SQL Project 8/2/2023

Beginning Microsoft SQL Server

USE AdventureWorks

\* In this first step, I use the “USE” clause to select the database that we will be working with. This is critical when working with a database server that stores multiple database records. \*

USE AdventureWorks

SELECT \*

FROM Person.Contact

\*\* In the second step, I add both a “SELECT” and “FROM” statement to our query. The USE clause gives our query the general location, but the SELECT and FROM statements both narrow it down. In this query, I selected AdventureWorks as our database, I wanted all information from it with our SELECT statement, and I chose the Person.Contact table with the FROM clause. \*\*

\*\* NOTE \*\* It is important to be aware that some SQL databases are case sensitive. The one that I am using today, Microsoft SQL Server, IS case sensitive!

\*\* NOTE 2 \*\* Right clicking on the Person.Contact table in the explore page, and then clicking “design” will bring you to the background data for the table. Here, analysts can more directly see what it is that they will be working with.

CREATE DATABASE StevenM

\*\*\* This command makes a new database for the server to explore. In this instance, I named the server using my first name and last initial. Note that you will need to refresh the objects page in order to see the result. \*\*\*

DROP DATABASE StevenM

\*\*\*\* This command deletes the Database that we just made. Like before, we will need to refresh the objects page to see the result. \*\*\*\*

\*\*\*\* NOTE \*\*\*\* I recreated the database for further use in the project. The database is named “StevenM”.

CREATE TABLE Cities(CityID int primary key, City navchar(25), Population bigint)

\*\*\*\*\* This new line of code will create our new table. NOTE: Be sure that the object drop down menu is displaying where you want this new table to go! The syntax is CREATE TABLE NAME(Column 1 integer primary key, Column 2 navchar(X), Column 3 big integer). \*\*\*\*\*

DROP TABLE dbo.Cities;

\*\*\*\*\*\* This command deletes the table from our database. Make sure to delete it from the correct database by referencing the object menu! \*\*\*\*\*

\*\*\*\*\*\* NOTE: I recreated the table afterwards to continue to step seven. \*\*\*\*\*\*

Delete the database and table using the object window

\*\*\*\*\*\*\* In this step, I right-clicked on the table, and then the database, to delete them. When making minor edits to the database, this is an acceptable practice. However, when deleting objects en masse, there are more efficient methods. \*\*\*\*\*\*\*

CREATE LOGIN Janet WITH PASSWORD ‘ABCxyz1234’

\* Here, we created a new user, “Janet”, with the new password “ABCxyz1234”. Janet will now have the ability to access our server by using her login verifications. \*

\* NOTE: Make sure that you execute this command in the right folder. In this case, we issue it in the “master” folder. \*

\* NOTE 2: Once the new username and password has been made, log out of SQL Server and reboot it. Under the authorization page, scroll to the user based verification, and try out the verifications there! Once done, close out and restart the program, logging in under the correct verification window. \*

ALTER LOGIN Janet WITH PASSWORD = ‘XYZabc1234’

\*\* This new line of code will generate a new password for Janet. It’s important to make sure that you are on the right page when issuing new passwords. Otherwise they will likely not work. \*\*

DROP LOGIN Janet

\*\*\* This string of code will delete Janet from the database. We can also do this, and the previous password change, from the objects menu. \*\*\*

CREATE DATABASE StevenM

CREATE TABLE Cities (

CityID int primary key,

City navrchar(25),

Population bigint,

);

NOTE: Double check that code set two is placed in the StevenM database!

USE StevenM;

GO

INSERT INTO Cities

VALUES (1, ‘Seattle’, 520000),

(2, ‘Redmond’, 110000),

(3, ‘Kirkland’, 85000),

(4, ‘Bothell’, 70000);

SELECT \*

FROM Cities

\*\*\*\* This block of codes needs to be done in order of highlighted color. They will:

1. Create our database
2. Create our table
3. Insert values into that table
4. Query and verify that everything is correct in that table

BEGIN TRAN;

UPDATE Cities

SET Population = Population+555

WAITFOR DELAY ’00:00:20’

COMMIT TRAN

PART 2

SELECT \* FROM Cities

\*\*\*\*\* This is known as a transaction query. In other words, two queries linked together. The first line and last line are linked together, as are the middle three. When we go to check it (seen in part two), there will be a 20 second delay from when we entered in the information to when we can see it. \*\*\*\*\*

BACKUP DATABASE StevenM

TO DISK = ‘C:\tmp\StevenM.bak’

GO

\*\*\*\*\*\* This query will create a new backup database for us to reference later. In this example, I saved it to a temporary folder on our desktop. If anything should happen with the database, i.e. corrupted files, viruses, accidentally deleted, etc., we will now have an additional and saved document on hand. \*\*\*\*\*\*

DROP DATABASE StevenM

\*\*\*\*\*\*\* In this final action, I deleted the database in order to restore it again. I assume that the database has been corrupted one way or another. From here, I go to the objects page, right click on ‘Databases,’ click ‘Restore Databases’ from the drop down menu, find the three little dots, navigate to my temporary folder, and restore the database ‘StevenM’ from there. When I go back to query it, everything is there! \*\*\*\*\*\*\*