Step 1 - Establish the connection

1. **Inbuilt method which is used to establish the connection is getConnection().**
2. **getConnection method is a static method present in DriverManager class / java.sql.DriverManager.**
3. **getConnection() is a static & overloaded method with two variations.**
    a. **public static Connection getConnection(String URL)**
    b. **public static Connection getConnection(String URL, String user, String password)**
4. **getConnection() takes URL as an argument, which consist of the information about database to be connected.**
5. **getConnection() throws SQLException which is checked exception and it is handle by try & catch block.**
6. **The return type of getConnetion() method is Connection Interface present in java.sql package.**
7. **Syntax:**

Connection connection = DriverManager.getConnection(String URL);

Interface reference variable	**Helper class present in**	**static & overloaded method for establishing connection**	
Present in Java.sql Package	**name**	**java.sql**	

Note:
Throwable class contains printstacktrace() which is non-static method & responsible to return detail information about exception like name, line number, reason, etc.

+: represent - public access specifier
- : represent - private access specifier
#: represent - protected access specifier

To setup java project for jdbc program:

Step 1: Access MySQL-connector jar file in project by using java-build-path which is present in jdbc software folder.
Step 2: import java.sql package to represent jdbc api in java class.
Step 3: Then write a code in main method.

Steps 2 - Creation of a platform.

1. JDBC programs consist of both java code and SQL queries. Java compiler can understand the syntax of java code only.
2. **Write query in the java application in the form of String.**
3. **To convert String object into SQL query, make use JDBC Drivers.**
4. SQL queries which is present in the JDBC program can be understood only within the database application.
5. **To make SQL queries reach to the database, programmer will be creating platform.**

   Note: The main purpose of platform is to carry SQL queries from Java application to Database.
6. **There are two types of platforms**
   a. **Statement type**
   b. **Prepared Statement type**
7. If query contains hardcoded values then programmer prefers statement type platform.
8. If query contains runtime values then programmer prefers Prepared Statement type platform.

Note:

9. **The values which are assigned by programmer is referred as 'Hardcoded values'.**
10. **For example: int a = 10;**
11. **The values which are taken from user with help of Scanner Class/HTML file is referred as 'Runtime Values'.**
12. **For example: int a = sc.nextInt();**

Statement Platform:

13. **If query contains hardcoded values, then programmers are preferring this type of platform.**
14. **For example:**

"insert into student values(101,'Ram',45)"

15. **Syntax for Statement Type -**

Statement statement = connection.createStatement();

Interface reference variable	**reference variable**	**Non-static method to create Statement type**
Present in Java.sql Package	**of Connection Interface**	**Platform & present in**

Step 3: - Execution of SQL Queries

1. **Programmer can perform two operations on the database are as follows.**
    a. **Write operation**
    b. **Read operation**
2. **Programmatically write operation is represent by insert, update & delete query.**
3. **To perform write operation on table, programmers are using executeUpdate().**
4. **Programmatically read operation is represent by select query.**
5. **To perform read operation on table, programmers are using executeQuery().**
6. **executeUpdate() & executeQuery() both are non-static methods present in platforms**

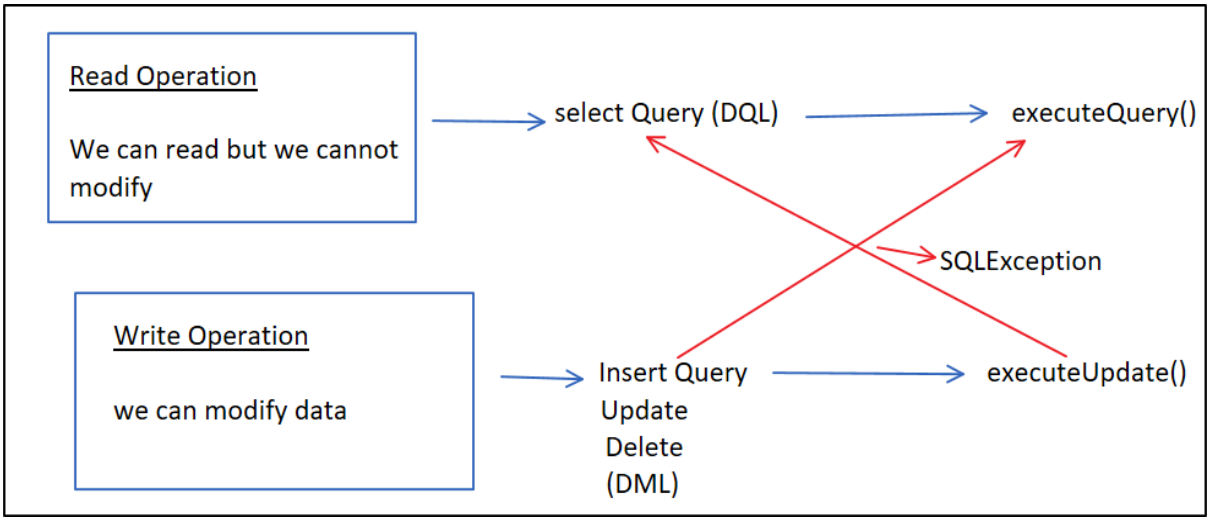
i.e., Statement Interface and PreparedStatement interface.

Note:

7. **executeUpdate() have return type as int. Its responsible to return no. of records get affected in a given table.**
8. **While working for statement type platform, we have to pass the query in step no. 3, which indicates that program consists of hardcoded values.**
9. **While working Prepared Statement type platform, we have to pass query in step2 of JDBC which indicates that program consists of runtime values.**

Reasons for SQLException:

10. **Incorrect URL.**
11. **Absence of MySQL-connector-jar file in respective java project**
12. **If there is mismatch between the type of operation & method used for its execution then it results in SQLException.**



Resultant data

1. **The data which is present in the table is referred as actual data.**
2. **Once query reaches Database, it performs three important functions.**
    a. **Compilation of Queries**
    b. **Execution of Queries**
    c. **Store output in a Buffer Memory**
3. **The output of read operation(select), after query execution is stored in buffer memory and it is called Resultant data.**
4. **The output of write operation (Insert, Update, Delete) is only to the Database table. Hence buffer memory contents will be empty i.e., programmers will not get resultant data for write operation.**

Note: To differentiate actual & resultant data, Database assigns BFR & ALR for resultant data where, BFR stands for Before first record and ALR stands for After last record.

Step4: Process of Resultant Data

5. Processing resultant data means verify or retrieve data from buffer memory in java application.
6. To perform above task, programmer prefers "ResultSet interface" present in java.sql package.
7. It contains following methods.
    a. isBeforeFirst()
    b. next()
    c. Getters
8. isBeforeFirst() is used to check whether Data is present in Buffer memory or not, return type of this method is boolean
9. executeQuery() method return type is ResultSet Interface.
10. next() method is used to shift cursor from one position to another. By default, cursor is present at BFR position of buffer memory.
11. It will check whether record is present at current position or not and returns boolean value, respectively.
12. To retrieve/fetch data after shifting of cursor, programmers will make use Getters based on column name & its type.
13. For example: If column name is "ename" and its type is "String" then call
    getString("ename") method.

Retrieve or Fetch Data

14. Getters and setters can be used without encapsulation as a normal methods like getters to fetch data & setters to assign/add data.
15. After execution of select query to verify whether records/ resultant data is present in a buffer memory, make use of "last()" method which is non-static method with return type Boolean.
16. It will shift cursor from starting position to last record present in buffer memory, if record is present at last position it will return "true" or else "false".

17. **To move cursor again back to Before First record position, make use of beforeFirst() method.**

Prepared Statement Platform:

- **As a programmer, we prefer Prepared Statement platform to pass runtime values, which results in flexibility in the code.**
- **To represent runtime values in query, make use of placeholder. It is denoted by '?'.**
- **To take runtime value from user, make use of Scanner class/HTML form.**
- **To assign those runtime values to the placeholder, make use of setter's method.**
- **For each type of value, we have separate setters.**
- **Setters' method consists of two arguments.**
    1. **Index of the placeholder to which runtime values has to be assign.**
    2. **Value/data to be assign.**
- **Syntax - pstmt.setString(int position, String data);**

- **The inbuilt method, used to create Prepared Statement platform is "prepareStatement(String query)" with return type PreparedStatement Interface.**

- **Syntax:**

  PreparedStatement preparedStatement = connection.prepareStatement(query);

Note:

- **While working for statement type platform, we have to pass the query in step no. 3, which indicates that program consists of hardcoded values.**
- **While working Prepared Statement type platform, we have to pass query in step2 of JDBC which indicates that program consists of runtime values.**

Stored Procedures

a. It is a method present in SQL which are used to store queries permanently in Database application.
b. It is set of SQL statements.
c. If query is present in stored procedure in order to compile & execute the query takes less time & increase the performance of an application.
d. Also, it is used to avoid repetition of a code.
e. For example:

```
CREATE PROCEDURE `databaseName`.`StoredProcedureName` ()
BEGIN

delete from employee where empId = 1002;

END$$
```

Stored Function:

1. It is a set of SQL statements to perform operation and returns a single value.
2. Syntax to create Stored Function:

```
CREATE FUNCTION `new_function` ()
RETURNS INTEGER
deterministic
BEGIN
    declare total integer;
    select count(*) into total from employee;
RETURN total;
END
```

Note: In order to execute the query which is present in a stored procedure & function, we need to call them from java application by using Callable Statement platform.

CallableStatement Platform

1. It is an interface present java.sql package.
2. As a programmer we make use of a CallableStatement platform to call stored procedure From Java application.
3. Syntax –

```
(Calling stored procedure)
call database.storedProcedureName

(Calling stored function)
{? = call storedFunctionName()}
```

4. To create CallableStatement platform, make use of prepareCall() method, as an argument it takes syntax of calling stored procedure.
5. Syntax

```
CallableStatement callableStatement =
    connection.prepareCall("call function/procedure");
```

Stored procedure with arguments:

1. It used to create stored procedure which is able to accepts runtimes values.
2. To create stored procedure with arguments, follow below syntax:

```
Create procedure databaseName.storedProcudureName(Argument1, Argument2, Argument)
Begin
        Insert into table Name values (Argument1, Argument2);
End
```

3. Syntax to declare arguments in stored procedure as follows:

```
IN Argument_Name Datatype (Size)
```

4. For example:

```
CREATE PROCEDURE
`jspiders`.`RetrieveStudentInfo` (IN RN INT(10))
BEGIN
select * from student where rollNo = RN;
END$$
```

5. To call stored procedure with argument from java application use following syntax:

```
Call database_Name.stored_Procedure_Name (?)
```

Note:

1. Make use of placeholder to provide values for stored procedure.
2. Use setters' method to assign values for placeholders.
3. Column name and argument name must & should be different.
4. Programmers can also create Stored Function with argument To take runtime values from user, like Stored Procedure.

Difference between Stored procedure & Function.

Stored Procedure	Stored Function
1. Returns single, multiple values and zero.	1. Returns single value. Return type is mandatory.
2. We can call functions from the procedure.	3. Procedure cannot be called from function.
4. Procedure supports input & output parameters.	5. Function supports only input parameters.
6. Transaction management is possible.	7. Transaction management is not possible.
8. Allows both DML & DQL	9. Allows only DQL statements,
10. Procedures does not require repeatedly compilation.	11. Needs to be compiled and execute every time.

Batch Processing:

1. It allows you to group related SQL statements into a batch and submit them with one call to the database.
2. Generally, we use batch processing, whenever we have multiple queries which needs to be executed in one call.
3. Here multiple queries are going to be collected together and we are going to execute all the queries at once.
4. It makes the performance fast.
5. Helps in increasing the data consistency.
6. Make use of addBatch() method to add multiple queries into a batch. It is present in Statement/PreparedStatement interface.
7. Call executeBatch() to execute batch which returns int[], where each index contains updated count.
8. It supports only DML queries.

Database Metadata:

1. **Metadata is an information about data itself.**
2. **Metadata summaries basic information about data, making finding and working with particular instances of data easier.**
3. **DatabaseMetaData is an interface that provides a variety of methods to obtain comprehensive information about the DB.**
4. **There are different methods available to get the metadata.**
    a. **getTables()**
    b. **getColumns()**
    c. **getUserName()**
    d. **getSchemas()**
    e. **getDatabaseProductName()**
    f. **getDatabaseProductVersion()**
    g. **getDriverName()**
    h. **getDriverVersiong()**
5. **To get object of Database call getMetadata() method from Connection interface, return of this method is DatabaseMetaData.**