**SQL Assignment 3**

1. Create a function and then call another function from within it. What is this process called?

A.

This process is commonly referred to as "function calling" or "calling a function within another function." When a function calls another function, it essentially invokes the execution of the second function from within the body of the first function. This allows for modular and organized code, as functions can be reused and composed to perform various tasks.

2.How to inspect the query's execution plan?

A. Inspecting the execution plan of a query is a crucial aspect of database optimization and performance tuning. The method to inspect the execution plan depends on the database management system (DBMS) you're using. Here are general steps for common database systems:

1. SQL Server:

Use the EXPLAIN keyword before your query, or use the SET SHOWPLAN\_ALL ON statement before executing your query. This will show you the execution plan without actually executing the query.

You can also use the SHOWPLAN\_XML option to get a more detailed execution plan in XML format.

Alternatively, you can use SQL Server Management Studio (SSMS) to display the execution plan graphically by clicking on "Include Actual Execution Plan" in the toolbar before executing your query.

2. Oracle:

Use the EXPLAIN PLAN FOR statement before your query. This will generate the execution plan without executing the query.

To view the execution plan, you can query the PLAN\_TABLE or use DBMS\_XPLAN.DISPLAY function.

SQL Developer and Oracle Enterprise Manager also provide graphical interfaces to view execution plans.

3. MySQL and MariaDB:

Use the EXPLAIN keyword before your query. This will show you the execution plan without actually executing the query.

You can also use EXPLAIN EXTENDED for additional information.

MySQL Workbench and other tools provide graphical interfaces to view execution plans.

4. PostgreSQL:

Use the EXPLAIN keyword before your query. This will show you the execution plan without actually executing the query.

You can also use EXPLAIN ANALYZE to execute the query and display the execution plan with actual runtime statistics.

PostgreSQL also provides various graphical tools for inspecting execution plans.

5. SQLite:

Use the EXPLAIN keyword before your query. This will show you the execution plan without actually executing the query.

SQLite does not have a graphical interface for execution plans by default, but you can use various third-party tools for this purpose.

General Tips:

Pay attention to indexes being used, join types, and whether full table scans are happening.

Look for potential bottlenecks such as high-cost operations or large intermediate result sets.

Experiment with different query structures and indexes to improve performance based on the insights gained from the execution plan.

3.What is the purpose of the MAXDOP and recompiling keywords in SQL queries?

A. MAXDOP (Maximum Degree of Parallelism) and query recompilation are both features in SQL Server that can affect query performance and behavior.

MAXDOP (Maximum Degree of Parallelism): This setting determines the maximum number of processors that can be used to execute a single query. By default, SQL Server allows parallel execution of queries, which means it can split a query into multiple tasks and execute them simultaneously across multiple CPU cores. However, in some cases, parallel execution can lead to contention for resources or inefficient query plans. MAXDOP allows you to limit the degree of parallelism to prevent such issues. By setting a specific MAXDOP value, you can control how many processors are used for executing a single query.

Recompilation: SQL Server generates execution plans for queries to determine the most efficient way to retrieve data. These plans are cached and reused to optimize query performance. However, there are scenarios where the cached execution plan may not be optimal due to changes in data distribution, schema, or statistics. In such cases, SQL Server may need to recompile the query to generate a new execution plan. This process is called query recompilation.

Recompiling: You can force SQL Server to recompile a query by using the OPTION (RECOMPILE) query hint. This can be useful when you know that the optimal execution plan for a query may change frequently due to volatile data or when the existing execution plan is not efficient.

Recompile hints: There are other hints you can use to influence query compilation, such as OPTION (RECOMPILE), OPTION (RECOMPILE, QUERYTRACEON 9481), etc. These hints provide SQL Server with information on how to compile the query, allowing you to control the behavior of query compilation and optimize performance.

MAXDOP helps control parallelism in query execution, while query recompilation allows for the generation of optimized execution plans based on current data and statistics. Both are important tools for optimizing SQL query performance.

4.How to build DDL statements from an existing database table, write steps for it?

A. Generating Data Definition Language (DDL) statements from an existing database table can be done in several steps. Here's a basic outline:

Identify the Database Type:

Determine which database management system (DBMS) you're working with (e.g., MySQL, PostgreSQL, SQL Server, etc.). The syntax for DDL statements can vary between different database systems.

Connect to the Database:

Use a database management tool or a command-line interface to connect to the database where the table resides.

Identify the Table:

Determine which table you want to generate DDL statements for.

Retrieve Table Structure:

Fetch the structure of the table. This typically includes column names, data types, constraints, indexes, etc.

Write DDL Statements:

Based on the retrieved structure, manually or programmatically write DDL statements to recreate the table. This involves:

Defining the table name.

Listing the columns with their respective data types and any constraints.

Specifying primary keys, foreign keys, unique constraints, etc.

Adding any indexes.

Handle Dependencies:

If the table has dependencies such as foreign key constraints referencing other tables, ensure that these dependencies are also included in the DDL statements.

Consider Special Cases:

Handle any special cases such as default values, auto-increment columns, partitioning, etc., as per the requirements.

Test the DDL Statements:

Before executing the DDL statements on the database, it's crucial to test them to ensure they accurately recreate the table structure without errors.

Execute the DDL Statements:

Once you're confident in the DDL statements, execute them against the database to create the new table structure.

Verify:

After executing the DDL statements, verify that the new table structure matches the original one and that any data or relationships are intact.

Document:

It's a good practice to document the DDL statements generated and executed for future reference or replication.

By following these steps, you can effectively generate DDL statements from an existing database table. If you're using a specific programming language or database tool, there might be libraries or features available to automate some of these steps.

5.How to update data in a table using an inner join, write an example?

A. Updating data in a table using an inner join in SQL is a common task.An example:

We have two tables: employees and salaries. We want to update the salary of employees who have been promoted. We'll use an inner join to match the employees in both tables and update their salary.

UPDATE employees

INNER JOIN salaries ON employees.employee\_id = salaries.employee\_id

SET salaries.salary = salaries.salary \* 1.1 -- increasing salary by 10% for promoted employees

WHERE employees.promoted = 1; -- Assuming 'promoted' is a flag indicating promotion.

UPDATE employees

INNER JOIN salaries ON employees.employee\_id = salaries.employee\_id

SET salaries.salary = salaries.salary \* 1.1 -- increasing salary by 10% for promoted employees

WHERE employees.promoted = 1; -- Assuming 'promoted' is a flag indicating promotion.

6.Differentiate between truncate, delete, and drop with a suitable example.

A. he context of databases, "truncate," "delete," and "drop" are SQL commands used to manipulate data or database objects. Let me explain each one with an example:

Truncate:

Truncate is used to remove all the rows from a table quickly and efficiently. It's a DDL (Data Definition Language) command.

It deallocates the data pages used to store the table's data, which makes it much faster than the DELETE command.

Truncate does not log individual row deletions, so it cannot be rolled back. It also resets any identity columns back to their seed value.

Example:

TRUNCATE TABLE employees;

This statement removes all rows from the "employees" table, but the table structure remains intact.

Delete:

DELETE is used to remove specific rows from a table based on a condition or without any condition.

It is a DML (Data Manipulation Language) command and can be rolled back (if in a transaction and if the database supports transactions).

DELETE operations are slower than TRUNCATE, especially on large tables, because they log individual row deletions.

Example:

DELETE FROM customers WHERE age > 50;

**This statement deletes all rows from the "customers" table where the age is greater than 50.**

**Drop:**

**DROP is used to remove objects like tables, views, indexes, etc., from the database.**

**It's a DDL (Data Definition Language) command.**

**When you drop a table, it removes the table definition and all its data, and you cannot roll back this operation.**

**Example:**

**DROP TABLE employees;**

This statement removes the entire "employees" table and all its data from the database schema.