What is **PreparedStatement** ?
==========

1. **PreparedStatement** is an interface which is available in the package java.sql and it itself inherits from **Statement** interface.

2. It is provided by Java for executing **DYNAMIC SQL QUERIES.**

3. To use **PreparedStatement** we have to follow 3 steps:

a) Get the object of **PreparedStatement** interface by calling a method **prepareStatement()** which itself belongs to **Connection** inteface and has the following prototype:

```
public PreparedStatement prepareStatement(String) throws SQLException
```

Example
=======

```
PreparedStatement ps=conn.prepareStatement("insert into allmovies values(?,?,?)");
```

b) Replace the placeholders with actual values by calling setter methods of PreparedStatement object.

Example
=======

```
int id=kb.nextInt();
String mname=.....
int rating=kb.nextInt();
ps.setInt(1,id);
ps.setString(2,mname);
ps.setInt(3, rating);
```

c) Once we have builtup the query the final task is to pass this query to database for execution and this is done by calling the methods **executeUpdate()** or **executeQuery()** of PreparedStatement object.

```
public int executeUpdate() throws SQLException
public ResultSet executeQuery() throws SQLException
```

Statement	**PreparedStatement**
1. Used for executing static queries and not preferred for dynamic queries.	1. Used for executing dynamic queries.
2. We use it when we want to execute a particular SQL query ONLY ONCE. For examle: **select * from allmovies;**	2. We use it when we want to execute same SQL query but with different values MULTIPLE TIMES.
3. Statement interface does not support placeholders or parameters.	3. PreparedStatement support placeholders through ? and parameters through setters.
4. It is preferred for executing DDL queries because generally a DDL query is executed one like **create table**.	4. We use PreparedStatement to use dynamic DML or DQL queries.
5. Statement has a low performance.	5. PreparedStatement has very high performance.

How To Retrieve Date Values From DB ?

1. To retrieve dates from the database we call the method getDate() belonging to ResultSet object.

2. The method **getDate()** is overloaded and accept either column position or column name as argument and returns an object of **java.sql.Date.**

3. The prototype of the method is :

public Date getDate(int) throws SQLException

public Date getDate(String) throws SQLException

4. But when we display these date values Java shows them in the pattern **"yyyy-mm-dd"** which is not user friendly.

5. To solve this issue Java allows us to format date values as per our choice and this is done by using the class **SimpleDateFormat** available in the package **java.text.**

```
Statement st=conn.createStatement();
ResultSet rs=st.executeQuery("Select ename,hiredate from employees");
SimpleDateFormat sdf=new SimpleDateFormat("dd/MMM/yy, EEE");
while(rs.next()){
    String name=rs.getString(1);
    Date hdate=rs.getDate(2);
    String dateStr=sdf.format(hdate);
    System.out.println(name+"\t"+dateStr);
}
```