Assignment 1 – Consultative Solution

Exercise 2

```
-- 1) Retrieve a list of ssn, fname, lname and salary for all employee.

      SELECT       SSN, FName, LName, Salary
      FROM         Employee

-- 2) Redo exercise a) but this time show the list in decreasing order by salary.

      SELECT       SSN, FName, LName, Salary
      FROM         Employee
      ORDER BY     Salary DESC

-- 3) Retrieve a list of ssn, fname, lname and address for all employees from
--    department 5.

      SELECT       SSN, FName, LName, Address
      FROM         Employee
      WHERE        Dno = 5

-- 4) Retrieve a list of all employees working in the 'Administration' department.

SELECT       Employee.*
FROM         Employee
             JOIN Department ON Employee.Dno = Department.DNumber
WHERE        Department.DName = 'Administration'
```

```
-- 5) Retrieve a list of all projects (number, and name) controlled by the
--    'Research' department.

      SELECT       Project.PNumber, Project.PName
      FROM         Project
                   JOIN Department ON Project.DNum = Department.DNumber
      WHERE        Department.DName = 'Research'

-- 6) Retrieve a list of all employees working on the 'ProductX' project.

      SELECT       Employee.*
      FROM         Employee
                   JOIN Works_on ON Employee.SSN = Works_on.Essn
                   JOIN Project ON Works_on.Pno = Project.PNumber
      WHERE        Project.PName = 'ProductX'


-- 7) Retrieve a list of all employees living in Houston.

      SELECT       SSN, LName, FName, Address
      FROM         Employee
      WHERE        Address LIKE '%Houston%'
```

Exercise 3

Here is a SQL-script for creating the MyTunes database without any data. The script has been sanitized by removing (a lot of) not needed statements from the script generated by SQL Management Studio.

```
      USE [master]
      GO

      IF DB_ID('MyTunes') IS NOT NULL
      BEGIN
            DROP DATABASE [MyTunes]
      END
      GO

      CREATE DATABASE [MyTunes]
      GO

      USE [MyTunes]
      GO

      SET ANSI_NULLS ON
      GO
```

```
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [Artist]
(
	[ID]		[int]			IDENTITY(1,1)		NOT NULL,
	[Name]		[nvarchar](50)					NOT NULL,
	CONSTRAINT [PK_Artist] PRIMARY KEY CLUSTERED ([ID])
)
GO

CREATE TABLE [Category]
(
	[ID]		[int]			IDENTITY(1,1)		NOT NULL,
	[Category]	[nvarchar](50)					NOT NULL,
	CONSTRAINT [PK_Category] PRIMARY KEY CLUSTERED ([ID])
)
GO

CREATE TABLE [PlayList]
(
	[ID]		[int]			IDENTITY(1,1)		NOT NULL,
	[Name]		[nvarchar](50)					NOT NULL,
	[Created]	[datetime]					NOT NULL,
	CONSTRAINT [PK_PlayList] PRIMARY KEY CLUSTERED ([ID])
)
GO

CREATE TABLE [PlayListSong]
(
	[PlayListID] [int]					NOT NULL,
	[SongID]		[int]				NOT NULL,
	[SeqNo]			[int]				NOT NULL,
	CONSTRAINT [PK_PlayListSong]
		PRIMARY KEY CLUSTERED ([PlayListID], [SongID])
)
GO
```


```
CREATE TABLE [Song]
(
	[ID]		[int]			IDENTITY(1,1)		NOT NULL,
	[Title]		[nvarchar](50)					NOT NULL,
	[ArtistID]	[int]						NOT NULL,
	[CategoryID] [int]						NULL,
	[FileName]	[nvarchar](max)					NOT NULL,
	[Duration]	[int]						NULL,
		CONSTRAINT [PK_Song] PRIMARY KEY CLUSTERED ([ID])
)
GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_Artist] ON [Artist]([Name])
GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_Category] ON Category([Category])
GO


ALTER TABLE [PlayListSong]
	ADD CONSTRAINT [FK_PlayListSong_PlayList]
		FOREIGN KEY([PlayListID]) REFERENCES [PlayList] ([ID])
		ON DELETE CASCADE
GO

ALTER TABLE [PlayListSong]
	ADD CONSTRAINT [FK_PlayListSong_Song]
		FOREIGN KEY([SongID]) REFERENCES [Song] ([ID])
		ON DELETE CASCADE
GO

ALTER TABLE [Song]
	ADD CONSTRAINT [FK_Song_Artist]
		FOREIGN KEY([ArtistID]) REFERENCES [Artist] ([ID])
		ON DELETE CASCADE
GO

ALTER TABLE [Song]
	ADD CONSTRAINT [FK_Song_Category]
		FOREIGN KEY([CategoryID]) REFERENCES [Category] ([ID])
		ON DELETE SET NULL
GO
```

Exercise 4

3 INSERT Statements for each relation:

```
INSERT INTO Artist(Name)
	VALUES ('Bryan Adams');
INSERT INTO Artist(Name)
	VALUES ('Queen');
INSERT INTO Artist(Name)
	VALUES ('Deep Purple');

INSERT INTO Category(Category)
	VALUES ('Rock');
INSERT INTO Category(Category)
	VALUES ('Pop');
INSERT INTO Category(Category)
	VALUES ('Techno');

INSERT INTO PlayList(Name, Created)
	VALUES ('PlayList 1', getDate());
INSERT INTO PlayList(Name, Created)
	VALUES ('PlayList 2', getDate());
INSERT INTO PlayList(Name, Created)
	VALUES ('PlayList 3', getDate());

INSERT INTO SONG(Title, ArtistID, CategoryID, FileName, Duration)
	VALUES ('Summer Of 69', 1, 1, 'Music/Sommer_of_69.mp3', 216);
INSERT INTO SONG(Title, ArtistID, CategoryID, FileName, Duration)
	VALUES ('Bohemian Rhapsody', 2, 1, 'Music/Bohemian_Rhapsody.mp3', 354);
INSERT INTO SONG(Title, ArtistID, CategoryID, FileName, Duration)
	VALUES ('Smoke On The Water', 3, 1, 'Music/Smoke_On_The_Water.mp3', 378);

INSERT INTO PlayListSong(PlayListId, SongId, SeqNo)
	Values(1,1,1);
INSERT INTO PlayListSong(PlayListId, SongId, SeqNo)
	Values(1,2,2);
INSERT INTO PlayListSong(PlayListId, SongId, SeqNo)
	Values(2,3,1);
```

3 UPDATE Statements:

```
UPDATE PlayList
	SET Name = 'Favorites'
	WHERE ID = 1;
UPDATE Song
	SET CategoryID = 2
	WHERE ID = 1;
UPDATE Category
	SET Category = 'Modern'
	WHERE ID = 3;
```

3 DELETE Statements:

```
DELETE FROM SONG
	WHERE ID = 2;
DELETE FROM PlayList
	WHERE ID = 3;
DELETE FROM Artist
	WHERE ID = 2;
```

Exercise 5 – More joins

```
-- 1. Retrieve song title and duration for all songs for a given artist name.
	SELECT		Song.Title, Song.Duration
	FROM		Song
			JOIN Artist ON Artist.ID = Song.ArtistID
	WHERE		Artist.Name = 'Bryan Adams'

-- 2. Retrieve song title, artist name, category name and duration for all songs ordered -
--    by song title.
	SELECT		Song.Title, Artist.Name, Category.Category, Song.Duration
	FROM		Song
			JOIN Artist ON Artist.ID = Song.ArtistID
			JOIN Category ON Category.ID = Song.CategoryID
	ORDER BY	Song.Title

-- 3. Retrieve a list of all artists having no songs.
	SELECT		Artist.ID, Artist.Name
	FROM		Artist
	WHERE		Artist.ID NOT IN
			(
				SELECT DISTINCT ArtistID
				FROM   SONG
			)

-- 4. Retrieve song title, artist name and duration for all songs from a given playlist.
	SELECT		Song.Title, Artist.Name, Song.Duration
	FROM		PlayListSong
			JOIN   Song		ON Song.ID = PlayListSong.SongID
			JOIN   PlayList		ON PlayList.ID = PlayListSong.PlayListID
			JOIN   Artist		ON Artist.ID = Song.ArtistID
	WHERE		PlayList.PlayListName = 'PlayList 1'
```