```
-- Title: Assignment07
-- Author: Himanshi
-- Desc: This file demonstrates how to use Functions
-- 2024-05-31,HimanshiGupta,Created File
```

Explain when you would use a SQL UDF.

SQL UDFs (User-Defined Functions) are used to encapsulate logic that can be reused within SQL queries. They offer several advantages and can be used in various scenarios:

Encapsulating Business Logic: When you have complex business logic that needs to be applied repeatedly across different queries or stored procedures, UDFs can encapsulate this logic in a single function, making it easier to maintain and ensuring consistency across your database.

Reusability: UDFs promote code reusability. Once defined, you can call them from multiple places within your SQL code, reducing redundancy and promoting a modular approach to database development.

Abstraction: UDFs help abstract complex operations into simpler, more manageable functions. For instance, you might have a complex calculation that needs to be performed in multiple queries. By encapsulating this calculation in a UDF, you can hide the complexity from the main query, making it more readable and maintainable.

Performance Optimization: In some cases, using UDFs can improve query performance. For example, if you have a complex calculation that needs to be performed on each row of a large table, performing the calculation within a UDF can be more efficient than repeating the logic in every query that needs it.

Code Maintainability: By encapsulating logic in UDFs, you can improve code maintainability. If you need to update the logic, you only need to do it in one place (the UDF definition) rather than searching through multiple queries or procedures.

However, it's important to use UDFs judiciously, as they can also have drawbacks:

Performance Overhead: While UDFs can improve performance in some cases, they can also introduce performance overhead, especially if they involve complex operations or are used in certain contexts (e.g., in the WHERE clause of a query).

Function Inlining: In some database systems, UDFs may not be "inlined" into the query plan, meaning that the optimizer may not be able to optimize queries effectively, leading to suboptimal performance.

Indexing Limitations: UDFs can sometimes limit the ability to use indexes effectively, particularly if they involve non-deterministic operations or complex computations.

In summary, SQL UDFs are a powerful tool for encapsulating logic and promoting code reusability and maintainability in SQL queries. However, they should be used judiciously, taking into account performance considerations and potential limitations in specific database systems.

Explain are the differences between Scalar, Inline, and Multi-Statement Functions.

SQL:

1. **Scalar Functions**:
    - **Return Type**: Scalar functions return a single value.
    - **Usage**: They are commonly used to encapsulate simple calculations or transformations that operate on individual values.
    - **Syntax**: Scalar functions are defined with a single RETURN statement and can include input parameters.
    - **Example**: A function that calculates the square of a number or formats a string would be typical examples of scalar functions.
2. **Inline Table-Valued Functions (Inline Functions)**:
    - **Return Type**: Inline functions return a table variable or a table-like result set.
    - **Usage**: They are used to encapsulate more complex queries or operations that return a set of rows.
    - **Syntax**: Inline functions are defined with a RETURNS TABLE clause and typically include a single SELECT statement.
    - **Example**: An inline function might retrieve a subset of data from a table or perform a join operation and return the result set as a table.
3. **Multi-Statement Table-Valued Functions (Multi-Statement Functions)**:
    - **Return Type**: Multi-statement functions also return a table variable or a table-like result set.
    - **Usage**: They are used for more complex logic that cannot be achieved with a single SELECT statement, such as iterative processing or multiple operations.
    - **Syntax**: Multi-statement functions have a BEGIN...END block and use INSERT statements to populate a table variable.

- **Example**: A multi-statement function might involve looping through records or performing multiple queries before constructing the final result set.

In summary:

- Scalar functions return a single value and are suitable for simple calculations or transformations.
- Inline functions return a table-like result set and are used for more complex queries that can be expressed in a single SELECT statement.
- Multi-statement functions also return a table-like result set but are used for more complex logic that requires multiple statements or iterative processing.

Each type of function has its own strengths and use cases, so the choice of which to use depends on the specific requirements of your query or application.