Homework 4

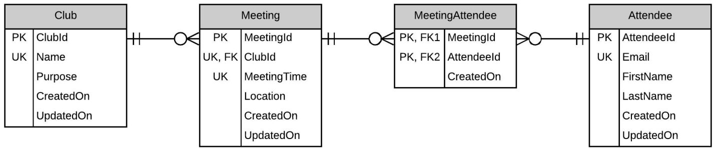
Due Oct 17 by 11:59pm **Points** 50 **Submitting** a file upload **File Types** sql and txt **Available** Oct 10 at 12am - Oct 17 at 11:59pm 8 days

This assignment was locked Oct 17 at 11:59pm.

The purpose of this exercise is to gain practice with data modification. Using the below diagram as your database schema to answer the questions below. You will be able to execute these scripts against your own database on *mssql.cs.ksu.edu*, or a new one you create if you installed SQL Server on your own machine.

Database Diagram

Reference this diagram as you answer the questions below.



	Club
PK	ClubId
UK	Name
	Purpose
	CreatedOn
	UpdatedOn

	Meeting
PK	MeetingId
UK, FK	ClubId
UK	MeetingTime
	Location
	CreatedOn
	UpdatedOn

	MeetingAttendee
PK, FK1	MeetingId
PK, FK2	AttendeeId
	CreatedOn

	Attendee
PK	AttendeeId
UK	Email
	FirstName
	LastName
	CreatedOn
	UpdatedOn

Questions

Write SQL solutions to the following four questions. You can submit them all in one file clearly indicating the question with a comment, or you can submit in multiple files.

Question 1

20 points - Write the SQL to create each of the tables in the above diagram, using the below requirements.

- Create all tables in a schema named "Homework4". This schema can be created using the **CREATE SCHEMA** statement.

  ```
  CREATE SCHEMA Homework4;
  ```

  Such as statement must be the only one in a batch. For simplicity, then, you **do not need to turn this statement in with your homework**. You can execute it with it being the only statement in your session, or you select this statement only before executing. You may delete it once you've executed it.
- For data types, you can use the following:
  - **INT** for all identifiers.
    - Also, use the **IDENTITY** property on primary keys rather than a sequence object.
  - **DATETIME2(0)** for *MeetingTime*
  - **DATETIMEOFFSET** for all columns named *CreatedOn* and *UpdatedOn*
  - **NVARCHAR** for all other fields - I'll let you determine the appropriate maximum number of characters for the **NVARCHAR** columns.
- Other column requirements:
  - Use the **IDENTITY** property on primary keys rather than a sequence object.

- All columns named *CreatedOn* and *UpdatedOn* should have a default constraint to capture the current date/time.
- All columns should disallow nulls, accomplished by using **NOT NULL**.

We have seen a few different ways to define constraints, but you can use the simple form as in this example.

```
CREATE TABLE Demo.Person
(
   PersonId INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
   FirstName NVARCHAR(32) NOT NULL,
   LastName NVARCHAR(32) NOT NULL,
   IsEmployee BIT NOT NULL DEFAULT(0),
   OrganizationId INT NOT NULL FOREIGN KEY
      REFERENCES Demo.Organization(OrganizationId),

   UNIQUE
   (
      FirstName ASC,
      LastName ASC
   )
);
```

Keep in mind that your tables will need to be created in such a way that the tables *referenced* by foreign keys are created before the tables *with* foreign keys.

Question 2

15 points - Write an insert statement to create two clubs, and a second insert statement to create two meetings for each club (total of 4 meetings).

Here is the information for the clubs.

Name	Purpose
ACM	The Association for Computing Machinery is the professional organization for computer scientists.
MIS Club	The Kansas State MIS Club is a student driven organization focused on the management of information systems.

Meeting times and location.

Club	Location	Meeting Times
ACM	Engineering Building 1114	October 9, 2018 at 6:30 pm November 13, 2018 at 6:30 pm
MIS Club	Business Building 2116	November 6, 2018 at 6:00 pm December 4, 2018 at 6:00 pm

When inserting the meetings, do not make any assumptions regarding the *ClubId*. Start with a derived table providing the information in the second table above, then join to the **Club** table on the name to obtain the *ClubId*.

Question 3

10 points - Write two insert statements to create yourself as an attendee, and that you attended the ACM meeting on October 9, 2018. Use your name and KSU email address for the insertion. Again, do not make an assumption regarding what the automatically generated *AttendeeId* will be, but since you are just inserting one attendee, you can use a function to obtain the generated *AttendeeId*.

Question 4

5 points - Write an update statement to modify the location of the meeting on December 4, 2018 of the MIS Club. The new meeting location should be "Business Building 4001". Don't forget to update the *UpdatedOn* column. Your query should only use the literals for the club name, meeting date/time, and the new location, making no assumptions on identifiers.

Helpful Tips

As you work on your solutions, here are a couple tips that you may find useful.

Refreshing Intellisense

After creating a new schema or table, IntelliSense, which supports auto-complete while typing, doesn't always detect the new objects right away, especially if you are trying to use the new objects in a different session than where you created them. In SQL Server Management Studio, you can force it to refresh the cache by using *Edit* → *IntelliSense* → *Refresh Local Cache*. As a shortcut, I typically just use *Ctrl+Shift+R*.

Recreating Tables

If you find that you made a mistake and need to re-create your table(s), then you need to drop the table first, using a **DROP TABLE** statement, such as:

```
DROP TABLE Demo.Person;
```

Of course, if there is another table referencing the table you need to drop, then you will need to drop it as well. For our purposes, this is the simplest approach since re-creating our tables is painless.

Because recreating our tables is painless, you may also keep a **DROP TABLE** statement using the **IF EXISTS** condition at the top of your script. This will drop the table if it exists, then your **CREATE TABLE** statements should always work.

Let's look at an example with two tables, *Demo.A* and *Demo.B.* The table *Demo.B* references *Demo.A* with a foreign key constraint. We will precede the **CREATE TABLE** statements with the conditional **DROP TABLE** statements in reverse order, so there will be no dependency on *Demo.A* when attempting to drop it.

```
DROP TABLE IF EXISTS Demo.B;
DROP TABLE IF EXISTS Demo.A;

CREATE TABLE Demo.A
(
   <column and constraint definitions here>
);

CREATE TABLE Demo.B
(
   <column and constraint definitions here>,
);
```

Homework 4 Rubric

Criteria	Ratings		Pts
Question 1	**20.0 pts** **Full Marks**	**0.0 pts** **No Marks**	20.0 pts
Question 2	**15.0 pts** **Full Marks**	**0.0 pts** **No Marks**	15.0 pts
Question 3	**10.0 pts** **Full Marks**	**0.0 pts** **No Marks**	10.0 pts
Question 4	**5.0 pts** **Full Marks**	**0.0 pts** **No Marks**	5.0 pts
			Total Points: 50.0