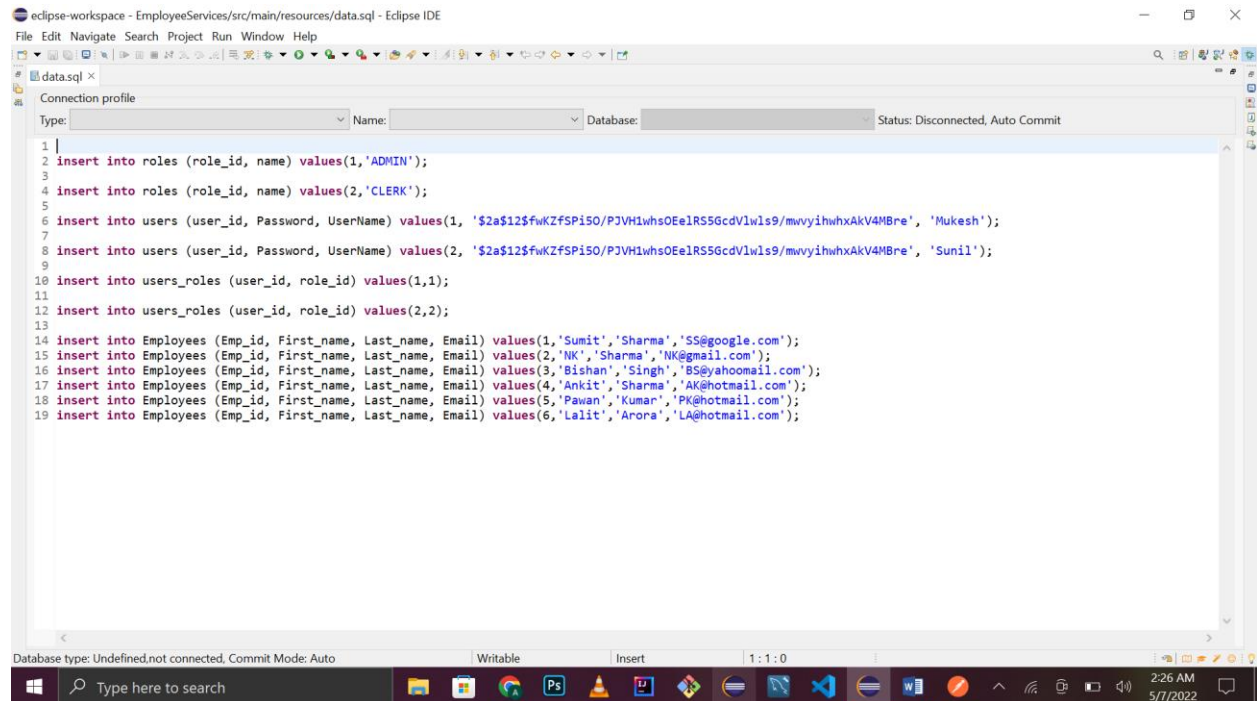


**BACK-END ASSIGNMENT 6**  
**EMPLOYEE SERVICE**  
**USING**  
**SPRING BOOT, JPA/ORM, SECURITY**

**SNAPSHOTS**  
**FOR THE CRUD OPERATION & MORE**  
**USING SPRING BOOT PROJECT**

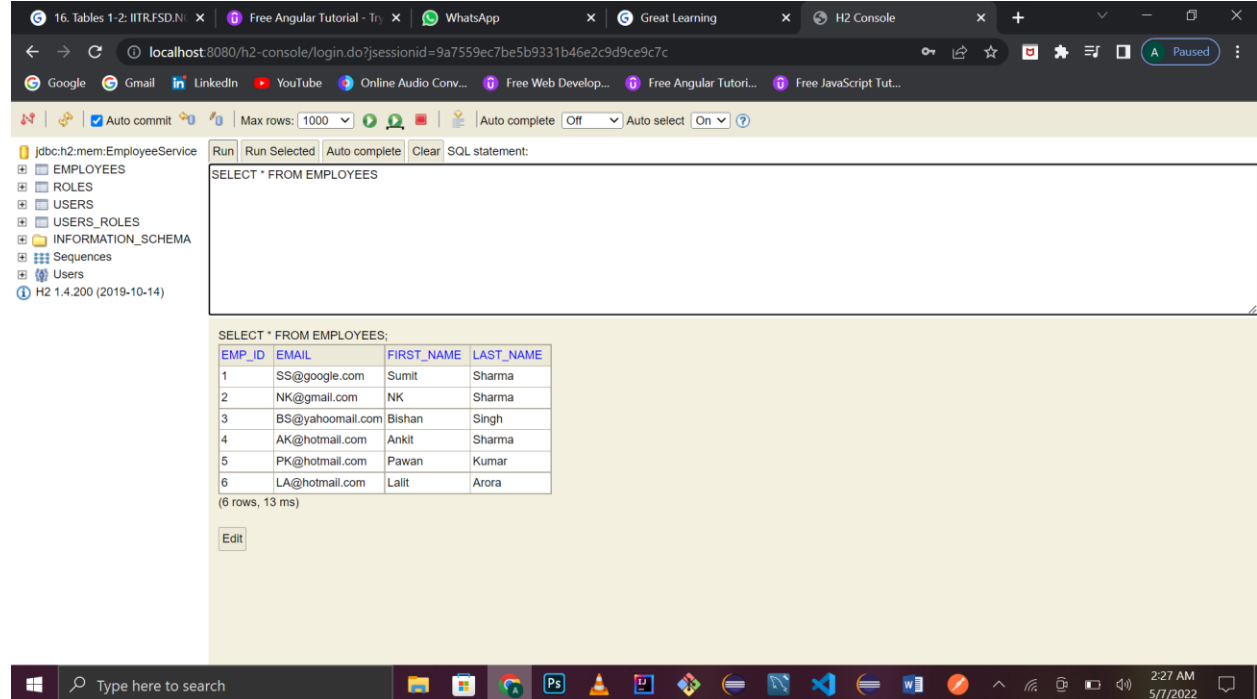
## DATA.SQL



```
1  
2 insert into roles (role_id, name) values(1,'ADMIN');  
3  
4 insert into roles (role_id, name) values(2,'CLERK');  
5  
6 insert into users (user_id, Password, UserName) values(1, '$2a$12$fWkZfSPi5O/P3VH1whsOEelRS5GcdV1wls9/mwvyihwhxAkV4MBre', 'Mukesh');  
7  
8 insert into users (user_id, Password, UserName) values(2, '$2a$12$fWkZfSPi5O/P3VH1whsOEelRS5GcdV1wls9/mwvyihwhxAkV4MBre', 'Sunil');  
9  
10 insert into users_roles (user_id, role_id) values(1,1);  
11  
12 insert into users_roles (user_id, role_id) values(2,2);  
13  
14 insert into Employees (Emp_id, First_name, Last_name, Email) values(1,'Sumit','Sharma','SS@google.com');  
15 insert into Employees (Emp_id, First_name, Last_name, Email) values(2,'NK','Sharma','NK@gmail.com');  
16 insert into Employees (Emp_id, First_name, Last_name, Email) values(3,'Bishan','Singh','BS@yahoo.com');  
17 insert into Employees (Emp_id, First_name, Last_name, Email) values(4,'Ankit','Sharma','AK@hotmail.com');  
18 insert into Employees (Emp_id, First_name, Last_name, Email) values(5,'Pawan','Kumar','PK@hotmail.com');  
19 insert into Employees (Emp_id, First_name, Last_name, Email) values(6,'Lalit','Arora','LA@hotmail.com');
```

## H2-CONSOLE

### EMPLOYEE TABLE



16. Tables 1-2: ITRFSD.N: x | Free Angular Tutorial - Tr x | WhatsApp x | Great Learning x | H2 Console x | +

localhost:8080/h2-console/login.do?sessionId=9a7559ec7be5b9331b46e2c9d9ce9c7c

Google | Gmail | LinkedIn | YouTube | Online Audio Conv... | Free Web Develop... | Free Angular Tutori... | Free JavaScript Tut...

Auto commit | Max rows: 1000 | Auto complete: Off | Auto select: On

jdbc:h2:mem:EmployeeService

Run | Run Selected | Auto complete | Clear | SQL statement:

SELECT \* FROM EMPLOYEES

SELECT \* FROM EMPLOYEES:

EMP_ID	EMAIL	FIRST_NAME	LAST_NAME
1	SS@google.com	Sumit	Sharma
2	NK@gmail.com	NK	Sharma
3	BS@yahoo.com	Bishan	Singh
4	AK@hotmail.com	Ankit	Sharma
5	PK@hotmail.com	Pawan	Kumar
6	LA@hotmail.com	Lalit	Arora

(6 rows, 13 ms)

Edit

H2 1.4.200 (2019-10-14)

## USERS TABLE

The screenshot shows the H2 Console interface. The left sidebar displays the database schema with tables: EMPLOYEES, ROLES, USERS, USERS\_ROLES, INFORMATION\_SCHEMA, Sequences, and Users. The main area shows the SQL statement `SELECT * FROM USERS;` and its results. The results are displayed in a table with columns USER\_ID, PASSWORD, and USERNAME. There are 2 rows of data.

USER_ID	PASSWORD	USERNAME
1	\$2a\$12\$fwKZISPI5O/PJVH1whsOEelRS5GcdVlws9/mwvylhwhxAkV4MBre	Mukesh
2	\$2a\$12\$fwKZISPI5O/PJVH1whsOEelRS5GcdVlws9/mwvylhwhxAkV4MBre	Sunil

(2 rows, 8 ms)

## ROLES TABLE

The screenshot shows the H2 Console interface. The left sidebar displays the database schema with tables: EMPLOYEES, ROLES, USERS, USERS\_ROLES, INFORMATION\_SCHEMA, Sequences, and Users. The main area shows the SQL statement `SELECT * FROM ROLES;` and its results. The results are displayed in a table with columns ROLE\_ID and NAME. There are 2 rows of data.

ROLE_ID	NAME
1	ADMIN
2	CLERK

(2 rows, 10 ms)

## USERS\_ROLES TABLE

The screenshot shows the H2 Console interface. The left sidebar lists the database schema: jdbc:h2:mem:EmployeeService, EMPLOYEES, ROLES, USERS, USERS\_ROLES, INFORMATION\_SCHEMA, Sequences, and Users. The main area displays the SQL statement `SELECT * FROM USERS_ROLES;` and its results. The results are shown in a table with two columns: USER\_ID and ROLE\_ID. The data contains two rows: (1, 1) and (2, 2). Below the table, it indicates "(2 rows, 9 ms)".

USER_ID	ROLE_ID
1	1
2	2

## GET-API to fetch list of all Employees from the database

The screenshot shows the Postman interface. The request is a GET to `localhost:8080/api/employees` using Basic Auth. The username is "Mukesh" and the password is "123456". The response is a JSON object with the following structure:

```
{
  "id": 1,
  "firstName": "Sumit",
  "lastName": "Sharma",
  "email": "SS@google.com"
}
```

The status is 200 OK, Time: 114 ms, Size: 851 B.

## OUTPUT

```
[
  {
    "id": 1,
    "firstName": "Sumit",
    "lastName": "Sharma",
    "email": "SS@google.com"
  },
  {
    "id": 2,
    "firstName": "NK",
    "lastName": "Sharma",
    "email": "NK@gmail.com"
  },
  {
    "id": 3,
    "firstName": "Bishan",
    "lastName": "Singh",
    "email": "BS@yahoo.com"
  },
  {
    "id": 4,
    "firstName": "Ankit",
    "lastName": "Sharma",
    "email": "AK@hotmail.com"
  },
  {
    "id": 5,
    "firstName": "Pawan",
    "lastName": "Kumar",
    "email": "PK@hotmail.com"
  },
  {
    "id": 6,
    "firstName": "Lalit",
    "lastName": "Arora",
    "email": "LA@hotmail.com"
  }
]
```

]

### GET-API to Fetch record of an Employee Using Employeeid

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and navigation tabs (Home, Workspaces, API Network, Reports, Explore). A search bar and an 'Invite' button are also present. The main workspace displays a GET request to the endpoint `localhost:8080/api/employees/3`. The request is configured with Basic Authentication, where the username is 'Mukesh' and the password is '123456'. The 'Show Password' checkbox is checked. The response status is 200 OK, with a time of 86 ms and a size of 487 B. The response body is displayed in JSON format, showing the details of the employee with ID 3: `{ "id": 3, "firstName": "Bishan", "lastName": "Singh", "email": "BS@yahoo.com" }`. The bottom status bar shows the system clock as 2:32 AM on 5/1/2022.

### POST-API to Add a New Employee to the Database

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and navigation tabs (Home, Workspaces, API Network, Reports, Explore). A search bar and an 'Invite' button are also present. The main workspace displays a POST request to the endpoint `localhost:8080/api/employees`. The request is configured with a JSON body containing the details of a new employee: `{ "firstName": "Sushil", "lastName": "Vats", "email": "SV@yahoo.com" }`. The response status is 200 OK, with a time of 348 ms and a size of 486 B. The response body is displayed in JSON format, showing the details of the newly created employee with ID 7: `{ "id": 7, "firstName": "Sushil", "lastName": "Vats", "email": "SV@yahoo.com" }`. The bottom status bar shows the system clock as 2:35 AM on 5/1/2022.

## PUT-API to Update the Record of an Existing Employee in the Database

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and navigation tabs (Home, Workspaces, API Network, Reports, Explore). A search bar and utility buttons (Invite, Upgrade) are also present. The main workspace displays a PUT request to the endpoint `localhost:8080/api/employees/7`. The request body is a JSON object: `{ "id": "7", "firstName": "Sushil Kumar", "lastName": "Vats", "email": "SV@yahoo.com" }`. The response status is 200 OK, with a time of 66 ms and a size of 492 B. The response body is displayed in JSON format: `{ "id": 7, "firstName": "Sushil Kumar", "lastName": "Vats", "email": "SV@yahoo.com" }`.

## DELETE-API to Delete an Existing record from the Database

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and navigation tabs (Home, Workspaces, API Network, Reports, Explore). A search bar and utility buttons (Invite, Upgrade) are also present. The main workspace displays a DELETE request to the endpoint `localhost:8080/api/employees/7`. The response status is 200 OK, with a time of 39 ms and a size of 434 B. The response body is displayed in Text format: `1 Deleted employee id -7`.

## GET-API to Search an Employee Record from the Database using FirstName

Postman interface showing a GET request to `localhost:8080/api/employees/search/Pawan`. The request is configured with Basic Auth (Username: Mukesh, Password: 123456). The response is a JSON object containing employee details for Pawan.

```
{  "id": 5,  "firstName": "Pawan",  "lastName": "Kumar",  "email": "PK@hotmail.com"}
```

## GET-API to Sort the Existing Records in Ascending Order using parameter as Employee FirstName

Postman interface showing a GET request to `localhost:8080/api/employees/sort?order=Asc`. The request is configured with Query Params (order: Asc). The response is a JSON object containing employee details for Ankit.

KEY	VALUE	DESCRIPTION
order	Asc	
Key	Value	Description

```
{  "id": 4,  "firstName": "Ankit",  "lastName": "Sharma",  "email": "AK@hotmail.com"}
```



### Output as per Order=Asc

```
[
  {
    "id": 4,
    "firstName": "Ankit",
    "lastName": "Sharma",
    "email": "AK@hotmail.com"
  },
  {
    "id": 3,
    "firstName": "Bishan",
    "lastName": "Singh",
    "email": "BS@yahoo.com"
  },
  {
    "id": 6,
    "firstName": "Lalit",
    "lastName": "Arora",
    "email": "LA@hotmail.com"
  },
  {
    "id": 2,
    "firstName": "NK",
    "lastName": "Sharma",
    "email": "NK@gmail.com"
  },
  {
    "id": 5,
    "firstName": "Pawan",
```

```
"lastName": "Kumar",  
"email": "PK@hotmail.com"  
},  
{  
  "id": 1,  
  "firstName": "Sumit",  
  "lastName": "Sharma",  
  "email": "SS@google.com"  
}  
]
```

GET-API to Sort the Existing Records in Descending Order using parameter as Employee FirstName

The screenshot shows the Postman interface with a GET request to `localhost:8080/api/employees/sort?order=desc`. The response is a JSON array with two employee records. The first record has `id: 1`, `firstName: Sumit`, `lastName: Sharma`, and `email: SS@google.com`. The second record has `id: 2`, `firstName: Kumar`, and `email: PK@hotmail.com`.

KEY	VALUE	DESCRIPTION
order	desc	
Key	Value	Description

```
1 {  
2   "id": 1,  
3   "firstName": "Sumit",  
4   "lastName": "Sharma",  
5   "email": "SS@google.com"  
6 }  
7 ]
```

**Output as per Order=Desc**

```
[  
  {  
    "id": 1,  
    "firstName": "Sumit",  
    "lastName": "Sharma",  
    "email": "SS@google.com"  
  },  
  {  
    "id": 5,  
    "firstName": "Pawan",  
    "lastName": "Kumar",  
    "email": "PK@hotmail.com"  
  },  
  {  
    "id": 2,  
    "firstName": "NK",  
    "lastName": "Sharma",  
    "email": "NK@gmail.com"  
  },  
  {  
    "id": 6,  
    "firstName": "Lalit",  
    "lastName": "Arora",  
    "email": "LA@hotmail.com"  
  },  
  {  
    "id": 3,  
    "firstName": "Bishan",
```

```

    "lastName": "Singh",
    "email": "BS@yahoo.com"
  },
  {
    "id": 4,
    "firstName": "Ankit",
    "lastName": "Sharma",
    "email": "AK@hotmail.com"
  }
]

```

**POST-API to Add a New User to the USER database**

The screenshot shows the Postman interface with a POST request to `localhost:8080/api/user`. The request body is a JSON object:

```

{
  "username": "Grant",
  "password": "123456",
  "roles": [
    {
      "id": "1",
      "name": "ADMIN"
    }
  ]
}

```

The response status is `200 OK` with a time of `386 ms` and a size of `547 B`. The response body is a JSON object:

```

{
  "id": 3,
  "username": "Grant",
  "password": "$2a$10$13B1Z.am2yqCSKcGFmmy8u06zNS8az1Gsng4uVRL1voABybdqERW",
  "roles": [
    {
      "id": "1",
      "name": "ADMIN"
    }
  ]
}

```

## POST-API to Add a New Role to the ROLE database

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and navigation tabs (Home, Workspaces, API Network, Reports, Explore). A search bar and utility buttons (Invite, Upgrade) are also present.

The main workspace displays a collection named "localhost:8080/api/role". The selected item is a POST request to "localhost:8080/api/role". The request body is set to "raw" and "JSON". The body content is:

```
1 {
2   "name": "MANAGER"
3 }
```

The response section shows a status of "200 OK", a time of "59 ms", and a size of "437 B". The response body is displayed in "Pretty" format:

```
1 {
2   "id": 3,
3   "name": "MANAGER"
4 }
```

The bottom of the screen shows the Windows taskbar with the search bar and various application icons. The system clock indicates 2:55 AM on 5/7/2022.

## Project Structure

The screenshot displays the Eclipse IDE interface for a project named 'EmployeeServices'. The Project Explorer on the left shows the project structure, with a red circle highlighting the 'src/main/java' directory. The main editor displays the 'pom.xml' file, which is a Maven POM for a Spring Boot project. The POM includes dependencies for Spring Boot starter-parent, spring-boot-starter-data-jpa, spring-boot-starter-security, and spring-boot-starter-web. The bottom status bar shows the time as 2:56 AM on 5/1/2022.

**Project Structure (src/main/java):**

- com.springboot.rest
  - EmployeeServicesApplication.java
  - EmployeeRestController.java
  - EmployeeRestEntity.java
  - Employee.java
  - Role.java
  - User.java
- com.springboot.rest.repository
  - EmployeeRepository.java
  - RoleRepository.java
  - UserRepository.java
- com.springboot.rest.security
  - MyUserDetails.java
  - WebSecurityConfig.java
- com.springboot.rest.service
  - EmployeeService.java
  - EmployeeServiceImpl.java
  - UserDetailsServiceImpl.java
- src/main/resources
  - static
  - templates
  - application.properties
  - data.sql
- src/test/java
- JRE System Library [JavaSE-11]
- Maven Dependencies
- src
- target

**EmployeeServices/pom.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocati
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.6.7</version>
  <relativePath /> <!-- lookup parent from repository -->
</parent>
<groupId>com.springboot.rest</groupId>
<artifactId>EmployeeServices</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>EmployeeServices</name>
<description>Restful Project Using Spring Boot</description>
<properties>
  <java.version>11</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
```