



UJIAN BACKEND JC09 PURWADHIKA

Hamdan Nasrullah
hamdan.nasrullah@gmail.com

RUNNING API BACKEND

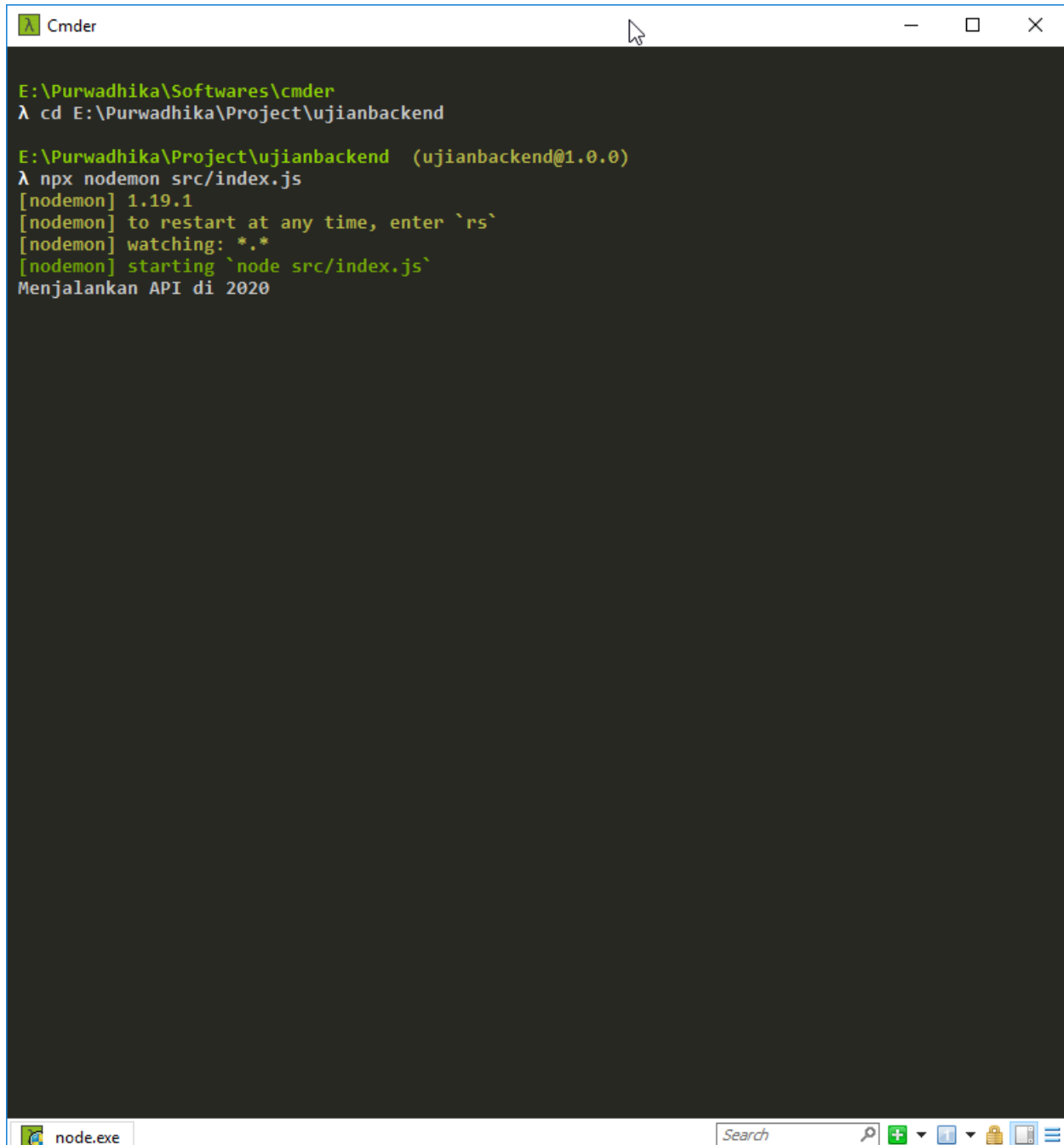
Instal node_modules:

```
npm install express mysql nodemon
```

Running API:

```
npx nodemon src/index.js
```

Port untuk test di Postman: 2020 (<http://localhost:2020/>)



```
E:\Purwadhika\Softwares\cmder
λ cd E:\Purwadhika\Project\ujianbackend

E:\Purwadhika\Project\ujianbackend (ujianbackend@1.0.0)
λ npx nodemon src/index.js
[nodemon] 1.19.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node src/index.js`
Menjalankan API di 2020
```

ADD MOVIES

The screenshot shows the Postman application interface. At the top, the 'POST' method is selected for the URL `http://localhost:2020/addmovie`. The request body is a JSON object:

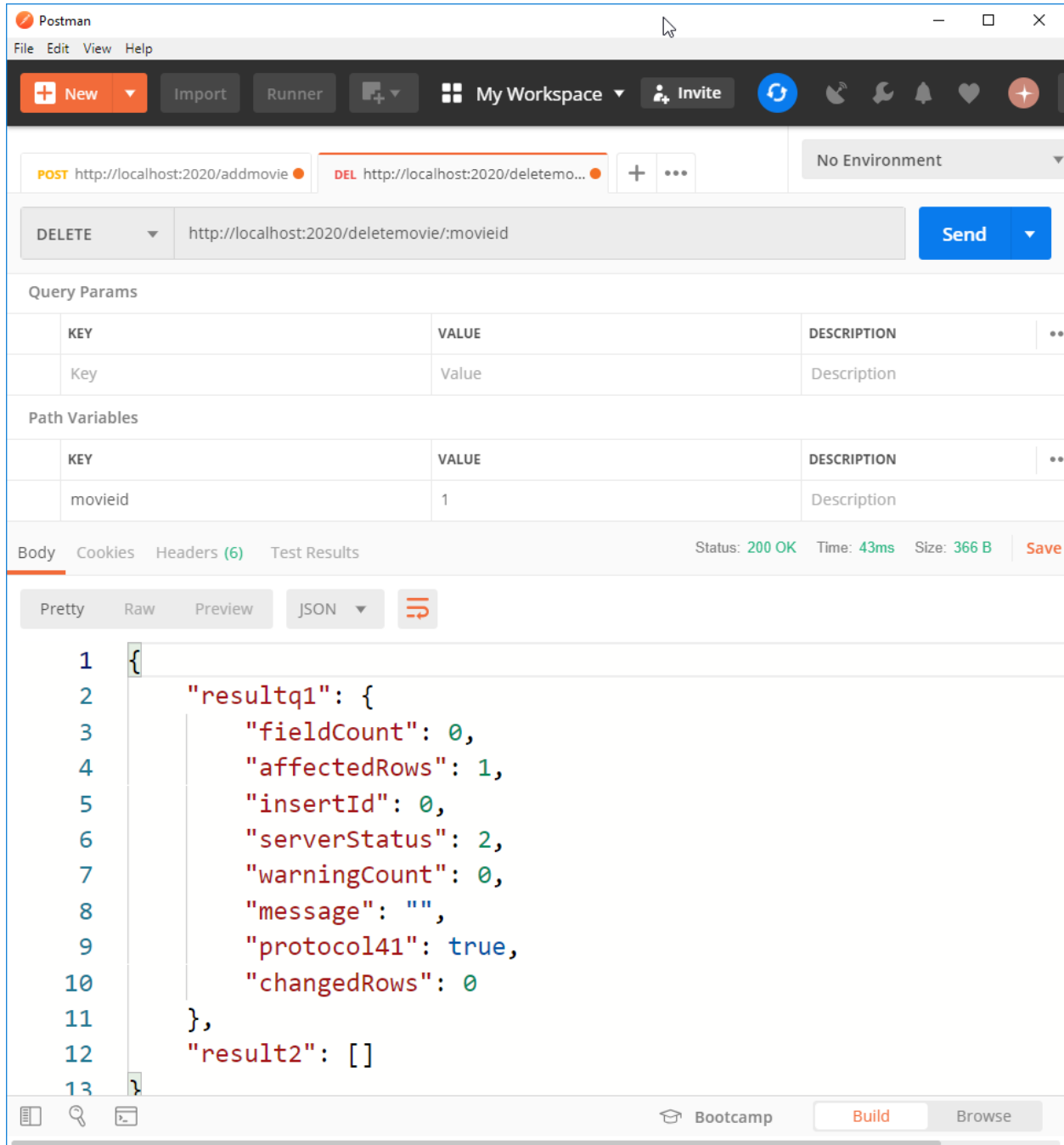
```
1 {  
2   "nama": "Bring The Soul: The Movie",  
3   "tahun": 2019,  
4   "deskripsi": "8/10"  
5 }
```

The response is displayed in the 'Body' tab, showing a JSON object with the same data plus an 'id' field:

```
1 [  
2   {  
3     "id": 4,  
4     "nama": "Bring The Soul: The Movie",  
5     "tahun": 2019,  
6     "deskripsi": "8/10"  
7   }  
8 ]
```

The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 63ms, Size: 289 B.

DELETE A MOVIE



Postman interface showing a DELETE request to `http://localhost:2020/deletemovie/:movieid`. The response is a JSON object with status 200 OK, time 43ms, and size 366 B.

Request:

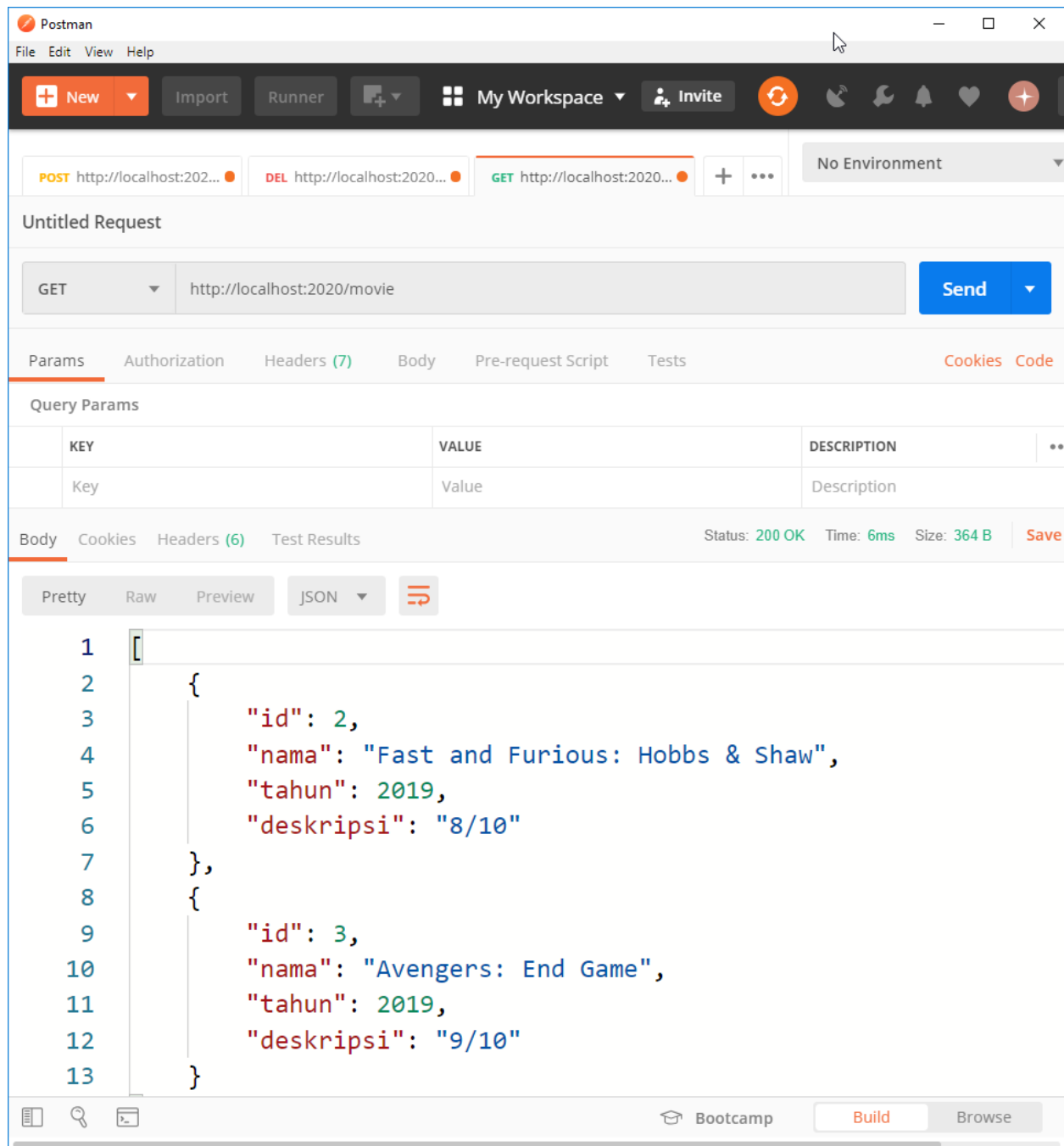
- Method: DELETE
- URL: `http://localhost:2020/deletemovie/:movieid`

Response:

```

1 {
2   "resultq1": {
3     "fieldCount": 0,
4     "affectedRows": 1,
5     "insertId": 0,
6     "serverStatus": 2,
7     "warningCount": 0,
8     "message": "",
9     "protocol41": true,
10    "changedRows": 0
11  },
12  "result2": []
13 }
  
```

SHOW MOVIES



The screenshot shows the Postman application interface. At the top, there's a menu bar (File, Edit, View, Help) and a toolbar with buttons for New, Import, Runner, My Workspace, and Invite. Below this, a request bar shows a GET request to `http://localhost:2020/movie` with a 'Send' button. The 'Params' tab is selected, showing a table for Query Params. The 'Body' tab is also visible, showing the response in JSON format. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 6ms', and 'Size: 364 B'.

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

```
[
  {
    "id": 2,
    "nama": "Fast and Furious: Hobbs & Shaw",
    "tahun": 2019,
    "deskripsi": "8/10"
  },
  {
    "id": 3,
    "nama": "Avengers: End Game",
    "tahun": 2019,
    "deskripsi": "9/10"
  }
]
```

EDIT A MOVIE

The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and a refresh button. The main area displays a 'PATCH' request to 'http://localhost:2020/editmovie/:movieid'. The 'Send' button is visible. Below the request bar, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', 'Cookies', and 'Code'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a JSON object with various fields including 'resultq1', 'fieldCount', 'affectedRows', 'insertId', 'serverStatus', 'warningCount', 'message', 'protocol41', and 'changedRows'. The status bar at the bottom shows 'Status: 200 OK', 'Time: 33ms', 'Size: 497 B', and a 'Save' button.

Untitled Request

PATCH http://localhost:2020/editmovie/:movieid

Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
movieid	2	Description

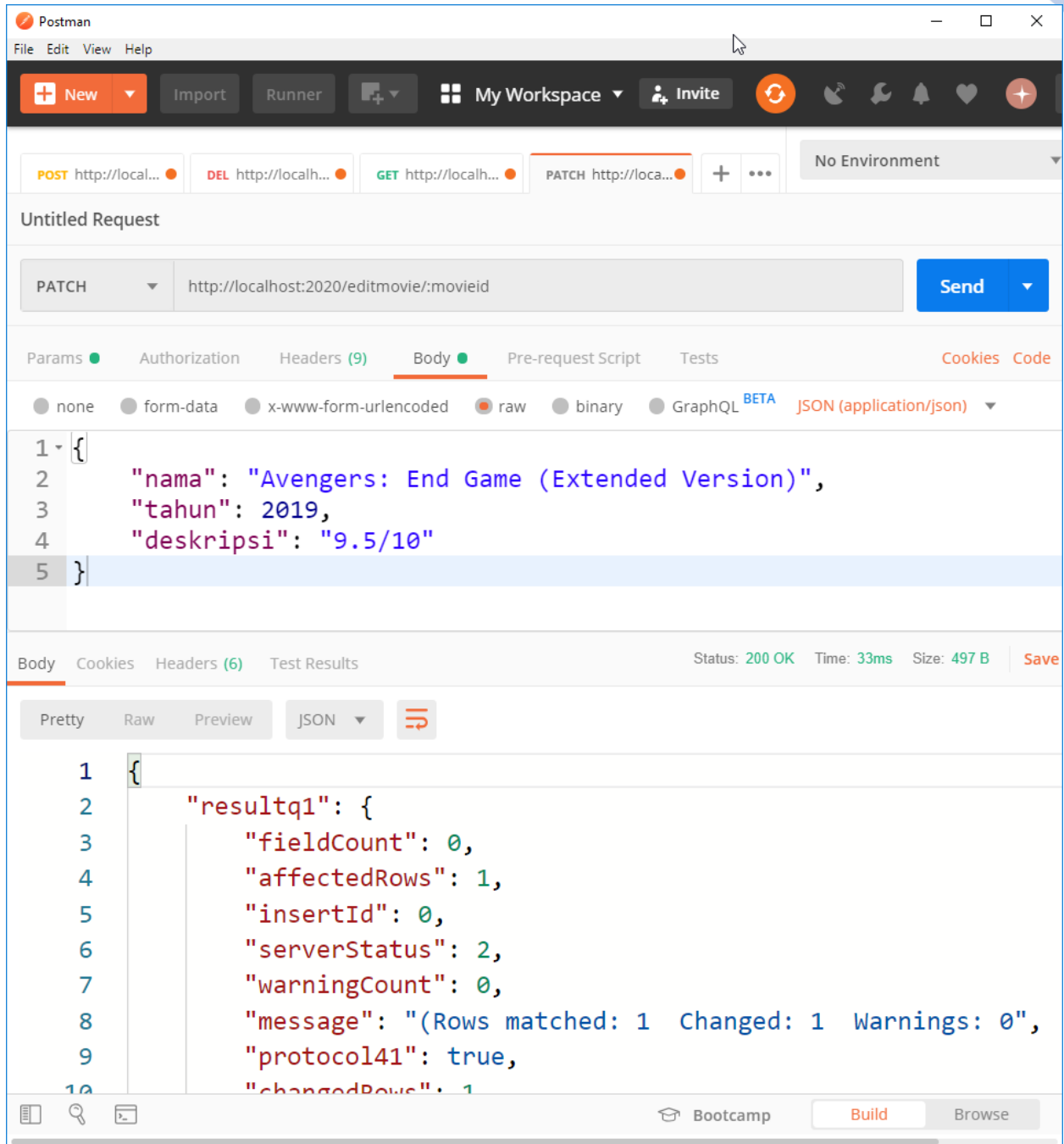
Body Cookies Headers (6) Test Results Status: 200 OK Time: 33ms Size: 497 B Save

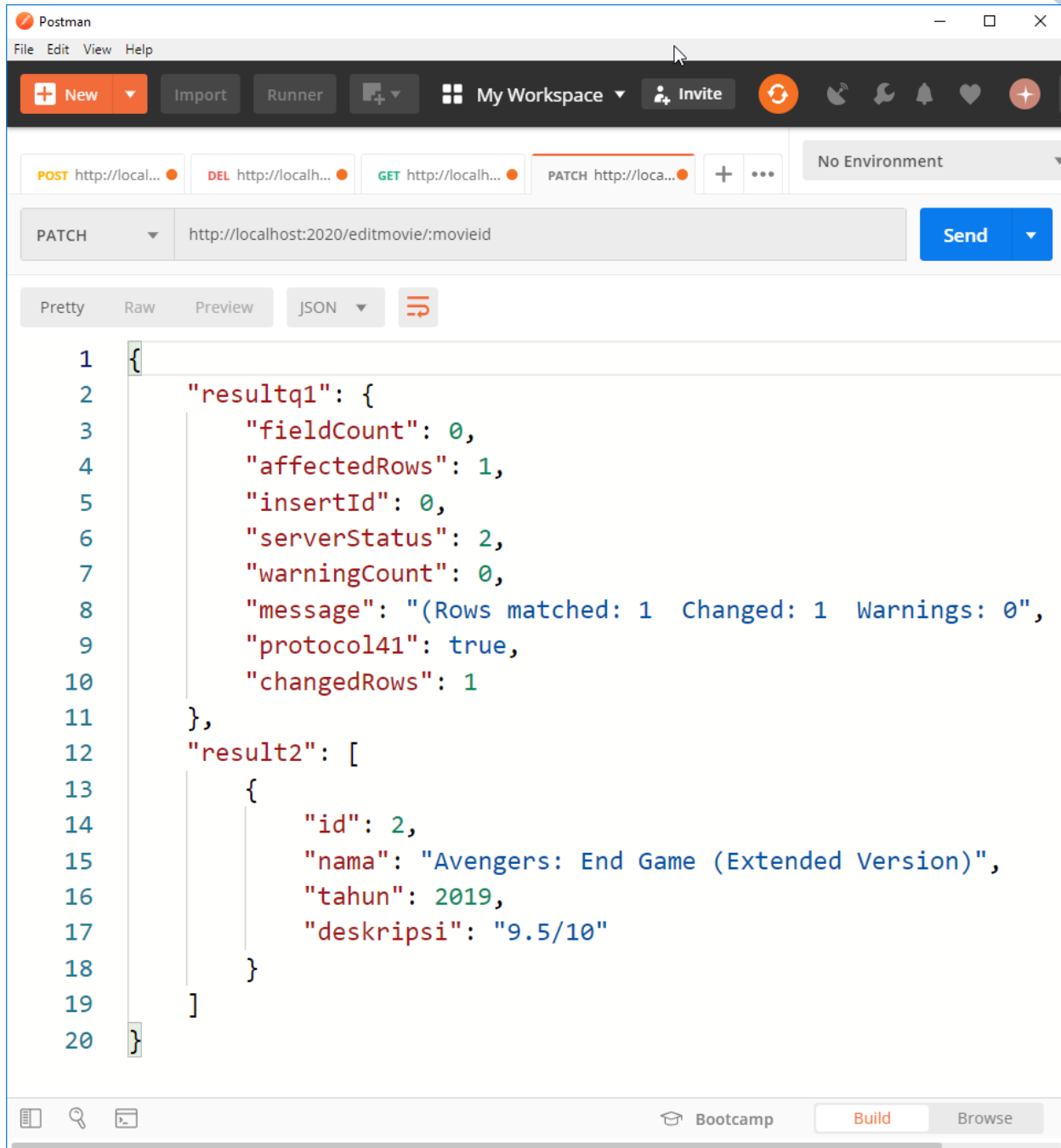
Pretty Raw Preview JSON

```

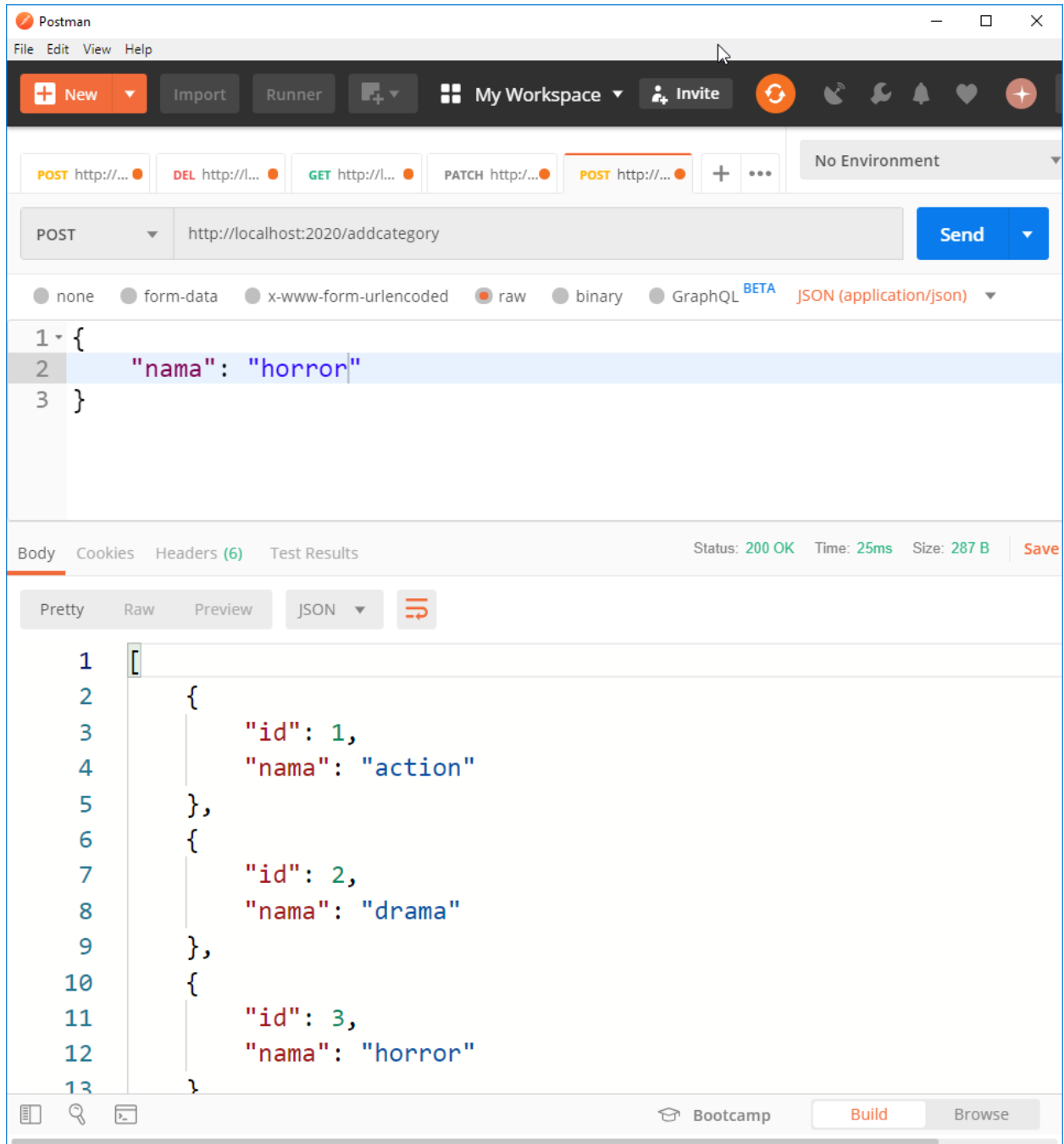
1 {
2   "resultq1": {
3     "fieldCount": 0,
4     "affectedRows": 1,
5     "insertId": 0,
6     "serverStatus": 2,
7     "warningCount": 0,
8     "message": "(Rows matched: 1 Changed: 1 Warnings: 0",
9     "protocol41": true,
10    "changedRows": 1
  
```

Bootcamp Build Browse





ADD CATEGORY



SHOW CATEGORY

The image shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and a refresh button. The main area is titled 'Untitled Request' and shows a 'GET' request to 'http://localhost:2020/category'. The 'Send' button is visible. Below the request bar, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', 'Cookies', and 'Code'. The 'Params' tab is active, showing a table for 'Query Params' with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table has one row with 'Key' and 'Value'. Below the table, there are tabs for 'Body', 'Cookies', 'Headers (6)', and 'Test Results'. The 'Body' tab is active, showing a JSON response in 'Pretty' format. The response is a list of two objects, each with 'id' and 'nama' fields. The status bar at the bottom shows 'Status: 200 OK', 'Time: 5ms', 'Size: 268 B', and a 'Save' button. The bottom of the window has a footer with 'Bootcamp', 'Build', and 'Browse' buttons.

Postman

File Edit View Help

New Import Runner My Workspace Invite

POST ... DEL ... GET ... PATCH ... POST ... PATCH ... DEL ... GET + ... No Environment

Untitled Request

GET http://localhost:2020/category Send

Params Authorization Headers (7) Body Pre-request Script Tests Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 5ms Size: 268 B Save

Pretty Raw Preview JSON

```
1 [
2   {
3     "id": 1,
4     "nama": "action scifi"
5   },
6   {
7     "id": 2,
8     "nama": "drama"
9   }
10 ]
```

Bootcamp Build Browse

EDIT CATEGORY

Postman

File Edit View Help

New Import Runner My Workspace Invite

POST htt... DEL htt... GET htt... PATCH ht... POST htt... PATCH ht... No Environment

Untitled Request

PATCH http://localhost:2020/editcategory/:categoryid Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
categoryid	1	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 62ms Size: 487 B Save

Pretty Raw Preview JSON

```

1 {
2   "resultq1": {
3     "fieldCount": 0,
4     "affectedRows": 1,
5     "insertId": 0,
6     "serverStatus": 2,
7     "warningCount": 0,
8     "message": "(Rows matched: 1 Changed: 1 Warnings: 0",
9     "protocol41": true,
10    "changedRows": 1
  
```

Bootcamp Build Browse

The screenshot displays the Postman interface for a PATCH request. The URL is `http://localhost:2020/editcategory/:categoryid`. The request body is a JSON object: `{ "nama": "action scifi" }`. The response status is 200 OK, with a time of 62ms and a size of 487 B. The response body is a JSON object: `{ "resultq1": { "fieldCount": 0, "affectedRows": 1, ... } }`.

Postman

File Edit View Help

New Import Runner My Workspace Invite

POST http... DEL http... GET http... PATCH ht... POST htt... PATCH ht... No Environment

Untitled Request

PATCH http://localhost:2020/editcategory/:categoryid Send

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

none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON (application/json)

```
1 {
2   "nama": "action scifi"
3 }
```

Status: 200 OK Time: 62ms Size: 487 B Save

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

```
1 {
2   "resultq1": {
3     "fieldCount": 0,
4     "affectedRows": 1,
```

Bootcamp Build Browse

Postman interface showing a successful PATCH request to `http://localhost:2020/editcategory/:categoryid`. The response is a JSON object with the following structure:

```
6      "serverStatus": 2,  
7      "warningCount": 0,  
8      "message": "(Rows matched: 1  Changed: 1  Warnings: 0",  
9      "protocol41": true,  
10     "changedRows": 1  
11   },  
12   "result2": [  
13     {  
14       "id": 1,  
15       "nama": "action scifi"  
16     },  
17     {  
18       "id": 2,  
19       "nama": "drama"  
20     },  
21     {  
22       "id": 3,  
23       "nama": "horror"  
24     }  
25   ]
```

The status is 200 OK, Time: 62ms, Size: 487 B. The interface also shows tabs for Body, Cookies, Headers (6), and Test Results. The bottom bar includes a search icon, a folder icon, and buttons for Bootcamp, Build, and Browse.

DELETE CATEGORY

The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:2020/deletecategory/:categoryid`. The request is configured with the following details:

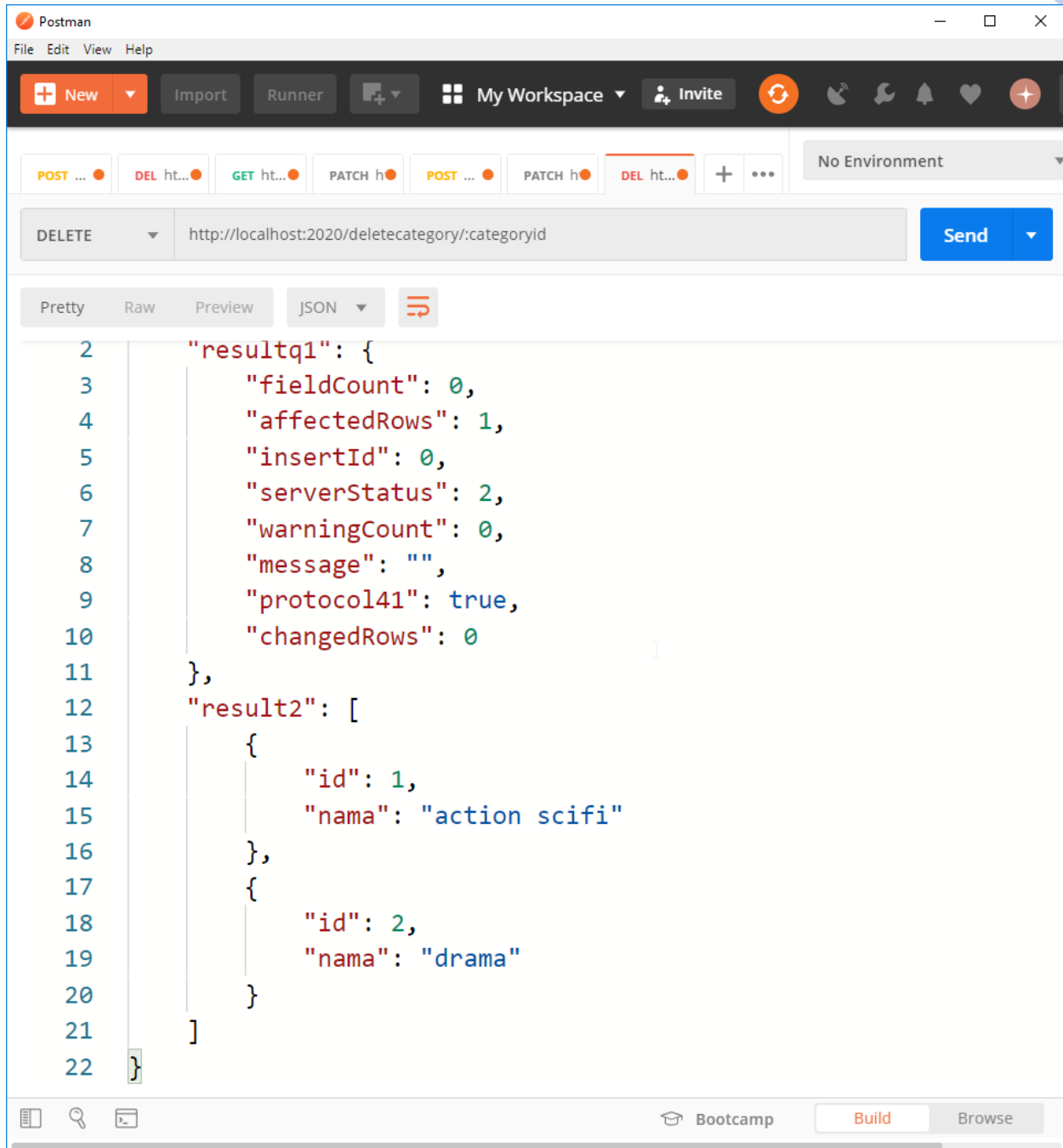
- Method:** DELETE
- URL:** `http://localhost:2020/deletecategory/:categoryid`
- Environment:** No Environment
- Params:**

KEY	VALUE	DESCRIPTION
Key	Value	Description
- Path Variables:**

KEY	VALUE	DESCRIPTION
categoryid	3	Description
- Response:** Status: 200 OK, Time: 30ms, Size: 420 B
- Body:**

```

1 {
2   "resultq1": {
3     "fieldCount": 0,
4     "affectedRows": 1,
5     "insertId": 0,
6     "serverStatus": 2,
7     "warningCount": 0,
8     "message": "",
9     "protocol41": true,
10    "changedRows": 0
  
```



The image shows the Postman application window. At the top, there's a menu bar (File, Edit, View, Help) and a toolbar with buttons for 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and a refresh button. Below the toolbar, there's a row of request type buttons: 'POST ...', 'DEL ht...', 'GET ht...', 'PATCH h...', 'POST ...', 'PATCH h...', and 'DEL ht...'. The 'DEL ht...' button is selected. The main area shows a 'DELETE' request to 'http://localhost:2020/deletecategory/:categoryid'. The response is displayed in the 'JSON' view, showing a JSON object with two main keys: 'resultq1' and 'result2'. 'resultq1' is an object with various status and count fields. 'result2' is an array of two objects, each with 'id' and 'nama' fields.

```
2      "resultq1": {
3          "fieldCount": 0,
4          "affectedRows": 1,
5          "insertId": 0,
6          "serverStatus": 2,
7          "warningCount": 0,
8          "message": "",
9          "protocol41": true,
10         "changedRows": 0
11     },
12     "result2": [
13         {
14             "id": 1,
15             "nama": "action scifi"
16         },
17         {
18             "id": 2,
19             "nama": "drama"
20         }
21     ]
22 }
```

ADD MOVIE CATEGORY

The screenshot shows the Postman application interface. At the top, the title bar says "Postman". Below it is a menu bar with "File", "Edit", "View", and "Help". A toolbar contains buttons for "New", "Import", "Runner", "My Workspace", "Invite", and a refresh icon. Below the toolbar is a row of request method buttons: "GET", "PATCH", "POST", "PATCH", "DEL", "GET", and "POST". The "POST" button is selected. To the right of these buttons is a dropdown menu showing "No Environment".

The main area is titled "Untitled Request". It shows a "POST" request to the URL "http://localhost:2020/movcat". A "Send" button is on the right. Below the URL bar are tabs for "Params", "Authorization", "Headers (9)", "Body", "Pre-request Script", "Tests", "Cookies", and "Code". The "Body" tab is selected. It shows a JSON body with the following content:

```
1 {  
2   "movie_id": 2,  
3   "category_id": 1  
4 }
```

Below the body tab are tabs for "Body", "Cookies", "Headers (6)", and "Test Results". The "Body" tab is selected. It shows a JSON response with the following content:

```
1 [  
2   {  
3     "id": 1,  
4     "movie_name": "Avengers: End Game (Extended Version)",  
5     "category_name": "action scifi"  
6   }  
7 ]
```

At the bottom of the interface, there is a status bar showing "Status: 200 OK", "Time: 25ms", "Size: 306 B", and a "Save" button. On the far right, there are buttons for "Build" and "Browse".

DELETE MOVIE CATEGORY

The image shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and a refresh button. The main area is divided into sections: 'Request' (method: DELETE, URL: http://localhost:2020/movcat/:movcatid), 'Query Params' (empty table), 'Path Variables' (table with key 'movcatid' and value '1'), 'Body' (selected tab, showing JSON response), 'Cookies', 'Headers (6)', and 'Test Results'. The JSON response is displayed in a pretty-printed format. At the bottom, there's a status bar showing 'Status: 200 OK', 'Time: 140ms', 'Size: 340 B', and a 'Save' button. The bottom right corner has a 'Bootcamp' logo and 'Build' and 'Browse' buttons.

DELETE http://localhost:2020/movcat/:movcatid

Send

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
movcatid	1	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 140ms Size: 340 B Save

Pretty Raw Preview JSON

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "serverStatus": 2,
6   "warningCount": 0,
7   "message": "",
8   "protocol41": true,
9   "changedRows": 0
10 }
```

Bootcamp Build Browse