Neetu Upadhyay
August 23, 2023,
Foundations Of Databases & SQL Programming

Assignment 07: Functions

Introduction

Welcome to Module07-Functions, a comprehensive dive into the world of SQL functions and their versatile applications. Functions are indispensable tools in SQL, allowing you to manipulate and process data, often simplifying complex queries into readable and maintainable code. In this module, we will explore the varied spectrum of SQL functions, from the basic aggregate functions to the more complex windowed functions.

Here's a quick breakdown of what you can expect:

- **Aggregate Functions (Review):** We begin with a refresher on aggregate functions. These functions allow you to perform calculations on sets of data, helping you quickly get summaries and insights.

- **Selecting with Common Functions:** Dive into some of the most frequently used SQL functions, understanding their use cases and syntax.

- **Partitioned or Windowed Functions:** Discover the power of windowed functions that operate on a set of table rows, providing a window into the data for more complex analyses.

- **Using Functions for Reporting:** Learn how to leverage functions to produce insightful reports, making data interpretation easier.

- **User Defined Functions:** Delve into creating your own functions tailored to specific requirements, bringing in flexibility and modularity to your SQL scripts.

- **Using UDFs for Check Constraints:** Understand how UDFs can be employed to set data integrity rules at a granular level.

Beyond SQL, this module also touches upon creating and formatting GitHub Pages:

- **Creating Advanced GitHub Pages:** Dive into the world of GitHub Pages, understanding their utility and how they can be used for documentation and showcasing projects.

- **Creating a Markdown File:** Get introduced to Markdown, a lightweight markup language, and understand how it plays a pivotal role in formatting GitHub Pages.

- **Formatting the Page:** Enhance your Markdown skills, ensuring your GitHub Pages are both appealing and informative.

By the end of this module, you will be equipped with a strong foundational understanding of SQL functions, their applications, and the knowledge to document and share your work on GitHub. Whether you're looking to improve data analysis, reporting, or just seeking to refine your SQL toolkit, this module promises to be an enriching journey.

1. **Explain when you would use a SQL UDF.**

   A User-Defined Function (UDF) in SQL is a feature that allows you to define your own functions that can accept parameters, perform an action, and return the result. UDFs can encapsulate a series of SQL statements into a single function, making it more modular and reusable. Here are scenarios when you might consider using a SQL UDF:

   a. **Reusable Logic**: If you find yourself writing the same SQL code in multiple queries or stored procedures, you can encapsulate this logic in a UDF and call the function instead.
   b. **Complex Calculations**: If you have calculations that are used repeatedly or are complex in nature, placing them in a UDF can make your SQL code cleaner and easier to read.
   c. **Modularity**: UDFs allow for modular programming. You can divide a complex query into smaller, more manageable pieces by using functions.
   d. **Consistency**: Ensuring that a specific logic or calculation is performed consistently across multiple queries or applications. By centralizing the logic in a UDF, you reduce the risk of inconsistencies.
   e. **Security**: UDFs can be used to abstract underlying table structures and provide users access to data without giving them direct access to tables.
   f. **Enhancing Native Functionality**: SQL provides a lot of built-in functions, but there might be specific operations unique to your application or business logic. UDFs allow you to extend the native functionality.
   g. **Check Constraints**: UDFs can be used in check constraints to enforce complex data integrity rules at the database level.
   h. **Readability**: Sometimes, using a well-named UDF can make your SQL more self-documenting. Instead of having to decipher a series of SQL statements, you can use a UDF with a descriptive name.

   However, it's essential to note that while UDFs have several benefits, they can also introduce performance overhead, especially scalar UDFs, which can be slower than inline SQL. It's always a good idea to monitor performance and ensure that the UDF doesn't become a bottleneck in your system.

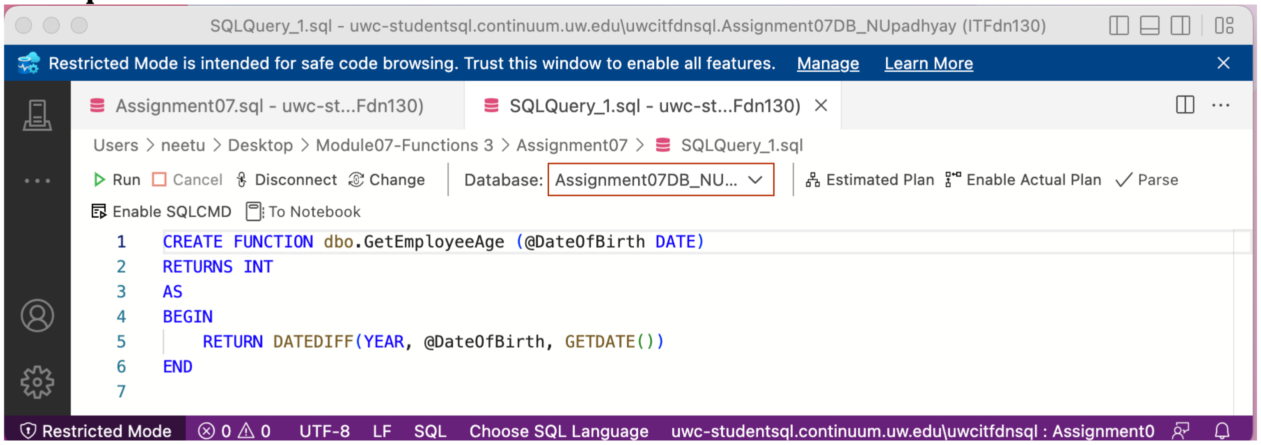
2. **Explain are the differences between Scalar, Inline, and Multi-Statement Functions.**
   SQL Server provides various types of User-Defined Functions (UDFs), with the primary ones being Scalar, Inline Table-Valued, and Multi-Statement Table-

Valued functions. Each serves a different purpose and has distinct characteristics:

1. **Scalar Functions:**

- Return Type: Scalar functions return a single value, not a table. This value can be any of the scalar data types, such as int, varchar, datetime, etc.
- Use Case: Useful when you need to perform calculations or operations that return a single value, like computing the age from a date of birth.

Example:



SQLQuery_1.sql - uwc-studentsql.continuum.uw.edu\uwcitfdnsql.Assignment07DB_NUpadhyay (ITFdn130)

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Users > neetu > Desktop > Module07-Functions 3 > Assignment07 > SQLQuery_1.sql

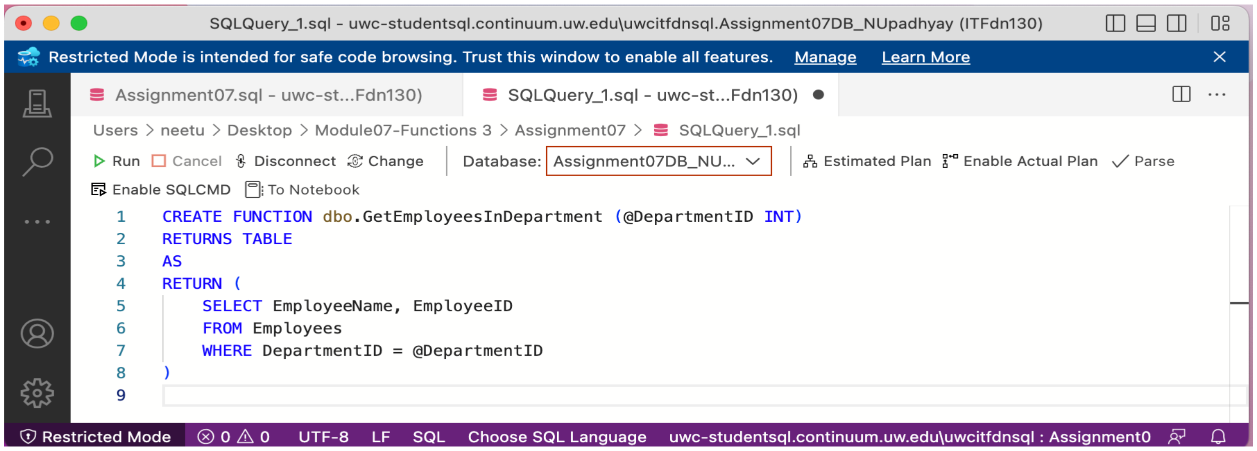
Database: Assignment07DB_NU...

```sql
CREATE FUNCTION dbo.GetEmployeeAge (@DateOfBirth DATE)
RETURNS INT
AS
BEGIN
    RETURN DATEDIFF(YEAR, @DateOfBirth, GETDATE())
END
```

2. **Inline Table-Valued Functions:**

- Return Type: Returns a table, but you define them using a single SELECT statement.
- Performance: Often perform better than multi-statement table-valued functions because they are essentially parameterized views and can be optimized better by the SQL Server query engine.
- Use Case: Useful when you need to return a table based on simple transformations or filters on existing tables.

Example:



SQLQuery_1.sql - uwc-studentsql.continuum.uw.edu\uwcitfdnsql.Assignment07DB_NUpadhyay (ITFdn130)

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Users > neetu > Desktop > Module07-Functions 3 > Assignment07 > SQLQuery_1.sql

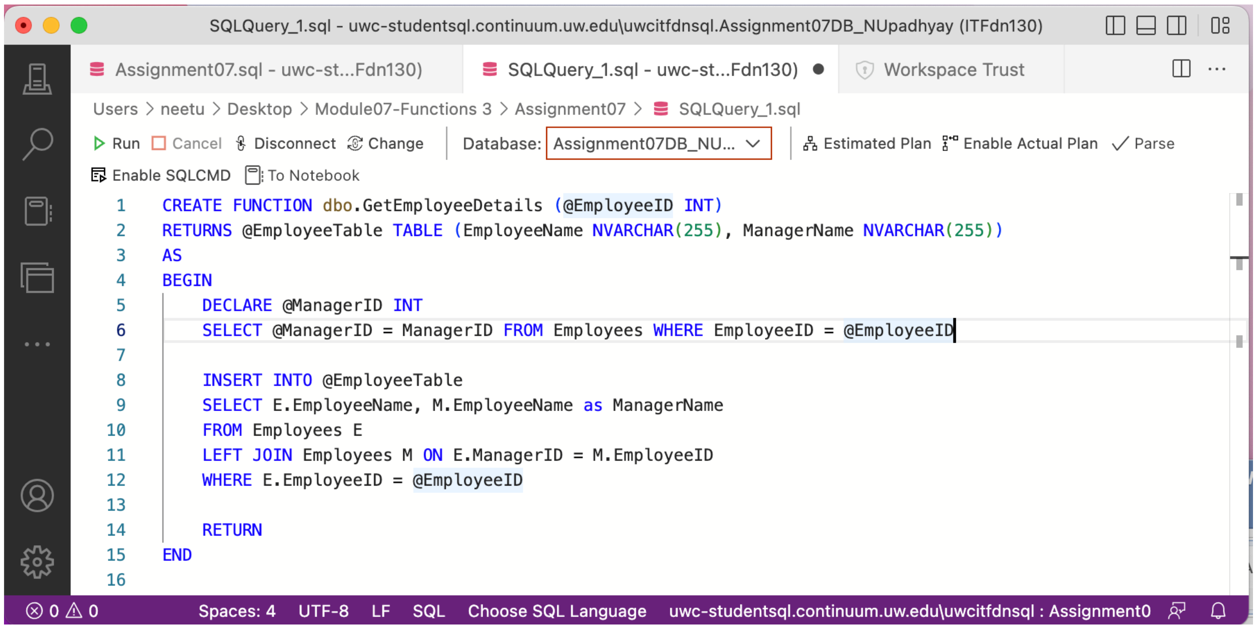
Database: Assignment07DB_NU...

```sql
CREATE FUNCTION dbo.GetEmployeesInDepartment (@DepartmentID INT)
RETURNS TABLE
AS
RETURN (
    SELECT EmployeeName, EmployeeID
    FROM Employees
    WHERE DepartmentID = @DepartmentID
)
```

3. Multi-Statement Table-Valued Functions:

- Return Type: Returns a table, but unlike inline table-valued functions, they can have multiple statements inside the function body.
- Performance: Typically slower than inline table-valued functions because SQL Server cannot optimize them as effectively. They often require a full table spool.
- Use Case: Useful when you need to perform complex logic, perhaps involving multiple steps or temporary storage, before arriving at the final table result.

Example:


SQLQuery_1.sql - uwc-studentsql.continuum.uw.edu\uwcitfdnsql.Assignment07DB_NUpadhyay (ITFdn130)
Assignment07.sql - uwc-st...Fdn130)
SQLQuery_1.sql - uwc-st...Fdn130)
Workspace Trust
Users > neetu > Desktop > Module07-Functions 3 > Assignment07 > SQLQuery_1.sql
Run
Cancel
Disconnect
Change
Database: Assignment07DB_NU...
Estimated Plan
Enable Actual Plan
Parse
Enable SQLCMD
To Notebook
⊗ 0 ⚠ 0
Spaces: 4
UTF-8
LF
SQL
Choose SQL Language
uwc-studentsql.continuum.uw.edu\uwcitfdnsql : Assignment0

```sql
CREATE FUNCTION dbo.GetEmployeeDetails (@EmployeeID INT)
RETURNS @EmployeeTable TABLE (EmployeeName NVARCHAR(255), ManagerName NVARCHAR(255))
AS
BEGIN
    DECLARE @ManagerID INT
    SELECT @ManagerID = ManagerID FROM Employees WHERE EmployeeID = @EmployeeID

    INSERT INTO @EmployeeTable
    SELECT E.EmployeeName, M.EmployeeName as ManagerName
    FROM Employees E
    LEFT JOIN Employees M ON E.ManagerID = M.EmployeeID
    WHERE E.EmployeeID = @EmployeeID

    RETURN
END
```

In summary:

- Scalar Functions return a single value.
- Inline Table-Valued Functions return a table and are defined with a single SELECT statement.
- Multi-Statement Table-Valued Functions return a table but can contain multiple T-SQL statements in their definition.

When creating UDFs, it's essential to understand their performance implications. While UDFs can make code more readable and modular, they can also impact query performance if not designed and used appropriately.

Summary

In "Module07-Functions", learners delve deep into the world of SQL functions, starting with a review of Aggregate Functions, which aid in data summarization. As we progress, we explore various Common Functions that enhance data selection and retrieval. The module also introduces Partitioned or Windowed Functions, empowering users to perform calculations across sets of table rows related to the current row. A significant focus is given to the utility of functions in generating insightful reports. Later, the emphasis shifts to User

Defined Functions (UDFs), which are custom functions tailored to specific needs. The use of UDFs in ensuring data integrity through Check constraints is also addressed. Finally, the module transitions into creating advanced GitHub pages, guiding learners on creating and formatting a Markdown file to ensure an appealing presentation of their work on GitHub.

Thanks!