Kinldy go through
      a. SpringCore(XML,Annotation,PureJava)
      b. JDBC(Statement,PreparedStatement)
      c. ORM(hibernate:: Configuration,SessionFactory,Session,Transacation)

++++++++++++++++++++
Basics of SpringBoot
++++++++++++++++++++
      a. SpringBoot is a mechanism to create SpringRelated projects in easy way.
      b. SpringBoot supports Autoconfiguration for all the types of projects.
      c. SpringBoot application is created using 2 approaches
            a. Maven
            b. Gradle

++++++++++++++++++++++++++++++++++
Benefits of Working with SpringBoot
++++++++++++++++++++++++++++++++++

eg:: SpringJDBC

+++++++++++++++++
XML Configuration
+++++++++++++++++
```
<bean id = "dataSource" class =
"org.springframework.jdbc.datasource.DriverManagerDataSource">
   <property name = "driverClassName" value = "com.mysql.cj.jdbc.Driver"/>
   <property name = "url" value = "jdbc:mysql://localhost:3306/PWSKILLS"/>
   <property name = "username" value = "root"/>
   <property name = "password" value = "root123"/>
</bean>
```

Maven depedancy
+++++++++++++++
```
<dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-jdbc</artifactId>
         <version>5.3.14</version>
</dependency>
```

++++++++++++++++++++++++++++
Working with SpringBoot Style
++++++++++++++++++++++++++++

application.properties
++++++++++++++++++++++
```
spring.datasource.url=jdbc:mysql://localhost:3306/PWSKILLS
spring.datasource.username=root
spring.datasource.password=root123
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
```

+++++++++++++++++
Maven Dependancy
+++++++++++++++++
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jdbc</artifactId>
</dependency>
```

```
Project to perform CURD operation using SpringJDBC
++++++++++++++++++++++++++++++++++++++++++++++++
application.properties
      spring.datasource.url=jdbc:mysql:///pwskillsbatch
      spring.datasource.username=root
      spring.datasource.password=root123

pom.xml
+++++++
<dependencies>
      <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jdbc</artifactId>
      </dependency>

      <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
      <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
      </dependency>
</dependencies>


Code to perform Insert Operation
      @Autowired
      private JdbcTemplate jdbcTemplate;
      int rowAffected = jdbcTemplate.update("insert into
books(name,cost)values(?,?)",book.getName(),book.getCost());

Code to perform update operation
      @Autowired
      private JdbcTemplate jdbcTemplate;
      int rowAffected = jdbcTemplate.update("update books set cost=? where id=?",
book.getCost(), book.getId());

Code to perform Delete Operation
      @Autowired
      private JdbcTemplate jdbcTemplate;
      return jdbcTemplate.update("delete from books where id = ?",id);

Code to perform select Operation using primary key
      @Autowired
      private JdbcTemplate jdbcTemplate;
      return jdbcTemplate.queryForObject("select id,name,cost from books where id =
?",
                                    new Object[] { id },
                                 (rs,rowNum) -> Optional.of(new
Book(rs.getInt(1),rs.getString(2),rs.getInt(3))));


Code to Perform select operation(retrieve all the records)

      @Autowired
      private JdbcTemplate jdbcTemplate;
      return jdbcTemplate.query("select * from books",

                        new RowMapper<Book>() {
```

```
                              @Override
                              public Book mapRow(ResultSet rs, int rowNum) throws
SQLException {
                                    System.out.println(rowNum);
                                    return new Book(rs.getInt(1), rs.getString(2),
rs.getInt(3));
                              }
                        }

            );
```

Code to count no of records

```
      @Autowired
      private JdbcTemplate jdbcTemplate;
      return jdbcTemplate.queryForObject("select count(*) from books",
Integer.class);
```

Code to retrieve records based on name and price

```
      @Autowired
      private JdbcTemplate jdbcTemplate;
      return jdbcTemplate.query("select * from books where name like ? and
cost<=?",
                        new Object[] {"%"+name+"%",price}, (rs, rowNum) -> new
Book(rs.getInt(1), rs.getString(2), rs.getInt(3)));
```

++++++++++++++++++++++++++++++++++++++
Working with NamedParameterJdbcTemplate
++++++++++++++++++++++++++++++++++++++

Code to update the record

```
      @Autowired
      private NamedParameterJdbcTemplate namedParameterTemplate;
      return jdbcTemplate.update("update books set cost =:cost where id=:id ", new
BeanPropertySqlParameterSource(book));
```

Code to read the record

```
      @Autowired
      private NamedParameterJdbcTemplate namedParameterTemplate;
      return namedParameterTemplate.queryForObject("select * from books where id
=:id", new MapSqlParameterSource("id", id),
                        (rs, rowNum) -> Optional.of(new Book(rs.getInt(1),
rs.getString(2), rs.getInt(3))));
```