1. ***Preparecall***

*The PreparedStatement interface extends the Statement interface, which gives you added functionality with a couple of advantages over a generic Statement object.This statement gives you the flexibility of supplying arguments dynamically.*

1. ***Resultsetmetadata***

*The metadata means data about data i.e. we can get further information from the data.If you have to get metadata of a table like total number of column, column name, column type etc. ResultSetMetaData interface is useful because it provides methods to get metadata from the ResultSet object.*

1. ***driver types***

***Type 1 − JDBC-ODBC Bridge Driver***

*In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.*

## *Type 2 − JDBC-Native API*

## *In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls, which are unique to the database. These drivers are typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge. The vendor-specific driver must be installed on each client machine.*

## *Type 3 − JDBC-Net pure Java*

## *In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.* *This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.*

## *Type 4 − 100% Pure Java*

## *In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself. Your application server might use a Type 1, 2, or 4 driver to communicate with the database, understanding the nuances will prove helpful.*

1. ***Sqlexception***

*In JDBC, we may get exceptions when we execute or create the query. Exceptions that occur due to the Database or Driver come under SQL Exception. Using Exception handling, we can handle the SQL Exception like we handle the normal exception. SQLException is available in the java.sql package. It extends the Exception class which means that we can use the methods available in the Exception class in the SQLException class as well.*

1. ***execute and executequery***

*execute: Returns true if the first object that the query returns is a ResultSet object. Use this method if the query could return one or more ResultSet objects.*

*executeQuery: Returns one ResultSet object.*

1. ***ACID properties***

*Atomicity. The term atomicity is the ACID Property in DBMS that refers to the fact that the data is kept atomic.*

*Consistency. This ACID Property will verify that the total sum of seats left in the train+sum of seats booked by users=total number of seats present in the train.*

*Isolation. Isolation is defined as a state of separation. Isolation is an ACID Property in DBMS where no data from one database should impact the other and where many ...*

*Durability. The ACID Property durability in DBMS refers to the fact that if an operation is completed successfully, the database remains permanent in the disk.*

1. ***pagination in jdbc***

***You can't do pagination with just JDBC methods. It's just something that JDBC isn't meant to handle****(it's meant to connect to databases, execute your queries blindly and return results). You'll have to work it out with queries, or as Tobias suggested, using a framework that gives you pagination out of the box.*