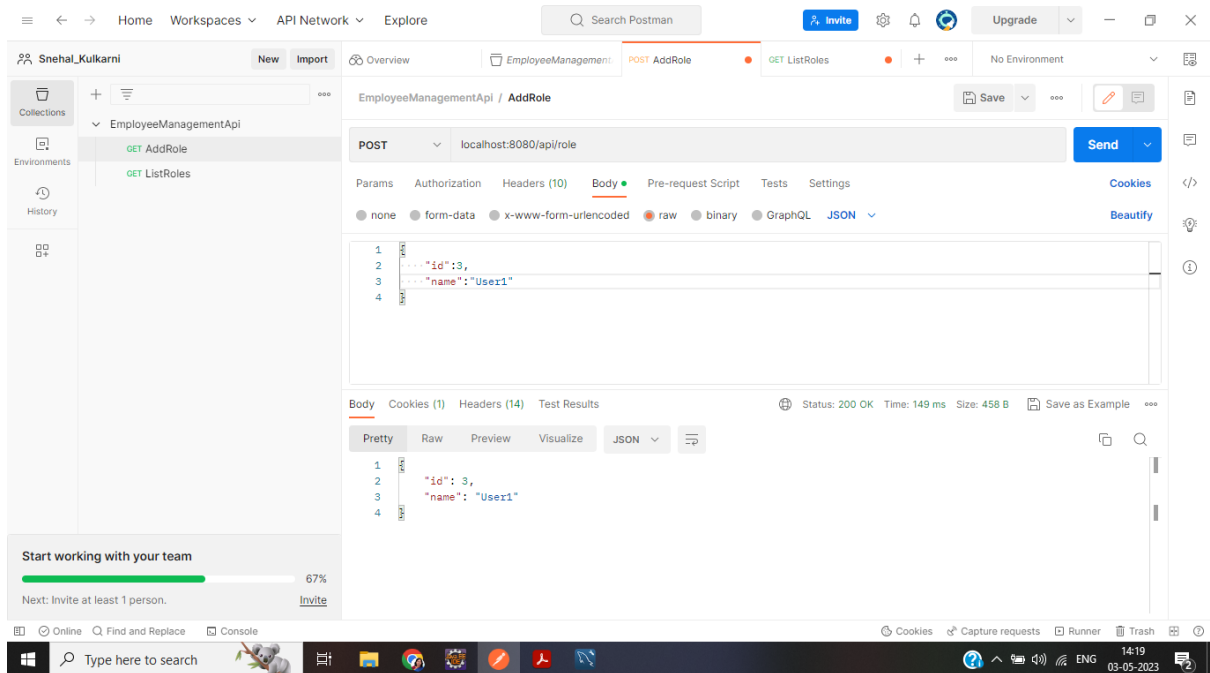


Add Role



List Roles

The screenshot shows the Postman application interface. The left sidebar displays the 'Collections' tab with a collection named 'EmployeeManagementApi' containing two requests: 'GET AddRole' and 'GET ListRoles'. The 'GET ListRoles' request is selected. The main panel shows the request details for 'GET localhost:8080/api/role/list'. The response is a JSON array of three roles, displayed in the 'Body' tab:

```
1 [{"id": 1,
2   "name": "Admin"},
3   {"id": 2,
4     "name": "User"},
5   {"id": 3,
6     "name": "User1"}]
```

The status bar at the bottom indicates the response status is 200 OK, with a time of 12 ms and a size of 507 B.

Add User As Admin

The screenshot shows the Postman application interface. The left sidebar displays the 'Collections' tab with a collection named 'EmployeeManagementApi' containing three requests: 'GET AddRole', 'GET ListRoles', and 'POST AddUser'. The 'POST AddUser' request is selected. The main panel shows the request details for 'POST localhost:8080/api/user'. The request body is a JSON object with the following fields:

```
1 {
2   "username": "SnehalK",
3   "password": "abc123",
4   "roles": [
5     {
6       "name": "Admin"
7     }
8   ]
9 }
```

The response is a JSON object with the following fields:

```
1 {
2   "id": 8,
3   "username": "SnehalK",
4   "password": "$2a$10$7Y4A8tELf13/wiShS8nehhwBBYxGBXsyp1C5uMBZCctgDM1k1w6",
5   "roles": [
6     {
7       "id": 1,
8       "name": "Admin"
9     }
10  ]
11 }
```

The status bar at the bottom indicates the response status is 200 OK, with a time of 168 ms and a size of 572 B.

Add User as User

The screenshot shows the Postman interface with a workspace named 'EmployeeManagementApi'. The 'AddUser' endpoint is selected, and a POST request is configured to 'localhost:8080/api/user'. The request body is a JSON object with the following structure:

```
1 {
2   "username": "YogeshK",
3   "password": "abc123",
4   "roles": [
5     {
6       "name": "User1"
7     }
8   ]
9 }
```

The response is displayed in the 'Body' tab, showing a successful status of 200 OK. The response body is a JSON object with the following structure:

```
1 {
2   "id": 9,
3   "username": "YogeshK",
4   "password": "$2a$10$wgm74bytyI.ISx5n6XX8fu3qqupIN4C3euUkcbp9rgLeseAisz8UK",
5   "roles": [
6     {
7       "id": 3,
8       "name": "User1"
9     }
10  ]
11 }
```

List Users

The screenshot shows the Postman interface with a workspace named 'EmployeeManagementApi'. The 'ListEmployees' endpoint is selected, and a GET request is configured to 'localhost:8080/api/employees/list'. The request body is a JSON object with the following structure:

```
1 {
2   "id": 1,
3   "firstName": "Kshitij",
4   "lastName": "Patake",
5   "email": "knp@spm.com"
6 },
7 {
8   "id": 2,
9   "firstName": "Aarush",
10  "lastName": "Deshpande",
11  "email": "aad@spm.com"
12 },
13 {
14   "id": 3,
15   "firstName": "Vedang",
16   "lastName": "Kulkarni",
17   "email": "vsk@spm.com"
18 }
```

The response is displayed in the 'Body' tab, showing a successful status of 200 OK. The response body is a JSON object with the following structure:

```
1 {
2   "id": 1,
3   "firstName": "Kshitij",
4   "lastName": "Patake",
5   "email": "knp@spm.com"
6 },
7 {
8   "id": 2,
9   "firstName": "Aarush",
10  "lastName": "Deshpande",
11  "email": "aad@spm.com"
12 },
13 {
14   "id": 3,
15   "firstName": "Vedang",
16   "lastName": "Kulkarni",
17   "email": "vsk@spm.com"
18 }
```

Add Employee (No Auth)

The screenshot shows the Postman application interface. On the left, the 'Collections' pane shows a collection named 'EmployeeManagementApi' with several endpoints, including 'createEmployee'. The main workspace displays a POST request to 'localhost:8080/api/employees'. The request body is a JSON object with the following fields: 'firstName': 'Kshitij', 'lastName': 'Patake', and 'email': 'knp@spm.com'. The response pane shows a status of '401 Unauthorized' with a time of 39 ms and a size of 443 B. The bottom status bar indicates the system is online and provides search and console options.

```
POST localhost:8080/api/employees
```

```
{  "firstName": "Kshitij",  "lastName": "Patake",  "email": "knp@spm.com"}
```

Status: 401 Unauthorized Time: 39 ms Size: 443 B

Add Employee (With User Role)

The screenshot shows the Postman application interface. On the left, the 'Collections' pane shows a collection named 'EmployeeManagementApi' with several endpoints, including 'createEmployee'. The main workspace displays a POST request to 'localhost:8080/api/employees'. The request body is a JSON object with the following fields: 'firstName': 'Vedang', 'lastName': 'Kulkarni', and 'email': 'vsk@spm.com'. The response pane shows a status of '403 Forbidden' with a time of 164 ms and a size of 566 B. The response body is a JSON object with the following fields: 'timestamp': '2023-05-03T09:12:38.146+00:00', 'status': 403, 'error': 'Forbidden', 'message': 'Forbidden', and 'path': '/api/employees'. The bottom status bar indicates the system is online and provides search and console options.

```
POST localhost:8080/api/employees
```

```
{  "firstName": "Vedang",  "lastName": "Kulkarni",  "email": "vsk@spm.com"}
```

Status: 403 Forbidden Time: 164 ms Size: 566 B

```
{  "timestamp": "2023-05-03T09:12:38.146+00:00",  "status": 403,  "error": "Forbidden",  "message": "Forbidden",  "path": "/api/employees"}
```

Add Employee (With Admin Role)

The screenshot shows the Postman interface with a workspace named 'EmployeeManagementApi'. A new POST request is created with the URL 'localhost:8080/api/employees'. The request body is a JSON object representing an employee:

```
{  "id": 2,  "firstName": "Aarush",  "lastName": "Deshpande",  "email": "aad@spm.com"}
```

. The response status is 200 OK, and the response body is a JSON object:

```
{  "id": 2,  "firstName": "Aarush",  "lastName": "Deshpande",  "email": "aad@spm.com"}
```

. The interface also shows a sidebar with collections and environments, and a bottom status bar with system information.

Database

The screenshot shows the MySQL Workbench interface. A query is executed:

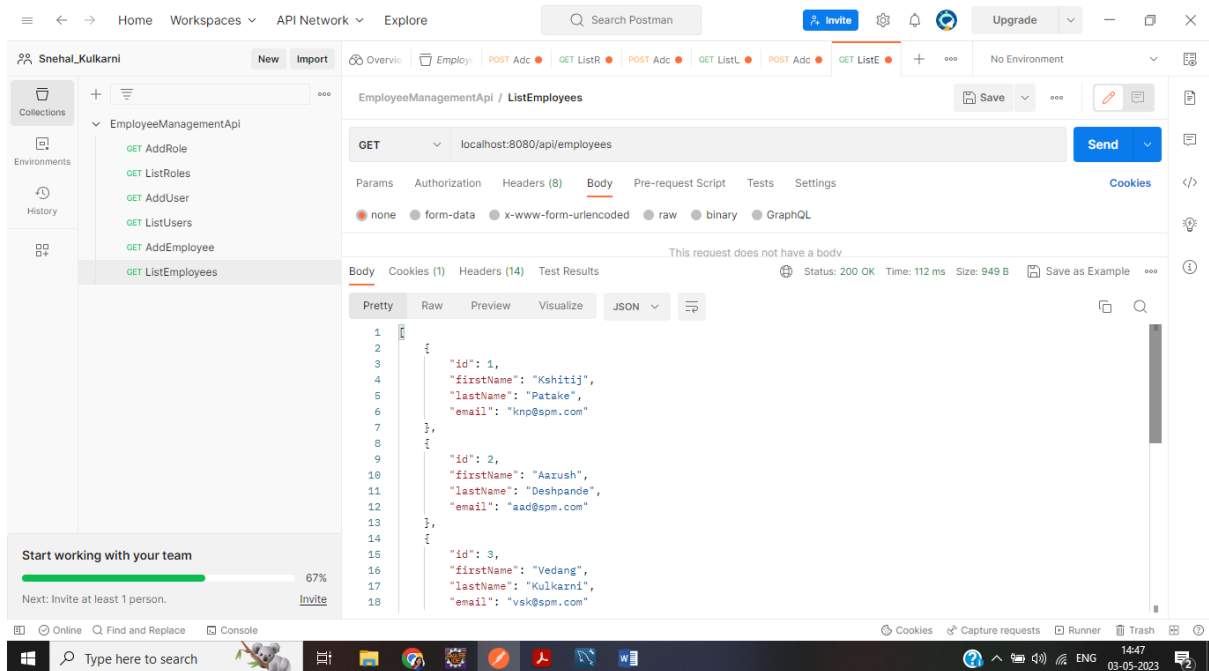
```
SELECT * FROM emp_db.employees;
```

. The result set shows 7 rows of employee data:

id	email	first_name	last_name
1	knp@spm.com	Kshiti	Patake
2	aad@spm.com	Aarush	Deshpande
3	vsk@spm.com	Vedant	Kulkarni
4	cyk@spm.com	Chintan	Kulkarni
5	pnj@spm.com	Prathmesh	Joshi
6	sq@spm.com	Soham	Gavade
7	aag@spm.com	Anshul	Gandhi

The interface also shows a sidebar with a schema tree, a query editor, and an output window showing the execution details of the query.

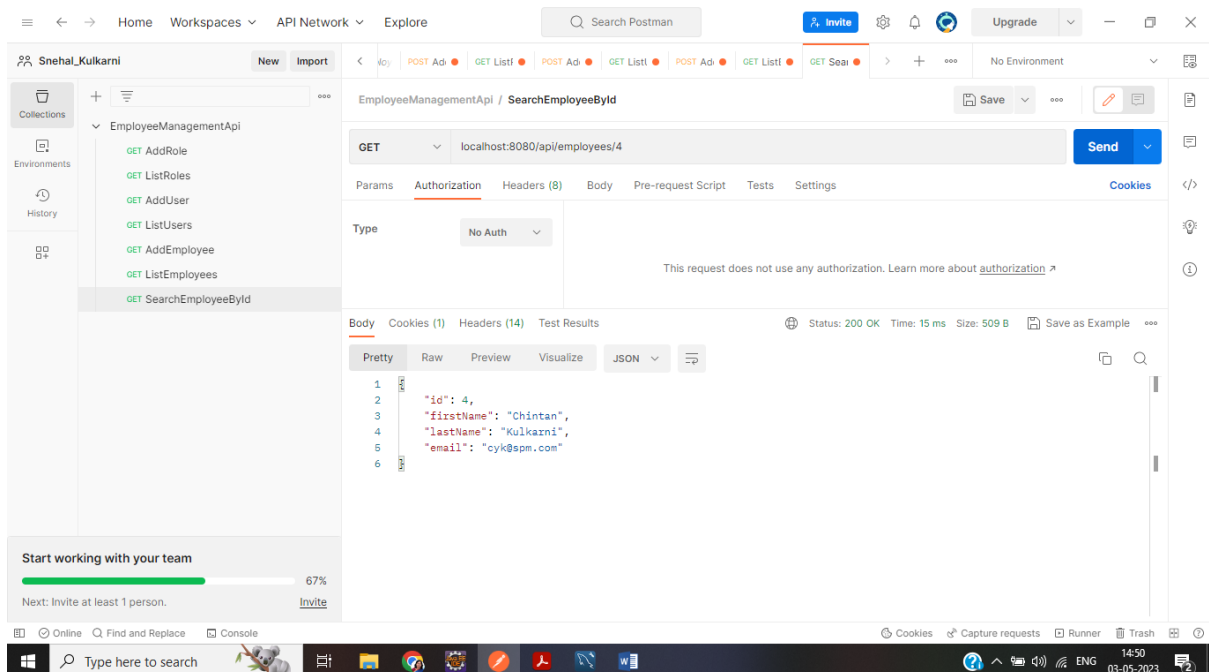
List of Employees



The screenshot shows the Postman application interface. On the left, the 'Collections' panel is expanded, showing a collection named 'EmployeeManagementApi' with several endpoints. The 'ListEmployees' endpoint is selected. The main panel displays the details of a GET request to 'localhost:8080/api/employees'. The request is successful, returning a 200 OK status. The response body is a JSON array of three employee objects, displayed in the 'Body' tab. The response is also shown in the 'Raw' tab as a JSON string.

```
1 {
2   "id": 1,
3   "firstName": "Kshitij",
4   "lastName": "Patake",
5   "email": "knp@spm.com"
6 },
7 {
8   "id": 2,
9   "firstName": "Aarush",
10  "lastName": "Deshpande",
11  "email": "aad@spm.com"
12 },
13 {
14   "id": 3,
15   "firstName": "Vedang",
16   "lastName": "Kulkarni",
17   "email": "vsk@spm.com"
18 }
```

Search an employee by ID



The screenshot shows the Postman application interface. On the left, the 'Collections' panel is expanded, showing a collection named 'EmployeeManagementApi' with several endpoints. The 'SearchEmployeeById' endpoint is selected. The main panel displays the details of a GET request to 'localhost:8080/api/employees/4'. The request is successful, returning a 200 OK status. The response body is a JSON object representing an employee with ID 4, displayed in the 'Body' tab. The response is also shown in the 'Raw' tab as a JSON string.

```
1 {
2   "id": 4,
3   "firstName": "Chintan",
4   "lastName": "Kulkarni",
5   "email": "cyk@spm.com"
6 }
```

Update an employee

The screenshot shows the Postman interface with a PUT request to `localhost:8080/api/employees`. The request body is a JSON object representing an employee to be updated:

```
{
  "id": 4,
  "firstName": "Chintan",
  "lastName": "Kulkarni",
  "email": "csyk@spm.com"
}
```

The response status is 200 OK. The response body is the same JSON object as the request body.

Start working with your team
Next: Invite at least 1 person. [Invite](#)

The screenshot shows the MySQL Workbench interface. The SQL query executed is:

```
SELECT * FROM emp_db.employees;
```

The result grid shows the following data:

	id	email	first_name	last_name
1	knp@spm.com	Kshitij	Patake	
2	aad@spm.com	Aarush	Deshpande	
3	vsk@spm.com	Vedant	Kulkarni	
4	csyk@spm.com	Chintan	Kulkarni	
5	pnj@spm.com	Pranimesh	Joshi	
6	sq@spm.com	Soham	Gavade	
7	aag@spm.com	Anshul	Gandhi	
10	cysk@spm.com	ChintanY	Kulkarni	

The output section shows the following messages:

```
35 14:55:09 SELECT * FROM emp_db.employees LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 sec
36 14:57:46 SELECT * FROM emp_db.employees LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 sec
```

Delete an employee by Id

The screenshot shows the Postman API client interface. On the left, a sidebar lists collections and environments. The main workspace displays a DELETE request to `localhost:8080/api/employees/10`. The request body is empty. The response status is `200 OK` with a response time of `717 ms` and a size of `459 B`. The response body contains the message: `Deleted employee id - 10`.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query: `SELECT * FROM emp_db.employees;`. The query results are displayed in a table with columns: `id`, `email`, `first_name`, and `last_name`. The results show 7 rows of employee data.

id	email	first_name	last_name
1	knp@spm.com	Kshitij	Patake
2	aad@spm.com	Aarush	Deshpande
3	vsk@spm.com	Vedant	Kulkarni
4	cyyk@spm.com	Chintan	Kulkarni
5	pnj@spm.com	Prathmesh	Joshi
6	sq@spm.com	Soham	Gavade
7	aag@spm.com	Anshul	Gandhi

Search an employee by first name

The screenshot shows the Postman application interface. On the left, the 'Collections' panel is expanded to 'EmployeeManagementApi', and the 'GET SearchByFirstName' endpoint is selected. The main panel shows the request details: a GET request to 'localhost:8080/api/employees/search/Anshul'. The 'Body' tab is active, displaying a JSON response:

```
1 {
2   "id": 7,
3   "firstName": "Anshul",
4   "lastName": "Gandhi",
5   "email": "aag@spm.com"
6 }
```

The status bar at the bottom indicates 'Status: 200 OK', 'Time: 69 ms', and 'Size: 508 B'. The Windows taskbar at the bottom shows the date as 03-05-2023 and time as 15:03.

Search an employee by first name (employees having same name)

The screenshot shows the Postman application interface. On the left, the 'Collections' panel is expanded to 'EmployeeManagementApi', and the 'GET SearchByFirstName' endpoint is selected. The main panel shows the request details: a GET request to 'localhost:8080/api/employees/search/Chintan'. The 'Body' tab is active, displaying a JSON array response:

```
1 [
2   {
3     "id": 4,
4     "firstName": "Chintan",
5     "lastName": "Kulkarni",
6     "email": "csyk@spm.com"
7   },
8   {
9     "id": 11,
10    "firstName": "ChintanY",
11    "lastName": "Kulkarni",
12    "email": "csyk@spm.com"
13  }
14 ]
```

The status bar at the bottom indicates 'Status: 200 OK', 'Time: 13 ms', and 'Size: 590 B'. The Windows taskbar at the bottom shows the date as 03-05-2023 and time as 15:04.

Records in ascending order

The screenshot shows the Postman application interface. On the left, a sidebar lists collections, with 'EmployeeManagementApi' expanded. It contains several endpoints, with 'GET Record in Ascending Order' selected. The main panel displays the details of this request. The URL is 'localhost:8080/api/employees/sort?order=asc'. The request method is 'GET'. The response is shown in the 'Body' tab, displaying a JSON array of employee records sorted by ID in ascending order. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 383 ms', and 'Size: 1 KB'.

EmployeeManagementApi / Record in Ascending Order

GET localhost:8080/api/employees/sort?order=asc

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies (1) Headers (14) Test Results Status: 200 OK Time: 383 ms Size: 1 KB Save as Example

```
1
2
3 {
4   "id": 2,
5   "firstName": "Aarush",
6   "lastName": "Deshpande",
7   "email": "aad@spm.com"
8 }
9
10 {
11   "id": 7,
12   "firstName": "Anshul",
13   "lastName": "Gandhi",
14   "email": "aag@spm.com"
15 }
16
17 {
18   "id": 4,
19   "firstName": "Chintan",
20   "lastName": "Kulkarni",
21   "email": "csyk@spm.com"
22 }
```

Records in descending order

The screenshot shows the Postman application interface. On the left, a sidebar lists collections, with 'EmployeeManagementApi' expanded. It contains several endpoints, with 'GET Records in Descending Order' selected. The main panel displays the details of this request. The URL is 'localhost:8080/api/employees/sort?order=desc'. The request method is 'GET'. The response is shown in the 'Body' tab, displaying a JSON array of employee records sorted by ID in descending order. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 167 ms', and 'Size: 1 KB'.

EmployeeManagementApi / Records in Descending Order

GET localhost:8080/api/employees/sort?order=desc

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description	Bulk Edit

Body Cookies (1) Headers (14) Test Results Status: 200 OK Time: 167 ms Size: 1 KB Save as Example

```
1
2 {
3   "id": 3,
4   "firstName": "Vedang",
5   "lastName": "Kulkarni",
6   "email": "vsk@spm.com"
7 }
8
9 {
10   "id": 6,
11   "firstName": "Soham",
12   "lastName": "Gavade",
13   "email": "sg@spm.com"
14 }
15
16 {
17   "id": 5,
18   "firstName": "Prathmesh",
19   "lastName": "Joshi",
20   "email": "pnj@spm.com"
21 }
```

Update role

EmployeeManagementApi / Update role

PUT localhost:8080/api/role/update

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 9,
3   "name": "Admin"
4 }
```

Body Cookies (1) Headers (14) Test Results

Status: 200 OK Time: 100 ms Size: 458 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 9,
3   "name": "Admin"
4 }
```

Start working with your team

Next: Invite at least 1 person. [Invite](#)

Update a User

EmployeeManagementApi / Update User

PUT localhost:8080/api/user/update

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 6,
3   "username": "Yogesh K",
4   "password": "1234",
5   "roles": [
6     {
7       "id": 8,
8       "name": "User"
9     }
10  ]
11 }
```

Body Cookies (1) Headers (15) Test Results

Status: 200 OK Time: 1542 ms Size: 647 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 6,
3   "username": "Yogesh K",
4   "password": "$2a$10$gaf0oXsRq/QzHCYvFDXEY.3/F6W3GQY8aBhHK7jTVI8gLOGF/DVEC",
5   "roles": [
6     {
7       "id": 8,
8       "name": "User"
9     }
10  ]
11 }
```

Start working with your team

Next: Invite at least 1 person. [Invite](#)

Delete user by id

The screenshot shows the Postman interface with a collection named 'EmployeeManagementApi'. The selected endpoint is 'Delete a user by id' with a DELETE method and URL 'localhost:8080/api/user/delete/11'. The request body is empty. The response status is 200 OK, and the response body is 'Deleted User id - 11'.

Start working with your team

Next: Invite at least 1 person. [Invite](#)

The screenshot shows the MySQL Workbench interface. The 'users' table is selected in the Navigator. The SQL query is 'SELECT * FROM emp_db.users;'. The result grid shows 8 rows of user data.

user_id	password	username
1	\$2a\$12\$W8paFzmL3J3H.M7byH9N1...	Admin
2	\$2a\$10\$BTAOKi8VhV23FB85Xkds.Oo...	Chintan
6	\$2a\$10\$..YU8xjKm3.MHTEVEeUmO...	Yogesh
7	\$2a\$10\$Wd0R0XlktxrmtVAbXkL8e...	Snehal
8	\$2a\$10\$7Y4A8tELf13/wSmhSBneh...	SnehalK
...

Output

#	Time	Action	Message	Duration / Fetch
53	18:58:59	SELECT * FROM emp_db.roles LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
54	18:59:18	SELECT * FROM emp_db.users LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Delete role by id

The screenshot shows the Postman application interface. On the left, the 'Collections' panel is expanded, showing a collection named 'EmployeeManagementApi'. The 'Delete role by id' endpoint is selected. The main panel displays the details of the DELETE request to 'localhost:8080/api/role/delete/4'. The 'Authorization' tab is active, showing 'Basic Auth' with 'Username: Admin' and 'Password: 12345'. The 'Body' tab shows the response: '1 Deleted Role id - 4'. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 87 ms', and 'Size: 454 B'.

EmployeeManagementApi / Delete role by id

DELETE localhost:8080/api/role/delete/4

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Type Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about

Username Admin

Password 12345

Body Cookies (1) Headers (14) Test Results

1 Deleted Role id - 4

Status: 200 OK Time: 87 ms Size: 454 B Save as Example

The screenshot shows the MySQL Workbench application interface. The 'Navigator' panel on the left shows the 'roles' table selected. The main panel displays the SQL query 'SELECT * FROM emp_db.roles;'. The 'Result Grid' shows the query results, which are 4 rows returned. The 'Output' panel at the bottom shows the execution details, including the time taken (18:58:29) and the message '4 row(s) returned'.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

college

college_db

e_commerce

emp_db

employees

roles

users

users_roles

Tables

Views

Stored Procedures

Functions

employee_db

Tables

employees

Views

Stored Procedures

Administration Schemas

Information

No object selected

roles 13 x

Output

Action Output

Time Action Message Duration / Fetch

52 18:58:29 SELECT * FROM emp_db.roles LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec

53 18:58:59 SELECT * FROM emp_db.roles LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec