Views, Procedures, Functions and Transactions Homework

Using the Chinook database and SQL Server Management Studio (SSMS), write SQL queries for the following questions. Submit your answers in a single SQL file to the Blackboard site. Use SQL's commenting syntax to include your name at the top of the file. Also include the question number as a comment before each answer. The answer key to the questions will be released after the deadline to submit homework has passed.

NOTE: You will need to create a view, a function, and a stored procedure in this assignment. To make building and testing your code easier I recommend placing the following code at the top of your page:

```
USE Chinook
IF OBJECT_ID('Track_v_edw') IS NOT NULL DROP VIEW Track_v_edw
IF OBJECT_ID('ArtistAlbum_fn_edw') IS NOT NULL DROP FUNCTION ArtistAlbum_fn_edw
IF OBJECT_ID('TracksByArtist_p_edw') IS NOT NULL DROP PROC TracksByArtist_p_edw
```

This code will automatically drop the objects if they exist so you won't have to do it manually when testing your CREATE statements. You will need to replace my initials with your own. Also note that each CREATE and ALTER statement needs to run in its own batch so you must insert the GO keyword before and after each statement.

1. **Create a new view called Track_v_[Your Initials].**
   Example: Track_v_edw
   Add all Columns from the Track table to the view.
   Add the Name column from the Genre table and call it GenreName.
   Add the Name column from the MediaType table and call it MediaTypeName.

2. **Create a new function called ArtistAlbum_fn_[Your Initials]**
   Example: ArtistAlbum_fn_edw
   The function will take a track ID as input and output the track's artist and album.
   The function will accept a single parameter called @TrackId with a datatype of int.
   The function will return a single value with a datatype of varchar(100).
   The Artist name and Album title will be concatenated into a single value with a dash in-between.
   Hint: You will need to write a SELECT statement for the concatenate line. You will also need to declare a variable in the function (I used the following: DECLARE @ArtistAlbum varchar(100)).

3. **Create a new stored procedure called TracksByArtist_p_[Your Initals]**
   Example: TracksByArtist_p_edw
   The procedure will need to return an Artist's name as well as any album titles and track names associated with the artist.
   Include the following in the output: Artist name as ArtistName, album title as AlbumTitle, track name as TrackName
   The procedure will take a single parameter called @ArtistName with a datatype of varchar(100).
   The procedure will do a search of the ArtistName column based on the value entered for the parameter. Partial matches should be returned.
   Hint: The WHERE clause should include a LIKE keyword as well as the @ArtistName parameter. You may need to use concatenation in your WHERE clause.

4. **Write a SELECT statement using the Track_v view_[Your Initals]. (2 rows)**
   Include the following columns from the view: Name, GenreName, MediaTypeName.
   Add the Title column from the Album table.
   Filter on the track name where the name equals "Babylon".

5. **Write a SELECT statement using the Track_v_[Your Initals] view and ArtistAlbum_fn_[Your Initals] function. (1 row)**
   The SELECT statement will have Track_v as the data source and include 2 columns.
   The first column will contain the ArtistAlbum_fn function with TrackId as a parameter.
   Name the first column "Artist and Album".
   The second column will be the track name column. Name it TrackName.
   Filter the statement on GenreName equals "Opera".

6. **Execute the TracksByArtist_p_[Your Initals] procedure twice.**
   Execute it once with a parameter of "black". (56 rows)
   Execute it once with a parameter of "white". (1 row)

7. **Alter the TracksByArtist_p_[Your Initals] procedure.**
   Give the @ArtistName parameter a default value of "Scorpions".

8. **Execute the TracksByArtist_p_[Your Initals]. (12 rows)**
   Execute the procedure without a parameter.
   Hint: Your procedure should return data.

9. **Begin a Transaction and run an UPDATE statement.**
   Begin a transaction.
   Update the LastName field in the Employee table to equal your last name.
   Only update the record with an EmployeeId = 1.

10. **View the results, rollback the transaction and view the results again. (2 rows total)**
    Display the EmployeeId and LastName from the Employee table where the ID equals 1.
    Rollback the transaction you started in question #9.
    Display the EmployeeId and LastName from the Employee table where the ID equals 1.
    Hint: You should see different results.