

Create Admin User

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main workspace displays a POST request to `http://localhost:9200/api/v1/create-user`. The request body is a JSON object with the following fields: `firstName` (priya), `lastName` (kumar), `emailId` (priya123@gmail.com), `mobileNumber` (1234567890), and `password` (123). The response is a 200 OK status with a message: `"message": "User created successfully."`. The interface includes tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings. The Body tab is active, showing the JSON body. The response tab is also active, showing the response body. The bottom status bar indicates the response is 200 OK, 122 ms, and 204 B.

Explore Search Postman Invite Upgrade

Import Overview POST http://localhost:9200/ No Environment

http://localhost:9200/api/v1/create-user Save

POST http://localhost:9200/api/v1/create-user Send

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

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

```
1 {
2   "firstName": "priya",
3   "lastName": "kumar",
4   "emailId": "priya123@gmail.com",
5   "mobileNumber": "1234567890",
6   "password": "123"
7 }
8
```

Body Cookies Headers (5) Test Results 200 OK 122 ms 204 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "User created successfully."
3 }
```

Cookies Capture requests Bootcamp Runner Trash

Login successfully - Creation of Access Token

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a REST client request for `POST http://localhost:9200/api/v1/login`. The request body is in JSON format, containing `username` and `password` fields. The response is also in JSON format, showing a successful `200 OK` status and an `accessToken` value.

Request Details:

- Method: POST
- URL: `http://localhost:9200/api/v1/login`
- Body (JSON):

```
{  "username": "priya123@gmail.com",  "password": "123"}
```

Response Details:

- Status: 200 OK
- Time: 1075 ms
- Size: 226 B
- Body (JSON):

```
{  "accessToken": "Kd7svBD/66ts2QfvL0f0xW0375JkhU71Z/5JnN+VwTM="}
```

Get profile

The screenshot shows the Postman interface with a GET request configured. The URL is `http://localhost:9950/api/v1/get/profile`. The request is set to use an Authorization header. The response is a JSON object containing user profile data.

Request Configuration:

- Method: GET
- URL: `http://localhost:9950/api/v1/get/profile`
- Authorization: API Key
- Key: Authorization
- Value: `Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5...`

Response:

```
1 {
2   "id": 7,
3   "firstName": "priya",
4   "lastName": "kumar",
5   "emailId": "priya123@gmail.com",
6   "mobileNumber": "1234567890"
7 }
```

Update profile

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main workspace displays a POST request to `http://localhost:9951/api/v1/update-profile`. The request body is in JSON format, containing the following data:

```
{  "firstName": "priya",  "lastName": "dharsini",  "mobileNumber": "9999999999"}
```

The response is also in JSON format, showing a success message:

```
{  "message": "Profile updated successfully."}
```

The status bar at the bottom indicates a 200 OK response with a response time of 2.02 seconds and a size of 207 bytes.

Password changed

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a POST request to `http://localhost:9950/api/v1/change-password`. The request body is in JSON format, containing:

```
{  "currentPassword": "123",  "newPassword": "1234",  "retypePassword": "1234"}
```

The response status is `200 OK`. The response body is also in JSON format, containing:

```
{  "message": "Password has changed successfully, You need to login again."}
```

The bottom of the interface shows a status bar with icons for Cookies, Capture requests, Bootcamp, Runner, and Trash.

Login with invalid username or password

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a POST request to `http://localhost:9200/api/v1/login`. The request body is a JSON object: `{ "userName": "priya", "password": "123" }`. The response is a 400 Bad Request with a message: `"message": "Invalid username or password."`. The status bar at the bottom shows various icons for cookies, capturing requests, bootcamp, runner, trash, and help.

Explore

Search Postman

Invite

Upgrade

Overview

POST http://localhost:9200/api/v1/login

No Environment

Save

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

cookies

Beautiful

```
1 {
2   "userName": "priya",
3   "password": "123"
4 }
5
```

Body

cookies

Headers (4)

Test Results

400 Bad Request

114 ms

187 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "message": "Invalid username or password."
3 }
```

cookies

Capture requests

Bootcamp

Runner

Trash

?

Admin features

Quiz creation

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Explore', a search bar, and buttons for 'Invite', settings, notifications, and a profile icon. Below this, a toolbar shows 'Import', 'Overview', and tabs for different HTTP methods and URLs. The main workspace is divided into two sections: the top section for request configuration and the bottom section for response viewing.

Request Configuration:

- Method:** POST
- URL:** http://localhost:9951/api/v1/create/quiz
- Body Type:** raw (selected)
- Body Content:**

```
1 {  
2   "title": "Main test",  
3   "category": "Core java"  
4 }
```

Response Viewing:

- Status:** 200 OK
- Time:** 135 ms
- Size:** 204 B
- Body Type:** JSON (selected)
- Body Content:**

```
1 {  
2   "message": "Quiz created successfully."  
3 }
```

The interface also includes various utility buttons like 'Send', 'Save', 'Beautify', and 'Save Response'.

Quiz updation

The screenshot displays the Postman interface for a REST client. The top bar includes an 'Explore' button, a search bar labeled 'Search Postman', and several utility buttons: 'Invite', a settings gear, a notification bell, a circular refresh icon, and an 'Upgrade' button. Below the top bar, a tabbed interface shows the 'Overview' tab selected for the endpoint `http://localhost:9951/api/v1/update/quiz`. The request method is set to 'POST'. The 'Body' tab is active, showing a JSON payload:

```
{  "id": "8",  "title": "Main test",  "category": "Dot net"}
```

. The 'Headers' tab shows 9 headers. The 'Test Results' tab shows a successful response with status `200 OK`, a response time of `61 ms`, and a body size of `204 B`. The response body is displayed in the 'Body' tab, showing a JSON message:

```
{  "message": "Quiz updated successfully."}
```

. The bottom status bar includes icons for 'Cookies', 'Capture requests', 'Bootcamp', 'Runner', 'Trash', and a help icon.

Explore Search Postman Invite Upgrade

Overview POST http://loca GET http://localho POST http://localh POST http://loca

http://localhost:9951/api/v1/update/quiz Save

POST http://localhost:9951/api/v1/update/quiz Send

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

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

```
1 { 2   "id": "8", 3   "title": "Main test", 4   "category": "Dot net" 5 }
```

Body Cookies Headers (5) Test Results 200 OK 61 ms 204 B Save Response

Pretty Raw Preview Visualize JSON

```
1 { 2   "message": "Quiz updated successfully." 3 }
```

tor

Cookies Capture requests Bootcamp Runner Trash

Display quiz specifically by their id

The screenshot shows the Postman interface with a GET request to `http://localhost:9951/api/v1/get/quiz/8`. The response is a JSON object with the following data:

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Headers (5) Status Code: 200 OK

```
1 {  
2   "id": 8,  
3   "title": "Main test",  
4   "category": "Dot net"  
5 }
```

Display all quizzes

The screenshot shows the Postman interface with a GET request to `http://localhost:9951/api/v1/getAllQuiz` successfully executed. The response is a JSON array of quiz objects.

Request Details:

- Method: GET
- URL: `http://localhost:9951/api/v1/getAllQuiz`
- Authorization: API Key

Response Details:

- Status: 200 OK
- Time: 119 ms
- Size: 477 B

Response Body (JSON):

```
[{"id":1,"title":"Quiz 1","category":"SCIENCE"}, {"id":2,"title":"Q2","category":"MATH"}, {"id":4,"title":"Q4","category":"MATH"}, {"id":5,"title":"Q5","category":"SCIENCE"}, {"id":6,"title":"Q6","category":"MATH"}, {"id":8,"title":"Main test","category":"Dot net"}, {"id":9,"title":"Main test","category":"Core java"}]
```

Add question

The image shows the Postman application interface. At the top, there's a navigation bar with 'Explore', a search bar, an 'Invite' button, settings, notifications, a profile icon, and an 'Upgrade' button. Below this, a breadcrumb trail shows the path: `http://localhost:9951/api/v1/add/question`. The main area is divided into two sections. The top section is for editing the request, showing a **POST** method to `http://localhost:9951/api/v1/add/question`. The 'Body' tab is selected, and the 'raw' radio button is chosen. The body content is a JSON object:

```
{  "question": "How many weeks in a year?",  "optionA": "55",  "optionB": "51",  "optionC": "52",  "optionD": "53",  "rightOption": "C"}
```

. The bottom section shows the response, with a status code of **200 OK**. The 'Body' tab is selected, and the 'Pretty' view is chosen, displaying the response:

```
{  "message": "Question created successfully."}
```

Explore [Invite](#) [Upgrade](#)

port < `tp://` **POST** `htt` **POST** `http://` `eg. http://` **GET** `http://` `eg. http://` `eg. http://` > + ... No Environment

... / question / `http://localhost:9951/api/v1/add/question` / `http://localhost:9951/api/v1/add/question` Save

POST `http://localhost:9951/api/v1/add/question`

Params Headers **Body**

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#) [Beautify](#)

```
1  {
2    "question": "How many weeks in a year?",
3    "optionA": "55",
4    "optionB": "51",
5    "optionC": "52",
6    "optionD": "53",
7    "rightOption": "C"
8  }
```

Body Headers (5) Status Code 200 OK

Pretty Raw Preview [JSON](#)

```
1  {
2    "message": "Question created successfully."
3  }
```

Update Question

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main workspace is divided into two sections. The top section shows the request configuration for a POST request to `http://localhost:9951/api/v1/update/question`. The request body is in JSON format, containing an object with fields: `id` (12), `question` (What does DNA stands for?), `optionA` (Dynamic Not Available), `optionB` (Dynamic Network Adopter), `optionC` (Deoxyribonucleic acid), `optionD` (Do Not Accept), and `rightOption` (C). The bottom section shows the response, which is a JSON object with a `message` field: "Question updated successfully." The status bar at the bottom indicates a 200 OK response with a 124 ms latency and 208 B of data.

Explore [Invite](#) [Upgrade](#)

port Overview **POST** `http://localhost:9950,` **POST** `http://localhost:9951/` No Environment

http://localhost:9951/api/v1/update/question Save

POST `http://localhost:9951/api/v1/update/question Send`

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

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

```
1  {
2    "id" : "12",
3    "question" : "What does DNA stands for?",
4    "optionA" : "Dynamic Not Available",
5    "optionB" : "Dynamic Network Adopter",
6    "optionC" : "Deoxyribonucleic acid",
7    "optionD" : "Do Not Accept",
8    "rightOption" : "C"
9  }
```

Body Cookies Headers (5) Test Results 200 OK 124 ms 208 B [Save Response](#)

Pretty Raw Preview Visualize **JSON**

```
1  {
2    "message": "Question updated successfully."
3  }
```

Display question by their id

The screenshot shows the Postman interface with a GET request to `http://localhost:9951/api/v1/get/question/13`. The request is successful, returning a 200 OK status with a response time of 115 ms and a body size of 306 B. The response body is displayed in JSON format, showing a question with four options and a right option.

Request: GET `http://localhost:9951/api/v1/get/question/13`

Response: 200 OK, 115 ms, 306 B

```
{
  "id": 13,
  "question": "How many bones are in the human body?",
  "optionA": "200",
  "optionB": "157",
  "optionC": "209",
  "optionD": "206",
  "rightOption": "D"
}
```


Display all questions

The screenshot shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a REST client request to `http://localhost:9951/api/v1/getAllQuestion` using the GET method. The 'Authorization' tab is selected, showing an API Key. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)".

The 'Body' tab shows the response, which is a JSON array of 14 questions and options. The response status is 200 OK, with a response time of 70 ms and a size of 1.59 KB. The response is displayed in the 'Pretty' format.

```
[{"id":2,"question":"Ques2","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"C"}, {"id":3,"question":"Ques3","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"C"}, {"id":4,"question":"Ques4","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"D"}, {"id":5,"question":"Ques5","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"A"}, {"id":6,"question":"Ques6","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"B"}, {"id":7,"question":"Ques7","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"C"}, {"id":8,"question":"Ques8","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"D"}, {"id":9,"question":"Ques9","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"A"}, {"id":10,"question":"Ques10","optionA":"1.0","optionB":"2.0","optionC":"3.0","optionD":"4.0","rightOption":"B"}, {"id":12,"question":"What does DNA stands for?","optionA":"Dynamic Not Available","optionB":"Dynamic Network Adopter","optionC":"Deoxyribonucleic acid","optionD":"Do Not Accept","rightOption":"C"}, {"id":13,"question":"How many bones are in the human body?","optionA":"200","optionB":"157","optionC":"209","optionD":"206","rightOption":"D"}, {"id":14,"question":"How many weeks in a year?","optionA":"55","optionB":"51","optionC":"52","optionD":"53","rightOption":"C"}]
```

Delete question

The screenshot displays the Postman application interface. At the top, the 'plore' logo is on the left, and a search bar, 'Invite' button, settings, notifications, and a profile icon are on the right. Below the top bar, the 'Overview' tab is active, showing a list of requests. The selected request is a GET request to 'http://localhost:9951/api/v1/delete/question/11'. The 'Authorization' tab is selected, showing an 'API Key' type. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The 'Key' is 'Authorization', the 'Value' is 'Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5...', and it is added to the 'Header'. The 'Body' tab is selected, showing the response in 'Pretty' format: '1', '2', '3' on the left and 'message': 'Question deleted successfully.' on the right. The status bar at the bottom indicates a 200 OK response, 94 ms latency, and 208 B body size.

Search Postman

Invite

Upgrade

Overview

POST http://localhost:9950/

GET http://localhost:9951/a

admin-service / question / http://localhost:9951/api/v1/update/question

Save

GET http://localhost:9951/api/v1/delete/question/11

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Type

API Key

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)

Key

Value

Add to

Authorization

Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5...

Header

Body

Cookies

Headers (5)

Test Results

200 OK

94 ms

208 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1
2  "message": "Question deleted successfully."
3
```


Delete quiz question

The screenshot shows the Postman interface with a DELETE request to `http://localhost:9951/api/v1/delete/quiz/question/3`. The request is configured with an API Key for authorization. The response is a 200 OK status with a message: `"message": "QuizQuestion deleted successfully."`

Request Details:

- Method: `DELETE`
- URL: `http://localhost:9951/api/v1/delete/quiz/question/3`
- Authorization: API Key
- Key: `Authorization`
- Value: `Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5...`

Response Details:

- Status: `200 OK`
- Time: `88 ms`
- Size: `212 B`
- Message: `"message": "QuizQuestion deleted successfully."`

Question added to quiz

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a POST request to `http://localhost:9951/api/v1/add/quiz/question`. The request body is a JSON object: `{ "quizId": "6", "questionId": "10" }`. The response is a 200 OK status with a message: `"message": "Question added successfully."`.

Explore [Invite](#) [Upgrade](#)

Overview [POST http://localhost:9951/api/v1/add/quiz/question](#) [GET http://localhost:9951/api/v1/add/quiz/question](#) [GET http://localhost:9951/api/v1/add/quiz/question](#) [GET http://localhost:9951/api/v1/add/quiz/question](#) [POST http://localhost:9951/api/v1/add/quiz/question](#) [+](#) [...](#) No Environment

[http://localhost:9951/api/v1/add/quiz/question](#) [Save](#) [</>](#)

POST [http://localhost:9951/api/v1/add/quiz/question](#) [Send](#)

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

none form-data x-www-form-urlencoded raw binary GraphQL [JSON](#)

```
1 {
2   "quizId": "6",
3   "questionId": "10"
4 }
5
```

Body Cookies Headers (5) Test Results 200 OK 89 ms 206 B [Save Response](#)

[Pretty](#) [Raw](#) [Preview](#) [Visualize](#) [JSON](#)

```
1 {
2   "message": "Question added successfully."
3 }
```

Display question by quiz id

The screenshot shows the Postman interface with a GET request to `http://localhost:9951/api/v1/get/quiz/question/6`. The request is configured with an API Key authorization header. The response is displayed in the bottom panel, showing a 200 OK status and a JSON array of quiz questions.

Request Details:

- Method: GET
- URL: `http://localhost:9951/api/v1/get/quiz/question/6`
- Authorization: API Key
- Key: Authorization
- Value: `Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5...`
- Add to: Header

Response Details:

- Status: 200 OK
- Time: 70 ms
- Size: 376 B
- Save Response

Response Body (JSON):

```
[{"id":1,"quizId":6,"questionId":5}, {"id":2,"quizId":6,"questionId":7}, {"id":3,"quizId":6,"questionId":2}, {"id":4,"quizId":6,"questionId":3}, {"id":8,"quizId":6,"questionId":10}, {"id":9,"quizId":6,"questionId":9}]
```

Leader Board

The screenshot shows the Postman interface with a GET request to `http://localhost:9952/api/v1/leaderBoard/8`. The request is configured with an API Key authorization. The response is a 200 OK status with a response time of 53 ms and a body size of 217 B. The response body is displayed in JSON format, showing a user with the first name "Anand", last name "Tiwari", and a score of 1.

Request Details:

- Method: GET
- URL: `http://localhost:9952/api/v1/leaderBoard/8`
- Authorization: API Key
- Key: Authorization
- Value: `Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5 ...`

Response Details:

- Status: 200 OK
- Time: 53 ms
- Size: 217 B
- Save Response: [Save Response](#)

Response Body (JSON):

```
1 {
2   {
3     "firstName": "Anand",
4     "lastName": "Tiwari",
5     "score": 1
6   }
7 }
```

Log out

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main workspace is divided into several sections:

- Request Section:** Shows a POST request to `http://localhost:9200/api/v1/logout`. The `Send` button is visible.
- Authorization Section:** The `Authorization` tab is selected. It shows an `API Key` type. The `Key` is `Authorization` and the `Value` is `Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5...`. The `Add to` dropdown is set to `Header`.
- Body Section:** The `Body` tab is selected. It shows the response in `JSON` format, which is `{"message": "Logout successfully"}`.
- Response Section:** The status is `200 OK`, the time is `201 ms`, and the size is `197 B`. The `Save Response` button is visible.

User features

Create user

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a POST request to `http://localhost:9950/api/v1/create-user`. The request body is a JSON object with user details. The response is a 200 OK status with a success message.

Request:

```
POST http://localhost:9950/api/v1/create-user
```

Body (JSON):

```
{  "firstName": "yoga",  "lastName": "priya",  "emailId": "yoga@gmail.com",  "mobileNumber": "141414141"}
```

Response:

```
{  "message": "User created successfully."}
```

The response status is 200 OK, with a response time of 88 ms and a size of 204 B.

User login

The image shows the Postman application interface. At the top, there's a search bar and navigation icons. The main area displays a POST request to `http://localhost:9950/api/v1/login`. The request body is in JSON format, containing `"userName": "yoga@gmail.com",` and `"password": "abc"`. The response is also in JSON format, showing a `200 OK` status, a response time of `380 ms`, and a body size of `206 B`. The response body contains `"accessToken": "flJfkJnmPIXX04JoVk1gcw=="`. The interface includes tabs for Params, Authorization, Headers, Body, Pre-request Script, Tests, and Settings. The Body tab is active, showing the request and response bodies. The response is displayed in the 'Body' tab at the bottom, with a 'Save Response' button.

Explore [Invite](#) [Upgrade](#)

port < tp:// POST htt POST http:// [e.g] http:// [e.g] http:// [e.g] http:// [e.g] http:// > + ... No Environment Save

... `http://localhost:9950/api/v1/login` Save

POST `http://localhost:9950/api/v1/login` [Send](#)

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings [Cookies](#)

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#) [Beautify](#)

```
1 {
2   "userName": "yoga@gmail.com",
3   "password": "abc"
4 }
```

Body Cookies Headers (5) Test Results 200 OK 380 ms 206 B [Save Response](#)

Pretty Raw Preview Visualize [JSON](#)

```
1 {
2   "accessToken": "flJfkJnmPIXX04JoVk1gcw=="
3 }
```


Explore various quizzes

The screenshot displays the Postman API client interface. At the top, there's a search bar and navigation icons. The main area shows a GET request to `http://localhost:9951/api/v1/getAllQuiz`. The request is configured with an API Key for authorization. The response is a 200 OK status, received in 119 ms with a body size of 477 B. The response body is displayed in the 'Body' tab, showing a JSON array of quiz data.

Request Details:

- Method: GET
- URL: `http://localhost:9951/api/v1/getAllQuiz`
- Authorization: API Key

Response Details:

- Status: 200 OK
- Time: 119 ms
- Size: 477 B

Response Body (JSON):

```
[{"id":1,"title":"Quiz 1","category":"SCIENCE"}, {"id":2,"title":"Q2","category":"MATH"}, {"id":4,"title":"Q4","category":"MATH"}, {"id":5,"title":"Q5","category":"SCIENCE"}, {"id":6,"title":"Q6","category":"MATH"}, {"id":8,"title":"Main test","category":"Dot net"}, {"id":9,"title":"Main test","category":"Core java"}]
```

Answer questions

The screenshot displays the Postman interface for a REST client. The top bar includes a search field, an 'Invite' button, and a 'No Environment' dropdown. The main workspace shows a POST request to `http://localhost:9952/api/v1/submit/question`. The 'Body' tab is selected, showing a JSON payload: `{ "quizId": "6", "questionId": "9", "answer": "A" }`. The 'Send' button is visible. Below the request, the 'Body' tab of the response is selected, showing a 200 OK status with a message: `"message": "Answer submitted successfully."`. The response is formatted as JSON and displayed in the 'Pretty' view.

Postman interface showing a successful POST request to `http://localhost:9952/api/v1/submit/question`. The request body is a JSON object:

```
{  "quizId": "6",  "questionId": "9",  "answer": "A"}
```

The response is a 200 OK status with a message: `"message": "Answer submitted successfully."`

Check their results

The screenshot displays the Postman interface for a REST client. The top navigation bar includes the 'Postman' logo, a search bar, and buttons for 'Invite', 'Upgrade', and user settings. The main workspace shows a GET request to the URL `http://localhost:9952/api/v1/leaderBoard/6`. The 'Authorization' tab is active, showing an 'API Key' type with a key value of `flJfkJnmPIXXO4JoVk1gcw== flJfkJnmPIXXO4JoVk1gcw==`. The 'Body' tab is selected, displaying a JSON response in 'Pretty' format:

```
1 {
2   "firstName": "yoga",
3   "lastName": "priya",
4   "score": 1
5 }
```

The response status is `200 OK` with a response time of `682 ms` and a body size of `215 B`. The 'Save Response' button is visible. The bottom right corner shows a search icon and a copy icon.

Get next question

The screenshot shows the Postman interface with a GET request to `http://localhost:9952/api/v1/get/quiz/next/question/6`. The request is successful, returning a 200 OK status with a response time of 217 ms and a body size of 274 B. The response body is displayed in the 'Pretty' format, showing a JSON object with the following structure:

```
1 {
2   "id": 5,
3   "question": "Ques5",
4   "optionA": "1.0",
5   "optionB": "2.0",
6   "optionC": "3.0",
7   "optionD": "4.0",
8   "rightOption": null
9 }
```

The interface also shows the 'Authorization' tab, which is currently empty. A warning message is visible: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)."

Tables - User table

The screenshot displays a database management interface. On the left, a 'SCHEMAS' pane shows a tree view of the database structure. The 'quiz' schema is expanded, revealing tables: 'login_session', 'question', 'quiz', 'quiz_question', 'user', and 'user_quiz_ques_ans'. The 'user' table is selected. The main window shows a SQL query: `SELECT * FROM quiz.user;`. Below the query, a 'Result Grid' displays the data from the 'user' table. The grid has columns: 'id', 'first_name', 'last_name', 'email_id', 'password', 'mobile_number', 'is_admin', and 'create_time'. The data is as follows:

	id	first_name	last_name	email_id	password	mobile_number	is_admin	create_time
▶	1	Jhon	Deo	admin@gmail.com	12345678	9900887766	Y	202
	2	qwe	qwr	ery	asd	io	N	202
	3	Anand	Tiwari	anand@gmail.com	123	124123	N	202
	4	Mahesh	Shittlani	mahesh@yopmail.com	123456	1122334455	N	202
	6	X	Y	XY@yopmail.com	123456	1122334455	N	202
	7	priya	priya	priya123@gmail.com	1234	9999999999	Y	202
	8	yoga	priya	yoga@gmail.com	abc	141414141	N	202
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

At the bottom of the interface, there is a tab labeled 'user 1' and buttons for 'Apply' and 'Revert'. The word 'Cont' is visible on the far right edge of the image.

Table - login_session

The screenshot shows a database management interface. On the left, a 'Navigator' pane displays a tree of 'SCHEMAS' including 'learneracademy', 'product', 'productmanagement', and 'quiz'. The 'quiz' schema is expanded, showing tables like 'login_session', 'question', 'quiz', 'quiz_question', 'user', and 'user_quiz_qes_ans'. The main window displays the 'login_session' table. The toolbar at the top includes icons for file operations, a search icon, and a 'Limit to 1000 rows' dropdown. The SQL editor shows the query: `SELECT * FROM quiz.login_session;`. Below the editor, the 'Result Grid' shows the table's data. The table has four columns: 'id', 'user_id', 'access_token', and 'last_access'. The data rows show sessions for user_id 7 and 8, with various access tokens and timestamps. A row with all NULL values is also present. On the right side, there are buttons for 'Result Grid', 'Form Editor', 'Apply', and 'Revert'.

	id	user_id	access_token	last_access
▶	7	7	Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5JnN...	2022-08-31 00:53:50
	8	7	Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5JnN...	2022-08-31 17:26:49
	9	7	Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5JnN...	2022-08-31 17:31:28
	10	7	Kd7svBD/66ts2QfvLOfOxWO375JkhU71Z/5JnN...	2022-08-31 17:36:17
	11	8	f1JfkJnmPIXO4JoVk1gcw==	2022-08-31 19:38:08
*	NULL	NULL	NULL	NULL

Table - quiz

The screenshot shows a database management interface. On the left, a 'Navigator' pane displays a tree of schemas. The 'quiz' schema is expanded, showing a list of tables: login_session, question, quiz, quiz_question, user, and user_quiz_ques_ans. The 'quiz' table is selected. The main area shows the SQL query: `SELECT * FROM quiz.quiz;`. Below the query, a 'Result Grid' displays the data for the 'quiz' table. The grid has 7 columns: id, title, category, deleted, created_on, and updated_on. The data is as follows:

	id	title	category	deleted	created_on	updated_on
	5	Q5	SCIENCE	N	2022-07-23 12:34:26	2022-07-23 12:34:26
	6	Q6	MATH	N	2022-07-23 12:45:04	2022-07-23 12:45:04
	7			Y	2022-07-24 06:51:27	2022-07-24 06:51:27
	8	Main test	Dot net	N	2022-07-24 07:30:14	2022-08-31 18:15:14
	9	Main test	Core java	N	2022-08-31 18:11:52	2022-08-31 18:11:52
•	NULL	NULL	NULL	NULL	NULL	NULL

Table - question

Navigator

SCHEMAS

Filter objects

- learneracademy
- product
- productmanagement
- quiz**
 - Tables
 - login_session
 - question
 - quiz
 - quiz_question
 - user
 - user_quiz_ques_ans
 - Views
 - Stored Procedures
 - Functions
- sakila
- sample1
- sample2
- sample3
- sample4

Administration Schemas

_ques_ans

user

quiz_question

user

login_session

quiz

Limit to 1000 rows

1 •

SELECT * FROM quiz.question;

Result Grid

Filter Rows:

Edit:

Export/Import:

	id	question	option_a	option_b	option_c	option_d	right_option	deleted	crea
	11	Ques11	1.0	2.0	3.0	4.0	C	Y	2022
	12	What d...	Dynami...	Dynami...	Deoxyri...	Do Not ...	C	N	2022
	13	How ma...	200	157	209	206	D	N	2022
	14	How ma...	55	51	52	53	C	N	2022
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

question 1 ×

Table - quiz_question

The screenshot displays a database management interface. On the left, a 'Navigator' pane shows a tree of schemas. The 'quiz' schema is expanded, revealing tables: login_session, question, quiz, quiz_question, user, and user_quiz_ques_ans. The main area shows the 'quiz_question' table selected. A toolbar at the top includes icons for file operations, search, and execution. Below the toolbar, a SQL query is entered: `SELECT * FROM quiz.quiz_question;`. The 'Result Grid' below the query shows the table's data. The grid has columns: id, quiz_id, question_id, deleted, created_on, and updated_on. It contains 7 rows of data. At the bottom, there are tabs for 'Administration' and 'Schemas', and an 'Apply' button.

Navigator

SCHEMAS

Filter objects

- learneracademy
- product
- productmanagement
- quiz**
 - Tables
 - login_session
 - question
 - quiz
 - quiz_question
 - user
 - user_quiz_ques_ans
 - Views
 - Stored Procedures
 - Functions
- sakila
- sample1
- sample2
- sample3
- sample4

Administration **Schemas**

quiz_question

Limit to 1000 rows

1 • `SELECT * FROM quiz.quiz_question;`

Result Grid Filter Rows: Edit: Export/Import:

	id	quiz_id	question_id	deleted	created_on	updated_on
▶	1	6	5	N	2022-07-23 16:16:12	2022-07-23 16:16:12
	2	6	7	N	2022-07-23 16:16:12	2022-07-23 16:16:12
	3	6	2	Y	2022-07-23 16:16:24	2022-08-31 18:41:51
	4	6	3	N	2022-07-23 16:16:24	2022-07-23 16:16:24
	5	1	10	N	2022-07-23 16:34:09	2022-07-23 16:34:09
	6	8	12	N	2022-07-24 07:49:40	2022-07-24 07:49:40
	7	8	13	N	2022-07-24 07:49:40	2022-07-24 07:49:40

quiz_question 1 x Apply

Table - user_quiz_ques_answer

The screenshot displays a database management interface. On the left, a sidebar shows a tree view of databases including 'learneracademy', 'product', 'productmanagement', and 'quiz'. The 'quiz' database is expanded, showing tables like 'login_session', 'question', 'quiz', 'quiz_question', 'user', and 'user_quiz_ques_ans'. The main area shows a SQL query: `SELECT * FROM quiz.user_quiz_ques_ans;`. Below the query, a 'Result Grid' displays the data for the 'user_quiz_ques_answer' table. The table has columns: 'id', 'user_id', 'quiz_question_id', 'selected_option', and 'created_on'. The data rows are:

	id	user_id	quiz_question_id	selected_option	created_on
▶	10	3	7	C	2022-07-24 15:03:50
	11	3	6	C	2022-07-24 15:03:54
	12	8	9	A	2022-08-31 19:42:11
•	NULL	NULL	NULL	NULL	NULL