSN] ASC),
7     FOREIGN KEY ([Employee_SSN]) REFERENCES [dbo].[Person] ([SSN]) ON DELETE CASCADE
8 );
9
10

```

Choose SQL Language: chan0239-sql

Table 10: Expense

```
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;
```

SER... File Edit View Help

Welcome SQLQuery_2 - disconnected SQLQuery_5 - (87) c...an0239 X dbo.Expense X dbo.Employee

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	<input checked="" type="checkbox"/>					...
Employee_SSN	varchar(50)	<input type="checkbox"/>	<input type="checkbox"/>				...
Date	date	<input type="checkbox"/>	<input type="checkbox"/>				...
Amount	decimal(10,2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>				...
Description	varchar(200)	<input type="checkbox"/>	<input checked="" type="checkbox"/>				...

Table Properties

General

Table name: Expense
Schema: dbo
Description:

System Versioning

System Versioning Enabled:

Memory Optimized

Memory Optimized:

Scripts

```

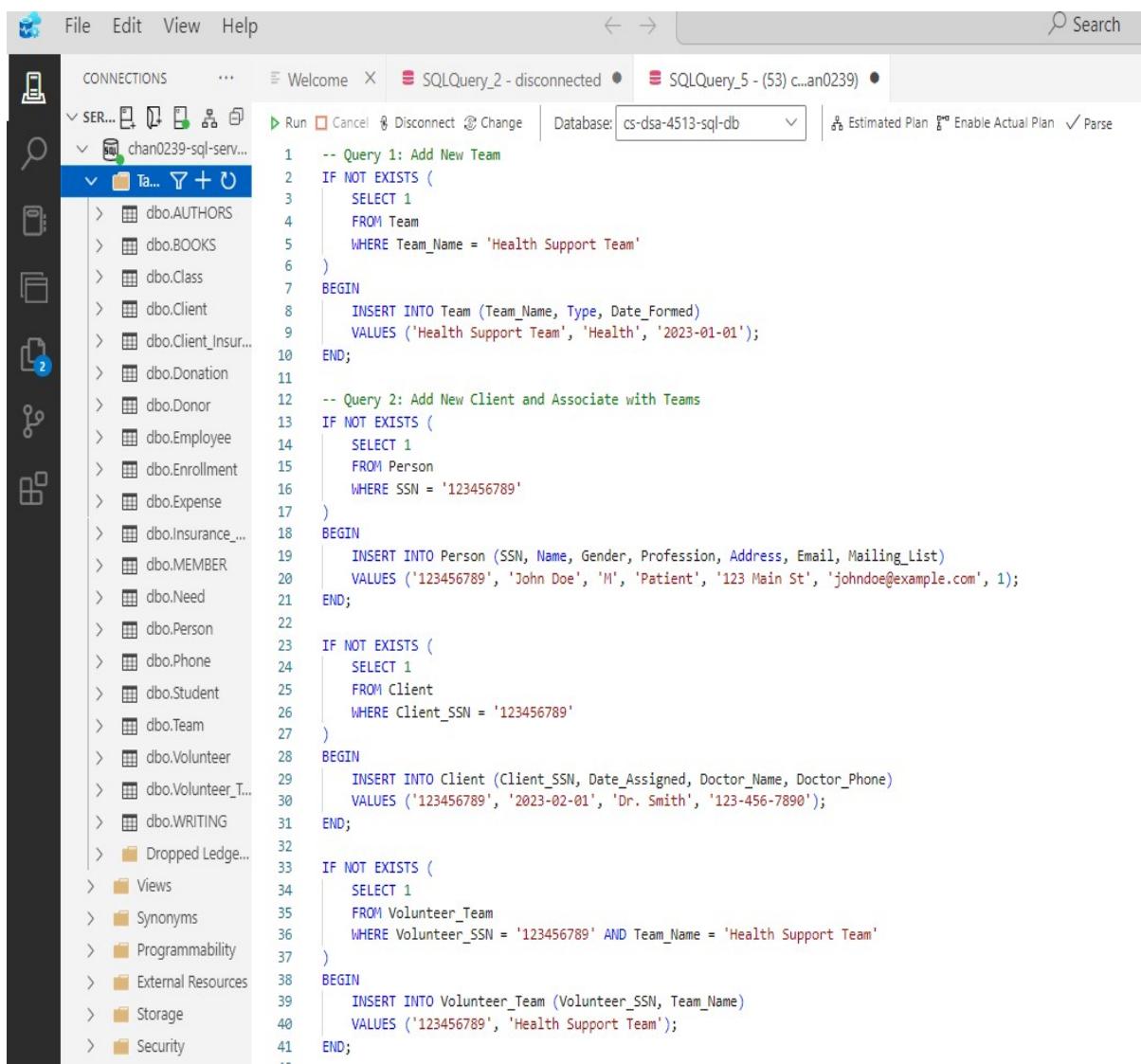
1 CREATE TABLE [dbo].[Expense] (
2     [Expense_ID] INT IDENTITY (1, 1) NOT NULL,
3     [Employee_SSN] VARCHAR (50) NOT NULL,
4     [Date] DATE NOT NULL,
5     [Amount] DECIMAL (10, 2) NULL,
6     [Description] VARCHAR (200) NULL,
7     PRIMARY KEY CLUSTERED ([Expense_ID] ASC),
8     FOREIGN KEY ([Employee_SSN]) REFERENCES [dbo].[Employee]([Employee_SSN])
9 );
10
11

```

AZURE > Choose SQL resource: [choose0227calcuus](#)

Task 5

5.1 SQL Query 1-15:



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The left sidebar displays a tree view of database objects under 'SER... chan0239-sql-serv...', including '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', and various system categories like 'Views', 'Synonyms', 'Programmability', 'External Resources', 'Storage', and 'Security'. The main window shows a query editor with the following T-SQL script:

```
1 -- Query 1: Add New Team
2 IF NOT EXISTS (
3     SELECT 1
4     FROM Team
5     WHERE Team_Name = 'Health Support Team'
6 )
7 BEGIN
8     INSERT INTO Team (Team_Name, Type, Date_Formed)
9     VALUES ('Health Support Team', 'Health', '2023-01-01');
10 END;
11
12 -- Query 2: Add New Client and Associate with Teams
13 IF NOT EXISTS (
14     SELECT 1
15     FROM Person
16     WHERE SSN = '123456789'
17 )
18 BEGIN
19     INSERT INTO Person (SSN, Name, Gender, Profession, Address, Email, Mailing_List)
20     VALUES ('123456789', 'John Doe', 'M', 'Patient', '123 Main St', 'johndoe@example.com', 1);
21 END;
22
23 IF NOT EXISTS (
24     SELECT 1
25     FROM Client
26     WHERE Client_SSN = '123456789'
27 )
28 BEGIN
29     INSERT INTO Client (Client_SSN, Date_Assigned, Doctor_Name, Doctor_Phone)
30     VALUES ('123456789', '2023-02-01', 'Dr. Smith', '123-456-7890');
31 END;
32
33 IF NOT EXISTS (
34     SELECT 1
35     FROM Volunteer_Team
36     WHERE Volunteer_SSN = '123456789' AND Team_Name = 'Health Support Team'
37 )
38 BEGIN
39     INSERT INTO Volunteer_Team (Volunteer_SSN, Team_Name)
40     VALUES ('123456789', 'Health Support Team');
41 END;
```

```
42
43 -- Query 3: Add New Volunteer and Associate with Teams
44 IF NOT EXISTS (
45     SELECT 1
46     FROM Person
47     WHERE SSN = '987654321'
48 )
49 BEGIN
50     INSERT INTO Person (SSN, Name, Gender, Profession, Address, Email, Mailing_List)
51     VALUES ('987654321', 'Jane Smith', 'F', 'Volunteer', '456 Elm St', 'janesmith@example.com', 1);
52 END;
53
54 IF NOT EXISTS (
55     SELECT 1
56     FROM Volunteer
57     WHERE Volunteer_SSN = '987654321'
58 )
59 BEGIN
60     INSERT INTO Volunteer (Volunteer_SSN, Join_Date, Last_Training_Date, Training_Location)
61     VALUES ('987654321', '2023-01-15', '2023-05-01', 'Main Campus');
62 END;
63
64 IF NOT EXISTS (
65     SELECT 1
66     FROM Volunteer_Team
67     WHERE Volunteer_SSN = '987654321' AND Team_Name = 'Health Support Team'
68 )
69 BEGIN
70     INSERT INTO Volunteer_Team (Volunteer_SSN, Team_Name)
71     VALUES ('987654321', 'Health Support Team');
72 END;
73
74 -- Query 4: Log Volunteer Hours for a Team
75 UPDATE Volunteer_Team
76 SET Work_Hours = 20
77 WHERE Volunteer_SSN = '987654321' AND Team_Name = 'Health Support Team';
78
79 -- Query 5: Add New Employee and Associate with Teams
80 IF NOT EXISTS (
81     SELECT 1
82     FROM Person
83     WHERE SSN = '555123456'
84 )
```

```
84    )
85    BEGIN
86        INSERT INTO Person (SSN, Name, Gender, Profession, Address, Email, Mailing_List)
87        VALUES ('555123456', 'Alice Johnson', 'F', 'Employee', '789 Pine St', 'alicejohnson@example.com', 1);
88    END;
89
90    IF NOT EXISTS (
91        SELECT 1
92        FROM Employee
93        WHERE Employee_SSN = '555123456'
94    )
95    BEGIN
96        INSERT INTO Employee (Employee_SSN, Salary, Marital_Status, Hire_Date)
97        VALUES ('555123456', 55000.00, 'Married', '2022-01-10');
98    END;
99
100   IF NOT EXISTS (
101        SELECT 1
102        FROM Volunteer_Team
103        WHERE Volunteer_SSN = '555123456' AND Team_Name = 'Health Support Team'
104    )
105    BEGIN
106        INSERT INTO Volunteer_Team (Volunteer_SSN, Team_Name)
107        VALUES ('555123456', 'Health Support Team');
108    END;
109
110    -- Query 6: Log Employee Expense
111    IF NOT EXISTS (
112        SELECT 1
113        FROM Expense
114        WHERE Employee_SSN = '555123456' AND Date = '2023-03-10'
115    )
116    BEGIN
117        INSERT INTO Expense (Employee_SSN, Date, Amount, Description)
118        VALUES ('555123456', '2023-03-10', 250.00, 'Travel Expense');
119    END;
120
121    -- Query 7: Add New Donor and Record Donations
122    IF NOT EXISTS (
123        SELECT 1
124        FROM Donor
125        WHERE SSN = '444987654'
126    )
```

The screenshot shows the SSMS interface with the following details:

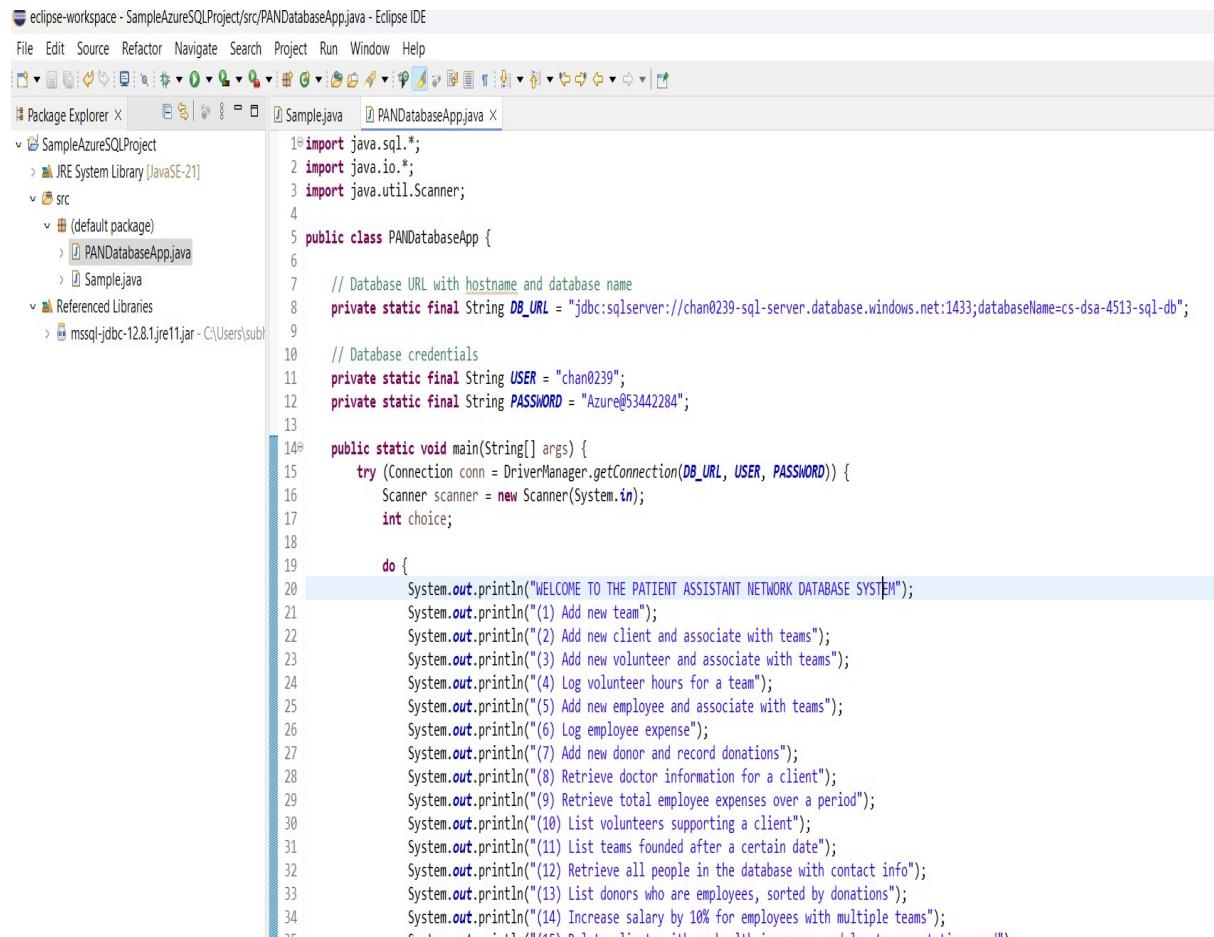
- File Edit View Help**
- CONNECTIONS**: SER... (selected), chan0239-sql-serv..., Ta... (highlighted)
- SQLQuery_2 - disconnected** and **SQLQuery_5 - (53) c..an0239**
- Database:** cs-dsa-4513-sql-db
- Buttons:** Run, Cancel, Disconnect, Change, Database, Estimated Plan, Enable Actual Plan, Parse
- Search Bar:** Search
- Object Explorer:** Shows tables (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), views, synonyms, programmability, external resources, storage, and security objects.
- Code Editor:** Displays 16 numbered lines of T-SQL code. Lines 126-166 are shown, starting with a closing parenthesis and ending with a final WHERE clause.

```
126 )
127 BEGIN
128     INSERT INTO Donor (SSN, Anonymity_Status)
129     VALUES ('444987654', 1);
130 END;
131
132 IF NOT EXISTS (
133     SELECT 1
134     FROM Donation
135     WHERE SSN = '444987654' AND Date = '2023-02-20'
136 )
137 BEGIN
138     INSERT INTO Donation (SSN, Date, Amount, Type, Campaign_Name, Check_Number, Card_Number, Card_Type, Expiration_Date)
139     VALUES ('444987654', '2023-02-20', 1000.00, 'one-time', 'Annual Fundraiser', NULL, '1111222233334444', 'Visa', '2024-12-31');
140 END;
141
142 -- Query 8: Retrieve Doctor Information for a Client
143 SELECT Doctor_Name, Doctor_Phone
144 FROM Client
145 WHERE Client_SSN = '123456789';
146
147 -- Query 9: Total Employee Expenses Over a Period
148 SELECT Employee_SSN, SUM(Amount) AS Total_Expenses
149 FROM Expense
150 WHERE Date BETWEEN '2023-01-01' AND '2023-12-31'
151 GROUP BY Employee_SSN
152 ORDER BY Total_Expenses DESC;
153
154 -- Query 10: List Volunteers Supporting a Client
155 SELECT Volunteer.Volunteer_SSN, Person.Name
156 FROM Volunteer
157 JOIN Volunteer_Team ON Volunteer.Volunteer_SSN = Volunteer_Team.Volunteer_SSN
158 JOIN Client ON Volunteer_Team.Team_Name = Client.Team_Name
159 JOIN Person ON Volunteer.Volunteer_SSN = Person.SSN
160 WHERE Client.Client_SSN = '123456789';
161
162 -- Query 11: List Teams Founded After a Certain Date
163 SELECT Team_Name
164 FROM Team
165 WHERE Date_Formed > '2023-01-01';
166
```

The screenshot shows the SSMS interface with the following details:

- File Bar:** File, Edit, View, Help.
- Toolbar:** Includes icons for New Query, Open, Save, Print, and Help.
- Connections:** Shows a connection to "chan0239-sql-serv..." under "SER...".
- Object Explorer:** Displays a tree view of database objects:
 - Tables: AUTHORS, BOOKS, Class, Client, Client_Insur..., Donation, Donor, Employee, Enrollment, Expense, Insurance_..., MEMBER, Need, Person, Phone, Student, Team, Volunteer, Volunteer_T...
 - Views
 - Synonyms
 - Programmability
 - External Resources
 - Storage
 - Security
- SQL Query Editor:** Contains a multi-line T-SQL script with numbered lines (166 to 203). The script includes several queries (e.g., Query 12, 13, 14, 15) and an UPDATE statement for increasing employee salaries.
- Status Bar:** Shows the database name "cs-dsa-4513-sql-db" and other connection-related information.

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



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure: SampleAzureSQLProject (selected), JRE System Library [JavaSE-11], src (selected), (default package) (selected), PANDatabaseApp.java (selected), Sample.java, and Referenced Libraries (mssql-jdbc-12.8.1.jre11.jar).
- Code Editor:** Displays the PANDatabaseApp.java code. The code is a Java application for a Patient Assistant Network Database System. It imports java.sql, java.io, and java.util.Scanner. It defines a class PANDatabaseApp with a main method. The main method connects to a database using JDBC and prints a menu of 14 options to the console. The code uses System.out.println for output and Scanner for input.

```
import java.sql.*;
import java.io.*;
import java.util.Scanner;

public class PANDatabaseApp {

    // Database URL with hostname and database name
    private static final String DB_URL = "jdbc:sqlserver://chan0239-sql-server.database.windows.net:1433;databaseName=cs-dsa-4513-sql-db";

    // Database credentials
    private static final String USER = "chan0239";
    private static final String PASSWORD = "Azure@53442284";

    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD)) {
            Scanner scanner = new Scanner(System.in);
            int choice;

            do {
                System.out.println("WELCOME TO THE PATIENT ASSISTANT NETWORK DATABASE SYSTEM");
                System.out.println("(1) Add new team");
                System.out.println("(2) Add new client and associate with teams");
                System.out.println("(3) Add new volunteer and associate with teams");
                System.out.println("(4) Log volunteer hours for a team");
                System.out.println("(5) Add new employee and associate with teams");
                System.out.println("(6) Log employee expense");
                System.out.println("(7) Add new donor and record donations");
                System.out.println("(8) Retrieve doctor information for a client");
                System.out.println("(9) Retrieve total employee expenses over a period");
                System.out.println("(10) List volunteers supporting a client");
                System.out.println("(11) List teams founded after a certain date");
                System.out.println("(12) Retrieve all people in the database with contact info");
                System.out.println("(13) List donors who are employees, sorted by donations");
                System.out.println("(14) Increase salary by 10% for employees with multiple teams");
            } while (choice != 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X PANDatabaseApp.java X

```
Sample.java PANDatabaseApp.java X
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);
68     }
69 }
```

Console X

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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X Sample.java PANDatabaseApp.java X

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

```
    retrieveDoctorInfo(conn, scanner);
    break;
case 9:
    retrieveEmployeeExpenses(conn, scanner);
    break;
case 10:
    listVolunteersForClient(conn, scanner);
    break;
case 11:
    listTeamsAfterDate(conn, scanner);
    break;
case 12:
    retrieveAllPeople(conn, scanner);
    break;
case 13:
    listDonorsWhoAreEmployees(conn, scanner);
    break;
case 14:
    increaseEmployeeSalary(conn);
    break;
case 15:
    deleteClients(conn);
    break;
case 16:
    importTeams(conn, scanner);
    break;
case 17:
    exportMailingList(conn, scanner);
    break;
case 18:
    System.out.println("Exiting the application.");
    break;
default:
    System.out.println("Invalid choice. Try again.");
```

Console X 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 X Sample.java PANDatabaseApp.java X

```
100     System.out.println("Invalid choice. Try again.");
101     break;
102 }
103 } while (choice != 18);
104 scanner.close();
105 } catch (SQLException e) {
106     e.printStackTrace();
107 }
108 }
109 }
110 }

111 private static void addTeam(Connection conn, Scanner scanner) {
112     System.out.print("Enter Team Name: ");
113     String teamName = scanner.nextLine();
114     String sql = "INSERT INTO Team (Team_Name, Type, Date_Formed) VALUES (?, ?, ?)";
115     try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
116         pstmt.setString(1, teamName);
117         pstmt.setString(2, "Health");
118         pstmt.setDate(3, Date.valueOf("2023-01-01"));
119         pstmt.executeUpdate();
120         System.out.println("Team added successfully.");
121     } catch (SQLException e) {
122         e.printStackTrace();
123     }
124 }
125 }

126 private static void addClient(Connection conn, Scanner scanner) {
127     System.out.print("Enter Client SSN: ");
128     String ssn = scanner.nextLine();
129     System.out.print("Enter Date Assigned (YYYY-MM-DD): ");
130     String dateAssigned = scanner.nextLine();
131     System.out.print("Enter Doctor Name: ");
132     String doctorName = scanner.nextLine();
133     System.out.println("Client added successfully.");
134 }
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the "SampleAzureSQLProject" with its contents: "src" (containing "PANDatabaseApp.java" and "Sample.java"), "JRE System Library [JavaSE-21]", and "Referenced Libraries" (including "mssql-jdbc-12.8.1.jre11.jar").
- Code Editor:** Displays the content of "PANDatabaseApp.java". The code is a Java application that inserts data into a database. It uses Scanner for input and PreparedStatement for database operations. The code includes imports for java.util.Scanner, java.sql.Connection, java.sql.DriverManager, java.sql.PreparedStatement, and java.sql.SQLException.
- Console:** Shows the output of the application's execution. The message "Client added and associated with team successfully." is printed to the console.

```
132 System.out.print("Enter Doctor Name: ");
133 String doctorName = scanner.nextLine();
134 System.out.print("Enter Doctor Phone: ");
135 String doctorPhone = scanner.nextLine();
136 System.out.print("Enter Associated Team: ");
137 String teamName = scanner.nextLine();
138
139 String clientSql = "INSERT INTO Client (SSN, Date_Assigned, Doctor_Name, Doctor_Phone) VALUES (?, ?, ?, ?)";
140 String teamSql = "INSERT INTO Volunteer_Team (SSN, Team_Name) VALUES (?, ?)";
141
142 try (PreparedStatement clientStmt = conn.prepareStatement(clientSql);
143      PreparedStatement teamStmt = conn.prepareStatement(teamSql)) {
144
145     clientStmt.setString(1, ssn);
146     clientStmt.setDate(2, Date.valueOf(dateAssigned));
147     clientStmt.setString(3, doctorName);
148     clientStmt.setString(4, doctorPhone);
149     clientStmt.executeUpdate();
150
151     teamStmt.setString(1, ssn);
152     teamStmt.setString(2, teamName);
153     teamStmt.executeUpdate();
154
155     System.out.println("Client added and associated with team successfully.");
156 } catch (SQLException e) {
157     e.printStackTrace();
158 }
159
160 private static void addVolunteer(Connection conn, Scanner scanner) {
161     System.out.print("Enter Volunteer SSN: ");
162     String ssn = scanner.nextLine();
163     System.out.print("Enter Join Date (YYYY-MM-DD): ");
164     String joinDate = scanner.nextLine();
165 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Shows the project structure under "SampleAzureSQLProject". It includes a JRE System Library [JavaSE-21], a source folder "src" containing files "PANDatabaseApp.java" and "Sample.java", and a Referenced Libraries folder containing "mssql-jdbc-12.8.1.jre11.jar" from the path C:\Users\subbu.
- Code Editor View:** Displays the content of the file "PANDatabaseApp.java". The code is a Java program that inserts data into a database. It uses Scanner for input and PreparedStatement for database operations. The code includes imports for java.util.Scanner, java.sql.Connection, java.sql.DriverManager, java.sql.PreparedStatement, and java.sql.SQLException.
- Console View:** Located at the bottom, it contains a single line of placeholder text: "Console X".

```
164 System.out.print("Enter Join Date (YYYY-MM-DD): ");
165 String joinDate = scanner.nextLine();
166 System.out.print("Enter Last Training Date (YYYY-MM-DD): ");
167 String trainingDate = scanner.nextLine();
168 System.out.print("Enter Training Location: ");
169 String location = scanner.nextLine();
170 System.out.print("Enter Associated Team: ");
171 String teamName = scanner.nextLine();
172
173 String volunteerSql = "INSERT INTO Volunteer (SSN, Join_Date, Last_Training_Date, Training_Location) VALUES (?, ?, ?, ?)";
174 String teamSql = "INSERT INTO Volunteer_Team (SSN, Team_Name) VALUES (?, ?)";
175
176 try (PreparedStatement volunteerStmt = conn.prepareStatement(volunteerSql);
177      PreparedStatement teamStmt = conn.prepareStatement(teamSql)) {
178
179     volunteerStmt.setString(1, ssn);
180     volunteerStmt.setDate(2, Date.valueOf(joinDate));
181     volunteerStmt.setDate(3, Date.valueOf(trainingDate));
182     volunteerStmt.setString(4, location);
183     volunteerStmt.executeUpdate();
184
185     teamStmt.setString(1, ssn);
186     teamStmt.setString(2, teamName);
187     teamStmt.executeUpdate();
188
189     System.out.println("Volunteer added and associated with team successfully.");
190 } catch (SQLException e) {
191     e.printStackTrace();
192 }
193
194 private static void logVolunteerHours(Connection conn, Scanner scanner) {
195     System.out.print("Enter Volunteer SSN: ");
196     String ssn = scanner.nextLine();
197     System.out.print("Enter Team Name: ");
198 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the "SampleAzureSQLProject" with its contents: "src" folder containing "PANDatabaseApp.java" and "Sample.java".
- Code Editor:** Displays the content of "PANDatabaseApp.java". The code is a Java application that interacts with a database to update volunteer hours and add employees.

```
196 System.out.print("Enter Volunteer SSN: ");
197 String ssn = scanner.nextLine();
198 System.out.print("Enter Team Name: ");
199 String teamName = scanner.nextLine();
200 System.out.print("Enter Hours Worked: ");
201 int hours = scanner.nextInt();
202 scanner.nextLine();
203
204 String sql = "UPDATE Volunteer_Team SET Hours_Worked = ? WHERE SSN = ? AND Team_Name = ?";
205
206 try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
207     pstmt.setInt(1, hours);
208     pstmt.setString(2, ssn);
209     pstmt.setString(3, teamName);
210     pstmt.executeUpdate();
211     System.out.println("Volunteer hours logged successfully.");
212 } catch (SQLException e) {
213     e.printStackTrace();
214 }
215
216 private static void addEmployee(Connection conn, Scanner scanner) {
217     System.out.print("Enter Employee SSN: ");
218     String ssn = scanner.nextLine();
219     System.out.print("Enter Salary: ");
220     double salary = scanner.nextDouble();
221     scanner.nextLine();
222     System.out.print("Enter Marital Status: ");
223     String maritalStatus = scanner.nextLine();
224     System.out.print("Enter Hire Date (YYYY-MM-DD): ");
225     String hireDate = scanner.nextLine();
226     System.out.print("Enter Associated Team: ");
227     String teamName = scanner.nextLine();
228
229 }
```

- Console:** Shows the output of the application's execution.

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

```
229
230     String employeeSql = "INSERT INTO Employee (SSN, Salary, Marital_Status, Hire_Date) VALUES (?, ?, ?, ?)";
231     String teamSql = "INSERT INTO Team_Employee (Team_Name, SSN) VALUES (?, ?)";
232
233     try (PreparedStatement employeeStmt = conn.prepareStatement(employeeSql);
234          PreparedStatement teamStmt = conn.prepareStatement(teamSql)) {
235
236         employeeStmt.setString(1, ssn);
237         employeeStmt.setDouble(2, salary);
238         employeeStmt.setString(3, maritalStatus);
239         employeeStmt.setDate(4, Date.valueOf(hireDate));
240         employeeStmt.executeUpdate();
241
242         teamStmt.setString(1, teamName);
243         teamStmt.setString(2, ssn);
244         teamStmt.executeUpdate();
245
246         System.out.println("Employee added and associated with team successfully.");
247     } catch (SQLException e) {
248         e.printStackTrace();
249     }
250 }
251
252 private static void logEmployeeExpense(Connection conn, Scanner scanner) {
253     System.out.print("Enter Expense ID: ");
254     int expenseId = scanner.nextInt();
255     scanner.nextLine();
256     System.out.print("Enter Employee SSN: ");
257     String ssn = scanner.nextLine();
258     System.out.print("Enter Expense Date (YYYY-MM-DD): ");
259     String date = scanner.nextLine();
260     System.out.print("Enter Expense Amount: ");
261     double amount = scanner.nextDouble();
262     scanner.nextLine();
263     System.out.print("Enter Expense Description: ");
```

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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X PANDatabaseApp.java X

```
261     double amount = scanner.nextDouble();
262     scanner.nextLine();
263     System.out.print("Enter Expense Description: ");
264     String description = scanner.nextLine();
265
266     String sql = "INSERT INTO Expense (Expense_ID, SSN, Date, Amount, Description) VALUES (?, ?, ?, ?, ?)";
267
268     try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
269         pstmt.setInt(1, expenseId);
270         pstmt.setString(2, ssn);
271         pstmt.setDate(3, Date.valueOf(date));
272         pstmt.setDouble(4, amount);
273         pstmt.setString(5, description);
274         pstmt.executeUpdate();
275         System.out.println("Employee expense logged successfully.");
276     } catch (SQLException e) {
277         e.printStackTrace();
278     }
279 }
280
281 private static void addDonor(Connection conn, Scanner scanner) {
282     System.out.print("Enter Donor SSN: ");
283     String ssn = scanner.nextLine();
284     System.out.print("Enter Anonymity Status (1 for anonymous, 0 for public): ");
285     int anonymityStatus = scanner.nextInt();
286     scanner.nextLine();
287     System.out.print("Enter Donation ID: ");
288     int donationId = scanner.nextInt();
289     scanner.nextLine();
290     System.out.print("Enter Donation Date (YYYY-MM-DD): ");
291     String date = scanner.nextLine();
292     System.out.print("Enter Donation Amount: ");
293     double amount = scanner.nextDouble();
294     scanner.nextLine();
```

Console X

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Shows the project structure under "SampleAzureSQLProject". It includes an "IREE System Library [JavaSE-21]", a "src" folder containing "PANDatabaseApp.java" and "Sample.java", and a "Referenced Libraries" section with "mssql-jdbc-12.8.1.jre11.jar".
- Code Editor View:** The active editor is "PANDatabaseApp.java". The code implements a function to add a donor and donation to a database. It uses Scanner for input and PreparedStatement for database operations. It handles exceptions and prints success messages.
- Console View:** The console output shows the command used to run the application: "PANDatabaseApp [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15-Nov-2024, 4:47:05 pm) [pid: 15840]".

```
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 }
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the "SampleAzureSQLProject" with its contents: "JRE System Library [JavaSE-21]", "src" containing "PANDatabaseApp.java" and "Sample.java", and "Referenced Libraries" containing "mssql-jdbc-12.8.1.jre11.jar".
- Code Editor:** Displays the content of "Sample.java". The code uses JDBC to query a database for employee expenses between specified start and end dates.

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X Sample.java PANDatabaseApp.java X
326     String ssn = scanner.nextLine();
327
328     String sql = "SELECT Doctor_Name, Doctor_Phone FROM Client WHERE SSN = ?";
329
330     try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
331         pstmt.setString(1, ssn);
332         ResultSet rs = pstmt.executeQuery();
333         if (rs.next()) {
334             System.out.println("Doctor Name: " + rs.getString("Doctor_Name"));
335             System.out.println("Doctor Phone: " + rs.getString("Doctor_Phone"));
336         } else {
337             System.out.println("No client found with the provided SSN.");
338         }
339     } catch (SQLException e) {
340         e.printStackTrace();
341     }
342 }
343
344 private static void retrieveEmployeeExpenses(Connection conn, Scanner scanner) {
345     System.out.print("Enter Start Date (YYYY-MM-DD): ");
346     String startDate = scanner.nextLine();
347     System.out.print("Enter End Date (YYYY-MM-DD): ");
348     String endDate = scanner.nextLine();
349
350     String sql = "SELECT SSN, SUM(Amount) AS Total_Expenses FROM Expense WHERE Date BETWEEN ? AND ? GROUP BY SSN ORDER BY Total_Expenses DESC";
351
352     try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
353         pstmt.setDate(1, Date.valueOf(startDate));
354         pstmt.setDate(2, Date.valueOf(endDate));
355         ResultSet rs = pstmt.executeQuery();
356         while (rs.next()) {
357             System.out.println("Employee SSN: " + rs.getString("SSN") + " | Total Expenses: " + rs.getDouble("Total_Expenses"));
358         }
359     } catch (SQLException e) {
```

- Console:** Shows the output of the application running in the console window.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "SampleAzureSQLProject". It includes the "src" folder containing "PANDatabaseApp.java" and "Sample.java", and the "Referenced Libraries" folder containing "mssql-jdbc-12.8.1.jre11.jar".
- Code Editor:** The active editor tab is "PANDatabaseApp.java". The code implements two static methods: "listVolunteersForClient" and "listTeamsAfterDate". Both methods use JDBC to query a database, involving preparation of statements, execution, and handling of SQL exceptions.
- Console:** The console tab shows the output of a Java application named "PANDatabaseApp" running under "javaw.exe" with PID 15840. The output is currently empty.

```
359     } catch (SQLException e) {
360         e.printStackTrace();
361     }
362 }
363
364 private static void listVolunteersForClient(Connection conn, Scanner scanner) {
365     System.out.print("Enter Client SSN: ");
366     String ssn = scanner.nextLine();
367
368     String sql = "SELECT Volunteer.SSN, Volunteer.Name FROM Volunteer "
369                 + "JOIN Volunteer_Team ON Volunteer.SSN = Volunteer_Team.SSN "
370                 + "JOIN Client ON Volunteer_Team.Team_Name = Client.Team_Name "
371                 + "WHERE Client.SSN = ?";
372
373     try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
374         pstmt.setString(1, ssn);
375         ResultSet rs = pstmt.executeQuery();
376         while (rs.next()) {
377             System.out.println("Volunteer SSN: " + rs.getString("SSN") + " | Name: " + rs.getString("Name"));
378         }
379     } catch (SQLException e) {
380         e.printStackTrace();
381     }
382 }
383
384 private static void listTeamsAfterDate(Connection conn, Scanner scanner) {
385     System.out.print("Enter Date (YYYY-MM-DD): ");
386     String date = scanner.nextLine();
387
388     String sql = "SELECT Team_Name FROM Team WHERE Date_Formed > ?";
389
390     try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
391         pstmt.setDate(1, Date.valueOf(date));
392         ResultSet rs = pstmt.executeQuery();
393         ...
394     }
```

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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer X PANDatabaseApp.java X

```
391     pstmt.setDate(1, Date.valueOf(date));
392     ResultSet rs = pstmt.executeQuery();
393     while (rs.next()) {
394         System.out.println("Team Name: " + rs.getString("Team_Name"));
395     }
396 } catch (SQLException e) {
397     e.printStackTrace();
398 }
399 }

400 private static void retrieveAllPeople(Connection conn, Scanner scanner) {
401     String sql = "SELECT SSN, Name, Gender, Profession, Address, Email, Phone FROM Person";
402     try (Statement stmt = conn.createStatement()) {
403         ResultSet rs = stmt.executeQuery(sql);
404         while (rs.next()) {
405             System.out.println("SSN: " + rs.getString("SSN") +
406                 " | Name: " + rs.getString("Name") +
407                 " | Gender: " + rs.getString("Gender") +
408                 " | Profession: " + rs.getString("Profession") +
409                 " | Address: " + rs.getString("Address") +
410                 " | Email: " + rs.getString("Email") +
411                 " | Phone: " + rs.getString("Phone"));
412         }
413     } catch (SQLException e) {
414         e.printStackTrace();
415     }
416 }
417 }

418 }

419 private static void listDonorsWhoAreEmployees(Connection conn, Scanner scanner) {
420     String sql = "SELECT Donor.SSN, SUM(Donation.Amount) AS Total_Donations, Donor.Anonymity_Status " +
421         "FROM Donor " +
422         "JOIN Donation ON Donor.SSN = Donation.SSN " +
423         "JOIN Employee ON Donor.SSN = Employee.SSN " +
424 }
```

Console X

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

SampleJava PANDatabaseApp.java

```

425     "GROUP BY Donor.SSN, Donor.Anonymity_Status " +
426     "ORDER BY Total_Donations DESC";
427
428     try (Statement stmt = conn.createStatement()) {
429         ResultSet rs = stmt.executeQuery(sql);
430         while (rs.next()) {
431             System.out.println("Donor SSN: " + rs.getString("SSN") +
432                 " | Total Donations: " + rs.getDouble("Total_Donations") +
433                 " | Anonymity Status: " + (rs.getInt("Anonymity_Status") == 1 ? "Anonymous" : "Public"));
434         }
435     } catch (SQLException e) {
436         e.printStackTrace();
437     }
438 }
439
440 private static void increaseEmployeeSalary(Connection conn) {
441     String sql = "UPDATE Employee " +
442         "SET Salary = Salary * 1.1 " +
443         "WHERE SSN IN (SELECT SSN FROM Team_Employee GROUP BY SSN HAVING COUNT(DISTINCT Team_Name) > 1)";
444
445     try (Statement stmt = conn.createStatement()) {
446         int rowsUpdated = stmt.executeUpdate(sql);
447         System.out.println(rowsUpdated + " employees' salaries increased.");
448     } catch (SQLException e) {
449         e.printStackTrace();
450     }
451 }
452
453 private static void deleteClients(Connection conn) {
454     String sql = "DELETE FROM Client " +
455         "WHERE SSN IN (SELECT SSN FROM Client_Needs WHERE Need_Type = 'transportation' AND Importance_Level < 5) " +
456         "AND SSN NOT IN (SELECT SSN FROM Client_Insurance WHERE Policy_ID IN (SELECT Policy_ID FROM Insurance_Policy WHERE Type =
457
458     try (Statement stmt = conn.createStatement()) {

```

Console X

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 X Samplejava PANDatabaseApp.java X

```
457     try (Statement stmt = conn.createStatement()) {
458         int rowsDeleted = stmt.executeUpdate(sql);
459         System.out.println(rowsDeleted + " clients deleted.");
460     } catch (SQLException e) {
461         e.printStackTrace();
462     }
463 }
464
465
466 private static void importTeams(Connection conn, Scanner scanner) {
467     System.out.print("Enter file name to import teams: ");
468     String fileName = scanner.nextLine();
469
470     try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
471         String line;
472         String sql = "INSERT INTO Team (Team_Name, Type, Date_Formed) VALUES (?, ?, ?)";
473
474         try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
475             while ((line = reader.readLine()) != null) {
476                 String[] parts = line.split(",");
477                 pstmt.setString(1, parts[0]);
478                 pstmt.setString(2, parts[1]);
479                 pstmt.setDate(3, Date.valueOf(parts[2]));
480                 pstmt.executeUpdate();
481             }
482         }
483
484         System.out.println("Teams imported successfully.");
485     } catch (IOException | SQLException e) {
486         e.printStackTrace();
487     }
488 }
489
490 private static void exportMailingList(Connection conn, Scanner scanner) {
491     System.out.print("Enter file name to export mailing list: ");
492 }
```

Console X
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 X PANDatabaseApp.java X

```
470     pstmt.setString(2, parts[1]);
471     pstmt.setDate(3, Date.valueOf(parts[2]));
472     pstmt.executeUpdate();
473 }
474 System.out.println("Teams imported successfully.");
475 } catch (IOException | SQLException e) {
476     e.printStackTrace();
477 }
478 }
479
480 private static void exportMailingList(Connection conn, Scanner scanner) {
481     System.out.print("Enter file name to export mailing list: ");
482     String fileName = scanner.nextLine();
483
484     String sql = "SELECT Name, Address FROM Person WHERE Mailing_List = 1";
485
486     try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
487          Statement stmt = conn.createStatement();
488          ResultSet rs = stmt.executeQuery(sql)) {
489
490         while (rs.next()) {
491             writer.write(rs.getString("Name") + ", " + rs.getString("Address"));
492             writer.newLine();
493         }
494
495         System.out.println("Mailing list exported successfully.");
496     } catch (IOException | SQLException e) {
497         e.printStackTrace();
498     }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
```

Console X

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 X Sample.java PANDatabaseApp.java

SampleAzureSQLProject JRE System Library [JavaSE-21]

src (default package) PANDatabaseApp.java

Sample.java

Referenced Libraries mssql-jdbc-12.8.1.jre11.jar - C:\Users\subbu

```
497     Statement stmt = conn.createStatement();
498     ResultSet rs = stmt.executeQuery(sql)) {
499
500         while (rs.next()) {
501             writer.write(rs.getString("Name") + ", " + rs.getString("Address"));
502             writer.newLine();
503         }
504
505         System.out.println("Mailing list exported successfully.");
506     } catch (IOException | SQLException e) {
507         e.printStackTrace();
508     }
509 }
510 }
511
512 }
```

Console X

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: 3333333333
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: 3333333333
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: 4444444444
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: 4444444444
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: 5555555555
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: 5555555555
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: Volunteer

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 Team
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
1212121212	1
2323232323	0
3434343434	1
4545454545	0
565656565	1

Table: Donation

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

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	john.doe@example.com	123-456-7890
987654321	Jane Smith	F	Volunteer	456 Elm St	jane.smith@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!