

Explain different JDBC driver.

There are several types of JDBC drivers that can be used to connect to a database:

1. Type 1 JDBC driver: Also known as the JDBC-ODBC bridge driver, this driver uses the ODBC driver to connect to the database. An example of this type of driver is the JDBC-ODBC Bridge driver that comes with the JDK.
2. Type 2 JDBC driver: Also known as the native-API driver, this driver uses the native API of the database management system (DBMS) to connect to the database. An example of this type of driver is the Oracle OCI Driver for Oracle Database.
3. Type 3 JDBC driver: Also known as the network-protocol driver, this driver uses a middleware component to connect to the database. An example of this type of driver is the DataDirect Connect for JDBC, which is a middleware that translates JDBC calls into a DBMS-independent network protocol.
4. Type 4 JDBC driver: Also known as the pure Java driver, this driver is written entirely in Java and communicates directly with the database through a socket connection. An example of this type of driver is the MySQL Connector/J driver for MySQL Database.


All these types of drivers have their own advantages and disadvantages, choosing the right one depends on the specific use case, and some are vendor specific, others are open-source and some are commercial.

How database connection can be established through JDBC.

To establish a database connection using JDBC, you need to follow these steps:

Step 1: Register the JDBC driver class


```
Class.forName("com.mysql.jdbc.Driver");
```

 Copy code

This step is used to dynamically load the JDBC driver class for the specific database management system you are connecting to.


Step 2: Create a connection object

```
Connection conn =  
DriverManager.getConnection("jdbc:mysql://hostname:port/dbname", "username",  
"password");
```

 Copy code

This step is used to establish the connection to the database using the `DriverManager.getConnection`` method, which requires a URL in the format ``jdbc:mysql://hostname:port/dbname``, the username, and the password to connect to the database.


Step 3: Create a statement object

 Copy code

```
Statement stmt = conn.createStatement();
```

This step is used to create a statement object by using the `connection.createStatement()`` method. This statement object is used to execute queries on the database.


Step 4: Execute the query

 Copy code

```
ResultSet rs = stmt.executeQuery("SELECT * FROM tableName");
```

This step is used to execute a query on the statement object by using the `statement.executeQuery()`` method. This method returns a `ResultSet` object which can be used to retrieve the data from the table.

Step 5: Close the connection

 Copy code

```
conn.close();
```

This step is used to close the connection by using the `connection.close()`` method. This will close the statement and `ResultSet` objects automatically.