Statement Vs PreparedStatement

In Java, both `Statement` and `PreparedStatement` are used for executing SQL queries against a database, but they have key differences in how they are used and their benefits.

1. `Statement`:

- **Basic Use**: `Statement` is used for executing simple SQL queries without any parameters.
- **SQL Injection**: Since `Statement` directly concatenates the query string, it is prone to SQL injection attacks if user input is not properly sanitized.
- **Performance**: It does not provide any performance optimization. Every time the query is executed, it is sent to the database, which can lead to inefficiencies, especially when executing the same query multiple times.

Example:
```
Statement stmt = connection.createStatement();
String sql = "SELECT * FROM users WHERE username = '" + username + "' AND password = '"
+ password + "'";
ResultSet rs = stmt.executeQuery(sql);
```

2. `PreparedStatement`:

- **Basic Use**: `PreparedStatement` is used for executing SQL queries that contain placeholders (parameters) for dynamic values, which are then set at runtime. This allows for more secure and optimized queries.
- **SQL Injection**: It is safer than `Statement` because the SQL query is precompiled and parameters are bound separately, preventing SQL injection.
- **Performance**: `PreparedStatement` can be more efficient than `Statement` for repeated queries because the query is precompiled and can be reused. The database can cache the execution plan and reuse it for future executions of the same query.

Example:
```
String sql = "SELECT * FROM users WHERE username = ? AND password = ?";
PreparedStatement pstmt = connection.prepareStatement(sql);
pstmt.setString(1, username);
pstmt.setString(2, password);
ResultSet rs = pstmt.executeQuery();
```

Key Differences:

Feature	`Statement`	`PreparedStatement`
SQL Query	Cannot use placeholders (`?`)	Uses placeholders (`?`) for dynamic data
SQL Injection	Prone to SQL injection if not sanitized	Safe from SQL injection due to parameter binding
Performance	Slower for repeated queries	Faster for repeated queries (precompiled queries)
Use Case	Simple queries without user input	Queries with dynamic input (user data)

In summary:

- **Use `PreparedStatement`** when you need to execute a query with dynamic values and prioritize security and performance.
- **Use `Statement`** for simple queries or one-time executions without user input.