# Day-13 Interview Questions

**1. What is JDBC?**

JDBC is a Java API that is used to connect and execute the query to the database. JDBC API uses JDBC drivers to connect to the database. JDBC API can be used to access tabular data stored into any relational database.

**2. What is the return type of Class.forName() method?**

The Class.forName() method returns the object of java.lang.Class object.

**3. What is the role of the JDBC DriverManager class?**

The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver().

**4. What are the functions of the JDBC Connection interface?**

The **Connection interface** maintains a session with the database. It can be used for transaction management. It provides factory methods that return the instance of Statement, PreparedStatement, CallableStatement, and DatabaseMetaData.

**5. Which interface is responsible for transaction management in JDBC?**

The **Connection interface** provides methods for transaction management such as commit(), rollback() etc.

**6.What are the different types of JDBC drivers, and how do they differ? Answer: There are four types of JDBC drivers:**

**Type-1 driver (JDBC-ODBC bridge driver):** It uses native ODBC (Open Database Connectivity) drivers to connect to databases. It is platform-dependent and less commonly used.
**Type-2 driver (Native-API driver):** It uses database-specific native libraries to connect to databases. It provides better performance than Type-1 drivers but is still platform-dependent.
**Type-3 driver (Network Protocol driver):** It uses a middleware server to communicate with the database. It is platform-independent and offers good performance.

**Type-4 driver (Thin driver):** It is a pure Java driver that communicates directly with the database using the database's native protocol. It is platform-independent and commonly used in Java applications.

**7.What are the basic steps for establishing a database connection using JDBC?**

**The basic steps for establishing a database connection using JDBC are:**
Load the JDBC driver (e.g., Class.forName("com.mysql.cj.jdbc.Driver")).
Define the database URL, username, and password.
Create a connection using DriverManager.getConnection().
Create a Statement or PreparedStatement for executing SQL queries.
Execute SQL queries and process the results.
Close the resources (Statement, Connection) when done.

**8.What is the difference between Statement and PreparedStatement in JDBC?**

**Statement:** It is used for executing static SQL queries with no or minimal parameterization. It is prone to SQL injection and may be less efficient when executing the same query with different parameters.

**PreparedStatement:** It is used for executing dynamic SQL queries with parameters. It is more secure against SQL injection and can be more efficient when executing the same query with different parameter values.

**9.How do you prevent SQL injection in JDBC applications?**

To prevent SQL injection, you should use PreparedStatement instead of Statement. PreparedStatements automatically handle parameterization and escape input values, making it difficult for attackers to inject malicious SQL code.

**10.Explain the ResultSet interface in JDBC.**

ResultSet is an interface in JDBC that represents the result set of a database query. It provides methods for retrieving and manipulating data retrieved from the database. You can navigate through the result set, retrieve values, and check for NULL values using ResultSet methods.

**11.What are the common exceptions that can be thrown in JDBC, and how do you handle them?**
Common exceptions in JDBC include SQLException, ClassNotFoundException, and others. You can handle exceptions using try-catch blocks. SQLExceptions often provide

error codes and details that can help diagnose and handle database-related errors gracefully.

**12.How can we set null value in JDBC PreparedStatement?**

By using the setNull() method of the PreparedStatement interface, we can set the null value to an index. The syntax of the method is given below.

**void** setNull(**int** parameterIndex, **int** sqlType) **throws** SQLException

**13. What are the benefits of PreparedStatement over Statement?**

The benefits of using PreparedStatement over the Statement interface is given below.

The PreparedStatement performs faster as compared to Statement because the Statement needs to be compiled every time we run the code whereas the PreparedStatement is compiled once and then executed only on runtime.

- PreparedStatement can execute Parameterized query whereas Statement can only run static queries.
- The query used in PreparedStatement appears to be similar every time. Therefore, the database can reuse the previous access plan whereas, Statement inline the parameters into the String, therefore, the query doesn't appear to be the same every time which prevents cache reusage.