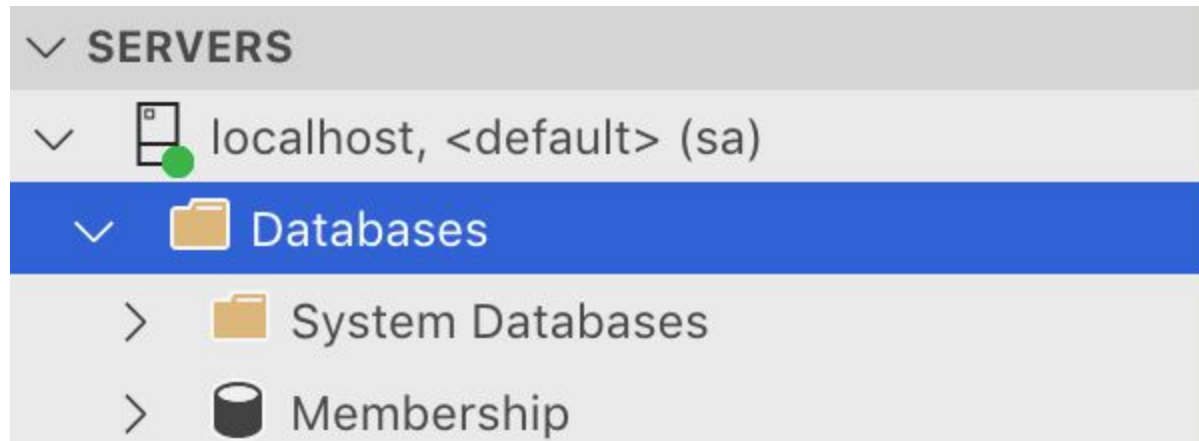


Kamran Shaikh
Lab8

1)
Code)

```
CREATE DATABASE Membership;
```

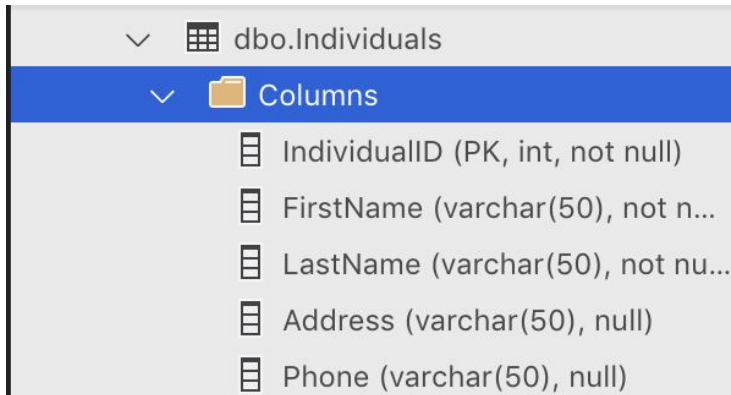
```
GO
```



2)

Code Individuals)

```
CREATE TABLE Individuals (IndividualID INT NOT NULL PRIMARY KEY IDENTITY, FirstName  
VARCHAR(50) NOT NULL, LastName VARCHAR(50) NOT NULL, Address VARCHAR(50) NULL, Phone  
VARCHAR(50) NULL)
```

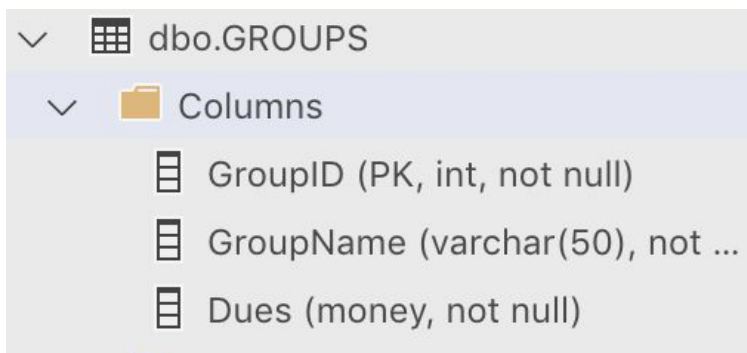


The screenshot shows the 'dbo.Individuals' table structure in SQL Server Enterprise Manager. The 'Columns' folder is expanded, showing five columns: IndividualID (PK, int, not null), FirstName (varchar(50), not null), LastName (varchar(50), not null), Address (varchar(50), null), and Phone (varchar(50), null).

Column Name	Data Type	Nullability	Other Properties
IndividualID	int	not null	Primary Key (PK)
FirstName	varchar(50)	not null	
LastName	varchar(50)	not null	
Address	varchar(50)	null	
Phone	varchar(50)	null	

Code Groups)

```
CREATE TABLE GROUPS (GroupID INT NOT NULL IDENTITY PRIMARY KEY, GroupName VARCHAR(50)  
NOT NULL, Dues MONEY NOT NULL DEFAULT 0 CHECK (Dues >= 0))
```



The screenshot shows the 'dbo.GROUPS' table structure in SQL Server Enterprise Manager. The 'Columns' folder is expanded, showing three columns: GroupID (PK, int, not null), GroupName (varchar(50), not null), and Dues (money, not null).

Column Name	Data Type	Nullability	Other Properties
GroupID	int	not null	Primary Key (PK)
GroupName	varchar(50)	not null	
Dues	money	not null	Default 0, Check constraint (Dues >= 0)

Code Memberships)

```
CREATE TABLE GroupsMembership (GroupID INT REFERENCES GROUPS(GroupID), IndividualID  
INT REFERENCES Individuals(IndividualID))
```



The screenshot shows the 'dbo.GroupsMembership' table structure in SQL Server Enterprise Manager. The 'Columns' folder is expanded, showing two columns: GroupID (FK, int, null) and IndividualID (FK, int, null).

Column Name	Data Type	Nullability	Other Properties
GroupID	int	null	Foreign Key (FK) to GROUPS(GroupID)
IndividualID	int	null	Foreign Key (FK) to Individuals(IndividualID)

Individuals and Groups have a many to many relationships. Individuals can be in multiple groups and groups can have multiple individuals.

3)

Code)

```
CREATE CLUSTERED INDEX IX_GROUPID ON GroupsMembership(GroupID)
```

```
CREATE NONCLUSTERED INDEX IX_IndividualID ON GroupsMembership(IndividualID)
```



4)

Code)

```
ALTER TABLE Individuals ADD DuesPaid BIT NOT NULL DEFAULT 0
```

Columns

IndividualID (PK, int, not null)

FirstName (varchar(50), not n...

LastName (varchar(50), not nu...

Address (varchar(50), null)

Phone (varchar(50), null)

DuesPaid (bit, not null)

5)

```
ALTER TABLE Invoices ADD CHECK ((PaymentDate IS NULL AND PaymentTotal = 0) OR  
(PaymentDate IS NOT NULL AND PaymentTotal > 0)), CHECK ((PaymentTotal + CreditTotal)  
<= InvoiceTotal)
```

6)

Code)

```
DROP TABLE GroupMembership
```

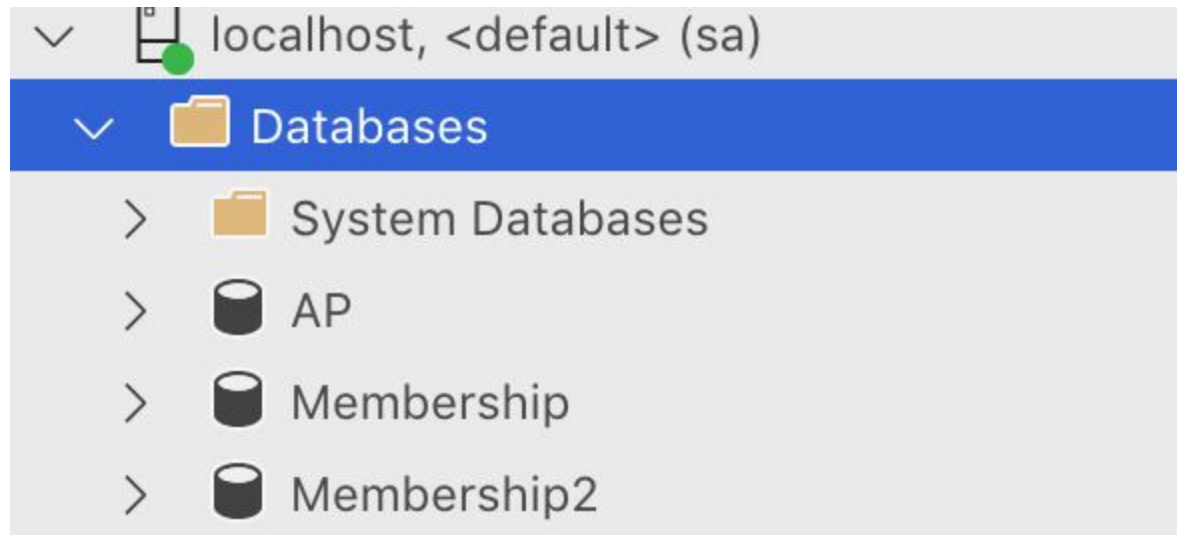
```
CREATE TABLE GroupMembership (GROUPID INT REFERENCES GROUPS (GROUPID), IndividualID INT  
REFERENCES Individuals (IndividualID), UNIQUE (GROUPID, IndividualID))
```

7)

Code)

```
CREATE DATABASE Membership2
```

```
GO
```



Remarks)

I really enjoyed this lab. It was a great way to learn how to create and alter databases and columns. I had some trouble with task 3 but after reviewing the notes I was able to figure it out. Overall these labs have really helped me learn about database implementation throughout the course.