SQL Stored Procedures: Theoretical Explanation

What is a Stored Procedure?

A **stored procedure** is a group of SQL statements that are stored and executed together as a unit. Once a stored procedure is created, you can invoke it multiple times with different parameters, making it a reusable component that simplifies complex operations.

Why Use Stored Procedures?

Stored procedures help automate repetitive tasks, ensure consistent execution of business rules, and improve performance by reducing network traffic and precompiling complex queries.

When to Use Stored Procedures?

- **Automating Repetitive Tasks**: Stored procedures are highly effective for automating routine tasks like processing payroll, generating reports, or updating data across multiple tables. For example, if you have a monthly payroll process that calculates salaries and deducts taxes, you can use a stored procedure to automate this rather than writing the same logic every month.
- **Encapsulating Business Logic**: Stored procedures can encapsulate complex business logic that might involve multiple database operations. For example, if a company needs to calculate customer loyalty points based on various factors like purchase history and feedback, a stored procedure can centralize this logic so that it's consistently applied.
- **Improving Performance**: Since stored procedures are precompiled and cached by the database, they execute faster than ad-hoc queries. This is especially useful for complex queries that are frequently executed. If your application often calculates a customer's total purchases for the year, you can create a stored procedure to do this efficiently.
- **Security**: Stored procedures can be used to enhance security. Instead of giving users direct access to tables, you can allow them to execute stored procedures that restrict what data they can access or modify. For example, instead of allowing a user to directly update the `employee` table, you can create a stored procedure that controls which columns they can update and enforces validation rules.

Real-Life Example:

Imagine an e-commerce platform where customers frequently place orders. Each time an order is placed, several tasks need to happen—updating inventory, sending a confirmation email, calculating shipping costs, and recording the order in the database. Instead of writing the same logic in multiple parts of the application, you can create a stored procedure to handle all these tasks in one place. Whenever an order is placed, the system simply calls the stored procedure, which performs all the necessary operations efficiently and consistently.