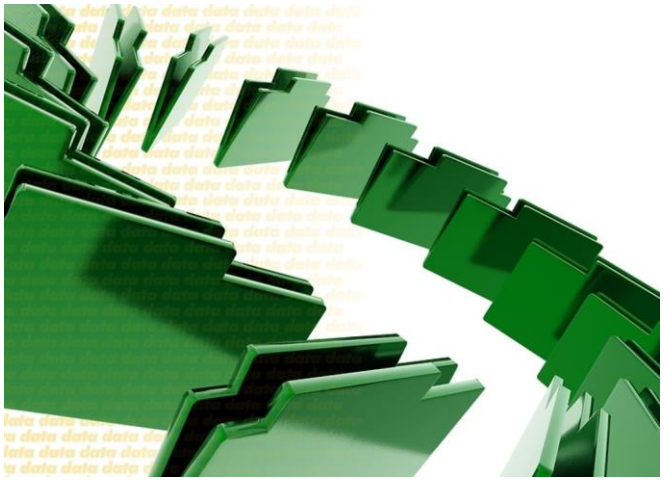

WEEK #10 ASSIGNMENT ANSWERS

2% Individual Assignment

Database Tables, Stored Procedures and Functions

Mark Morell

Database Management – Fall 2019

Assignment Type:

- Individual – Prepare and submit your results independently

Date Due:

- Thursday, November 14th by the end of the day

Instructions:

- Please submit your assignment electronically through eConestoga.
- Assignments should be submitted as Microsoft Word files using the course coversheet format. You **MUST** include your query as text in the Word document as well as a **FULL screenshot** of your SSMS screen (screen capture the entire application screen including the title bar through the bottom of the window). Multiple screenshots may be required.
- If you are using external sources (images, text, etc.) you must reference them as part of your assignment and not copy them as-is.
- Best practice is to research your answers and then write the response to the question in your own words.
- Please include the question number with your responses.

Late Assignment Penalty:

Days Late	Penalty %
1	5
2	10
3	20
4	40
5	60
6	80
7	100

Assignment Questions

Question #	Question	
1	Create a brand new database (if you didn't already do this in class – just show the database in your screenshot along with the syntax you would use) named "Entertainment". Create a new schema within this new database named "Television". Show your scripts to create both. -- Create the database: CREATE DATABASE Entertainment -- Create the schema in the database: USE Entertainment GO CREATE SCHEMA Television GO	1

2	Complete the following, making appropriate choices for column names as well as data types and primary key columns and constraints: a) Create a new table in the "Entertainment" database under the "Television" schema named "Show". The table will store the following information: • ID (required) • Name (required) • Genre (default as "Not Specified") • Year Launched (must be a year from 1900 on) b) Create a new table in the "Entertainment" database under the "Television" schema named "Episode". It will store the following data: • ID (required) • Show ID (required) • Show Season (must be a number > 0) • Show Episode Number (must be a number > 0) • Number of Viewers • Cost to Produce c) Create a new table in the "Entertainment" database under the "Television" schema named "Cast". It will store the following data: • ID (required) • Episode ID (required) • First Name (required) • Last Name (required) • Recording Hours (required & must be a number > 0) • Hourly Salary (required) Be sure to show the full SQL for creating the table (both with a screenshot and as text in your submission). -- Create the three tables: USE Entertainment GO CREATE TABLE Television.Show (id INT NOT NULL, name NVARCHAR(50) NOT NULL, genre VARCHAR(25) CONSTRAINT DF_genre DEFAULT 'Not Specified', debutYear SMALLINT NOT NULL, CONSTRAINT PK_id PRIMARY KEY (id), CONSTRAINT CK_year CHECK (debutYear >= 1900)); CREATE TABLE Television.Episode (id INT NOT NULL, showId INT NOT NULL, seasonNum SMALLINT NOT NULL, episodeNum SMALLINT NOT NULL, episodeName nvarchar(100) NULL, numViewer INT NULL, produceCost MONEY NULL,	5

```
        CONSTRAINT PK_episodeId PRIMARY KEY (id),
        CONSTRAINT FK_showId FOREIGN KEY (showId) REFERENCES Television.Show (id),
        CONSTRAINT CK_seasonNum CHECK (seasonNum > 0),
        CONSTRAINT CK_episodeNum CHECK (episodeNum > 0)
       );

CREATE TABLE Television.Cast
       ( id                    INT NOT NULL,
        episodeId              INT NOT NULL,
        firstName              NVARCHAR(32) NOT NULL,
        lastName               NVARCHAR(32) NOT NULL,
        recordingNumHour INT NOT NULL,
        hourlySalary           MONEY NOT NULL,
        CONSTRAINT PK_castId PRIMARY KEY (id, episodeId),
        CONSTRAINT FK_episodeId FOREIGN KEY (episodeId) REFERENCES Television.Episode (id),
        CONSTRAINT CK_recordingNumHour CHECK (recordingNumHour > 0)
       );
```

3	Insert a variety of appropriate data in each of the above tables. Each table should have 5-10 rows of data. -- Insert data into the three tables: USE Entertainment GO INSERT INTO Television.Show VALUES(1, 'Game of Thrones', 'Action, Adventure, Drama', 2011); INSERT INTO Television.Show VALUES(2, 'The X-Files', 'Crime, Drama, Mystery', 1993); INSERT INTO Television.Show VALUES(3, 'The Handmaids Tale', 'Drama, Sci-Fi, Thriller', 2017); INSERT INTO Television.Show VALUES(4, 'Star Trek: The Next Generation', 'Action, Adventure, Sci-Fi', 1987); INSERT INTO Television.Show VALUES(5, 'Knightfall', 'Action, Adventure, Drama', 2017); SELECT * FROM Television.Show INSERT INTO Television.Episode VALUES (1, 1, 8, 1, 'Winterfell', 4500000, 2300000); INSERT INTO Television.Episode VALUES (2, 1, 8, 2, 'A Knight of the Seven Kingdoms', 4750000, 1450000); INSERT INTO Television.Episode VALUES (3, 2, 1, 1, 'Pilot', 890000, 457000); INSERT INTO Television.Episode VALUES (4, 2, 1, 2, 'Deep Throat', 121000, 592000); INSERT INTO Television.Episode VALUES (5, 3, 3, 1, 'Night', 1200000, 998000); INSERT INTO Television.Episode VALUES (6, 3, 3, 2, 'Mary and Martha', 1250000, 1200000); INSERT INTO Television.Episode VALUES (7, 4, 7, 1, 'Descent: Part II', 256000, 527000); INSERT INTO Television.Episode VALUES (8, 4, 7, 2, 'Liaisons', 625000, 543000); INSERT INTO Television.Episode VALUES (9, 5, 1, 2, 'Find Us The Grail', 325000, 897000); INSERT INTO Television.Episode VALUES (10, 5, 1, 3, 'The Black Wolf and the White Wolf', 675000, 927000); SELECT * FROM Television.Episode INSERT INTO Television.Cast VALUES (1, 1, 'Emilia', 'Clarke', 47, 900); INSERT INTO Television.Cast VALUES (1, 2, 'Emilia', 'Clarke', 102, 950); INSERT INTO Television.Cast VALUES (2, 3, 'David', 'Duchovny', 120, 530); INSERT INTO Television.Cast VALUES (2, 4, 'David', 'Duchovny', 265, 500); INSERT INTO Television.Cast VALUES (3, 7, 'Brent', 'Spiner', 85, 999); SELECT * FROM Television.Cast	2

4	Create a new function that given a cast member's ID and show ID will calculate the total salary for that cast member over ever episode of the show that they have appeared on.
	
	-- Create a function to return salary over a show for a cast member
	CREATE FUNCTION Television.ufnCastShowSalary(@castId INT, @showId INT)
	RETURNS MONEY AS
	BEGIN
	  DECLARE @totalSalary MONEY;
	
	  SELECT @totalSalary = SUM(c.recordingNumHour * c.hourlySalary)
	  FROM Television.Cast AS c
	  INNER JOIN Television.Episode AS e ON e.id = c.episodeId
	  WHERE c.id = @castId AND
	  e.showId = @showId;
	
	  RETURN @totalSalary;
	END;
		2

5	Create a new stored procedure that takes a cast member's ID as *optional* input and produces the following results:
	• Full name of the cast member (formatted nicely)
	• Name of the show that they appeared on
	• Number of episodes of the show that they appeared on
	• The total salary for the cast member on the show (*Hint: Use the above function to make your life easier)*
	
	-- Create a procedure to return cast information with an optional parameter
	CREATE PROCEDURE Television.sp_castInformation
	  @castID AS INT = 0
	AS
	BEGIN
	  SELECT c.lastName + ', ' + c.firstName AS fullName,
	  s.name AS showName,
	  COUNT('x') AS numEpisodes,
	  Television.ufnCastShowSalary(c.id, s.id) AS showSalary
	  FROM Television.Cast AS c
	  INNER JOIN Television.Episode AS e ON e.id = c.episodeId
	  INNER JOIN Television.Show AS s ON s.id = e.showId
	  WHERE c.id = @castID OR @castID = 0
	  GROUP BY (c.lastName + ', ' + c.firstName), s.name, Television.ufnCastShowSalary(c.id, s.id)
	END;
		3

6	Show two executions of the above stored procedure including one where no input was provided and another where a specific cast member ID was provided. EXECUTE Television.sp_castInformation 1 -- No parameter: EXECUTE Television.sp_castInformation	2
	Total	**15**