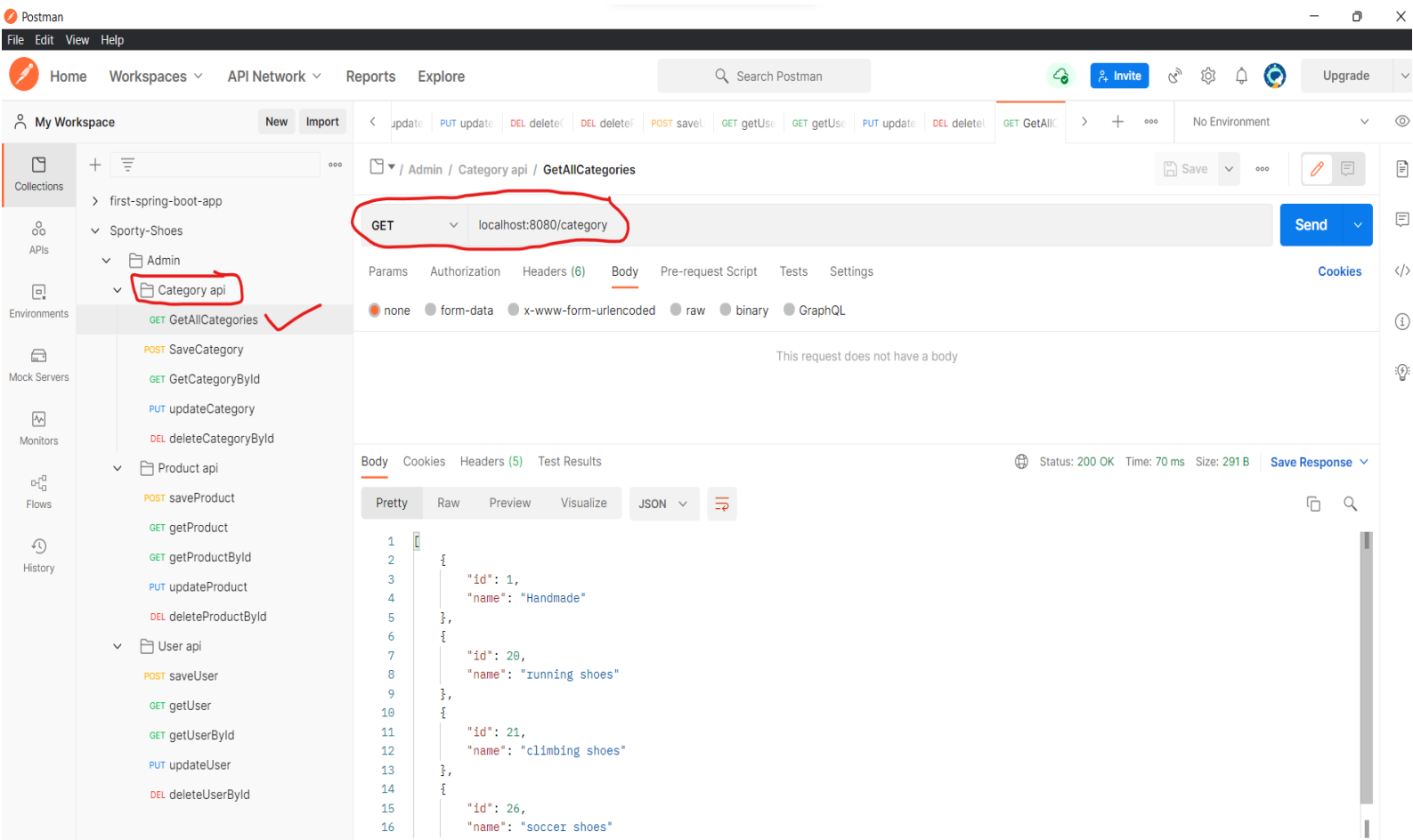


Sporty-Shoes

An E-commerce Website

*****Category API*****

1.getAllCategory

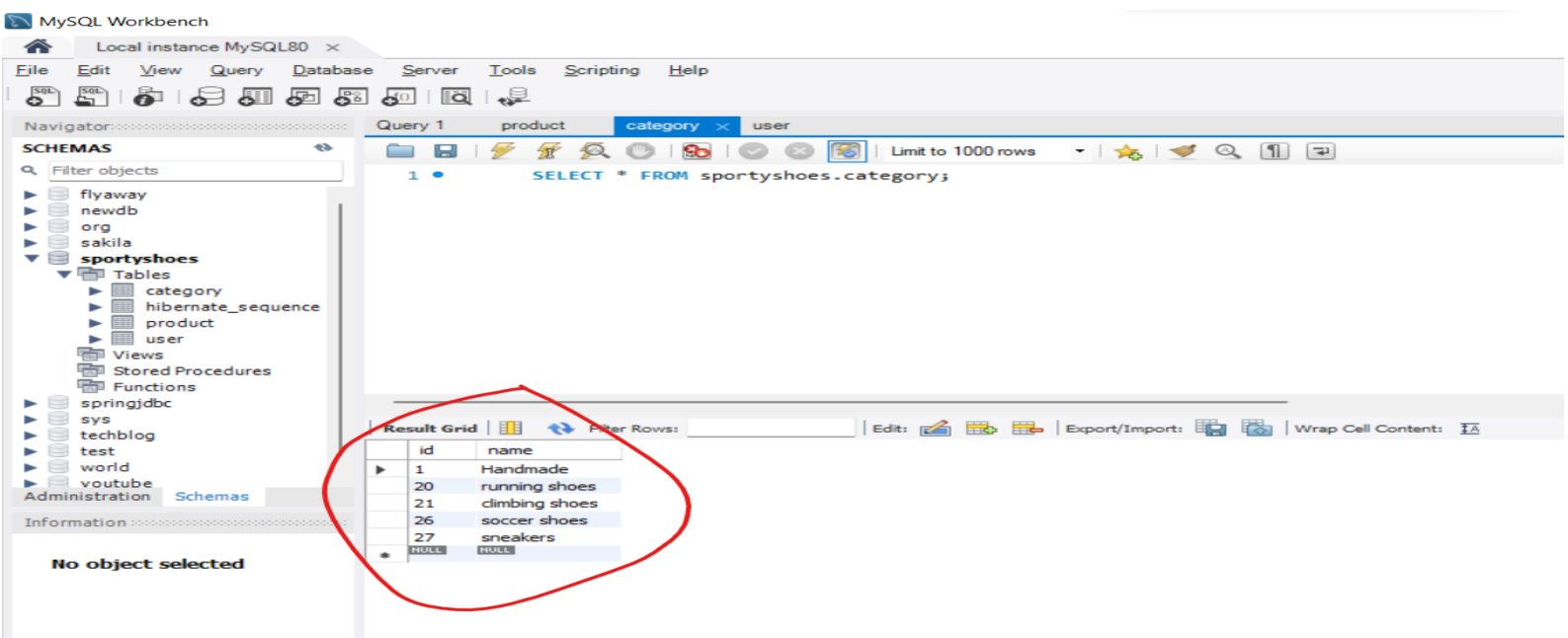


Postman interface showing a GET request to `localhost:8080/category`. The response is a JSON array of shoe categories.

```
GET localhost:8080/category
```

Response Body (JSON):

```
[{"id": 1, "name": "Handmade"}, {"id": 20, "name": "running shoes"}, {"id": 21, "name": "climbing shoes"}, {"id": 26, "name": "soccer shoes"}]
```



MySQL Workbench interface showing a query result for the `sportyshoes.category` table. The result grid displays the following data:

id	name
1	Handmade
20	running shoes
21	climbing shoes
26	soccer shoes
27	sneakers
NULL	NULL

2.saveCategory

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

Search Postman

My Workspace

New Import

Admin / Category api / SaveCategory

POST localhost:8080/category

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

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

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 28,
3   "name": "Leather"
4 }
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

flyaway newdb org sakila sportyshoes

Tables

category hibernate_sequence product user

Views Stored Procedures Functions

springjdbc sys techblog test world voutube

Administration Schemas

Information

No object selected

Query 1 product category user

Limit to 1000 rows

1 • SELECT * FROM sportyshoes.category;

Result Grid

	id	name
▶	1	Handmade
	20	running shoes
	21	climbing shoes
	26	soccer shoes
	27	sneakers
	28	Leather

3.getCategoryById

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

Search Postman

My Workspace

New Import

Collections

- first-spring-boot-app
 - Sporty-Shoes
 - Admin
 - Category api
 - GET GetAllCategories
 - POST SaveCategory
 - GET GetCategoryById ✓
 - PUT updateCategory
 - DEL deleteCategoryById
 - Product api
 - POST saveProduct
 - GET getProduct
 - GET getProductById
 - PUT updateProduct
 - DEL deleteProductById
 - User api
 - POST saveUser
 - GET getUser
 - GET getUserById
 - PUT updateUser
 - DEL deleteUserById

GET localhost:8080/category/28

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 28,
3   "name": "Leather"
4 }
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- flyaway
- newdb
- org
- sakila
- sportyshoes**
 - Tables
 - category
 - hibernate_sequence
 - product
 - user
 - Views
 - Stored Procedures
 - Functions
- springjdbc
- sys
- techblog
- test
- world
- voutube

Administration Schemas

Information

No object selected

Query 1 product category user

Limit to 1000 rows

1 • `SELECT * FROM sportyshoes.category;`

Result Grid

id	name
1	Handmade
20	running shoes
21	climbing shoes
26	soccer shoes
27	sneakers
28	Leather ✓
NULL	NULL

4.updateCategory

The screenshot shows the Postman interface. On the left, the 'Collections' pane is expanded to 'first-spring-boot-app' > 'Sporty-Shoes' > 'Admin' > 'Category api'. The 'PUT updateCategory' endpoint is selected and highlighted with a red checkmark. The main panel shows the request configuration for 'localhost:8080/category'. The 'Body' tab is active, showing a JSON body:

```
{  "id": "28",  "name": "Leather shoes"}
```

 with red checkmarks next to the values. The 'Headers' tab shows 5 headers. The 'Test Results' tab is also visible.

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left shows the 'sportyshoes' database with tables 'category', 'hibernate_sequence', 'product', and 'user'. The 'category' table is selected. The main panel shows the query:

```
SELECT * FROM sportyshoes.category;
```

 The 'Result Grid' shows the following data:

id	name
1	Handmade
20	running shoes
21	climbing shoes
26	soccer shoes
27	sneakers
28	Leather shoes
NULL	NULL

The row with id 28 and name 'Leather shoes' is highlighted with a red checkmark.

5. deleteCategoryById

The screenshot shows the Postman API client interface. On the left, the 'Collections' pane is expanded to 'first-spring-boot-app' > 'Sporty-Shoes' > 'Admin' > 'Category api'. The 'deleteCategoryById' endpoint is selected, marked with a red checkmark. The main panel shows the 'DELETE' method and the URL 'localhost:8080/category/28', which is circled in red. Below the URL bar, the 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response in 'Pretty' format:

```
1 {
2   "id": 28,
3   "name": "Leather shoes"
4 }
```

The response is also marked with a red checkmark.

The screenshot shows the DBeaver database client interface. The 'SCHEMAS' pane on the left is expanded to 'sportyshoes' > 'Tables'. The 'category' table is selected. The main panel shows the SQL query 'SELECT * FROM sportyshoes.category;' and the 'Result Grid' tab. The result grid shows the following data:

id	name
1	Handmade
20	running shoes
21	climbing shoes
26	soccer shoes
27	sneakers
NULL	NULL

The 'category' table is circled in red. A tooltip for 'running shoes' is visible over the row with id 20. The bottom status bar indicates 'No object selected'.

*****Product API*****

1.saveProduct

The screenshot shows the Postman interface for the `saveProduct` endpoint. The endpoint is a `POST` request to `localhost:8080/product`. The request body is a JSON object with the following data:

```
{  "name": "Woodland",  "price": "6000",  "category": "climbing shoes"}
```

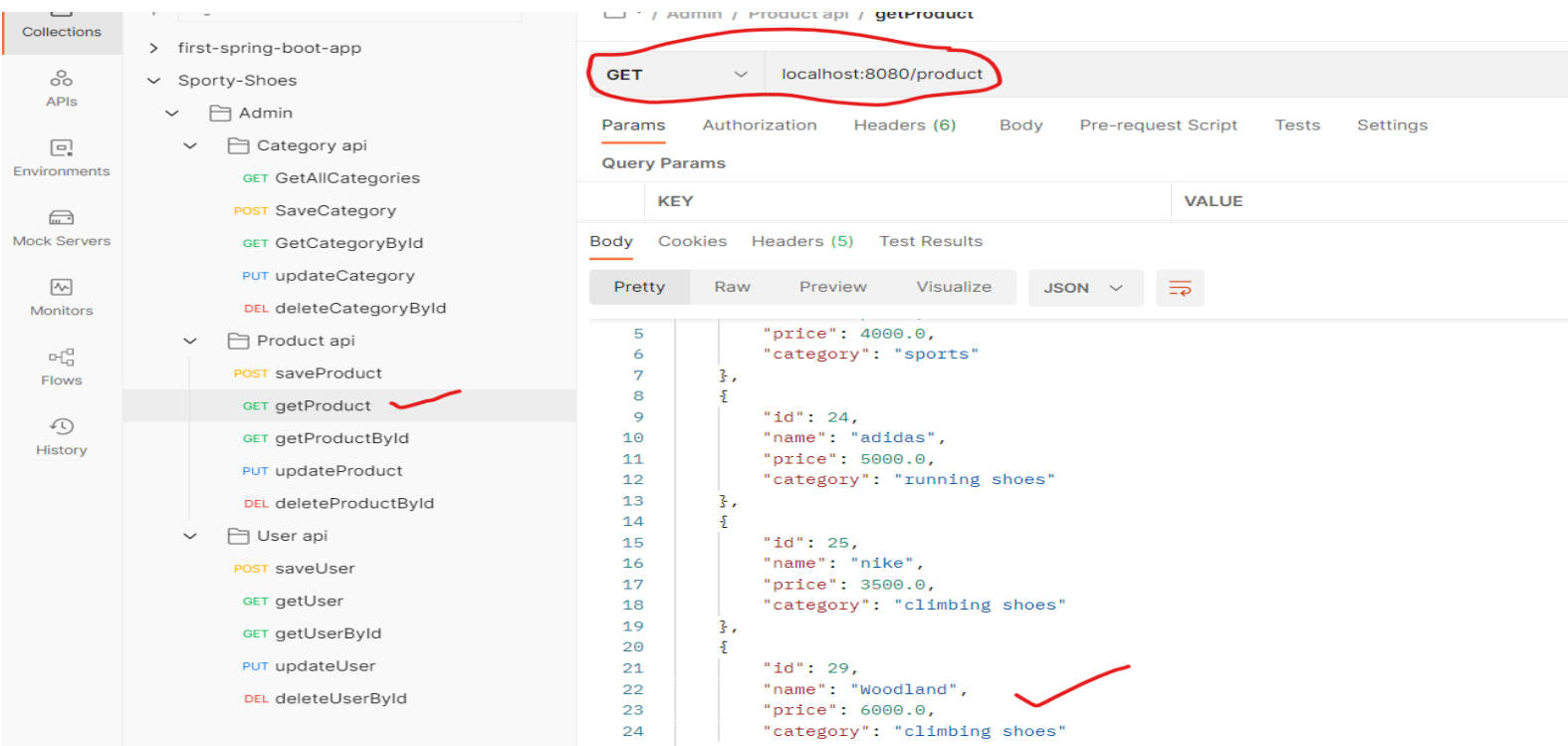
The response is a JSON object with the following data:

```
{  "id": 29,  "name": "Woodland",  "price": 6000.0,  "category": "climbing shoes"}
```

The screenshot shows the MySQL Workbench interface. The `product` table is selected, and the query `SELECT * FROM sportyshoes.product;` is executed. The result grid shows the following data:

id	category	name	price
23	sports	puma	4000
24	running shoes	adidas	5000
25	climbing shoes	nike	3500
29	climbing shoes	Woodland	6000
NULL	NULL	NULL	NULL

2.getProduct



The screenshot shows the Postman interface for the `GET /localhost:8080/product` endpoint. The response is a JSON array of shoe products. A red circle highlights the endpoint, and a red checkmark is next to the last item in the array.

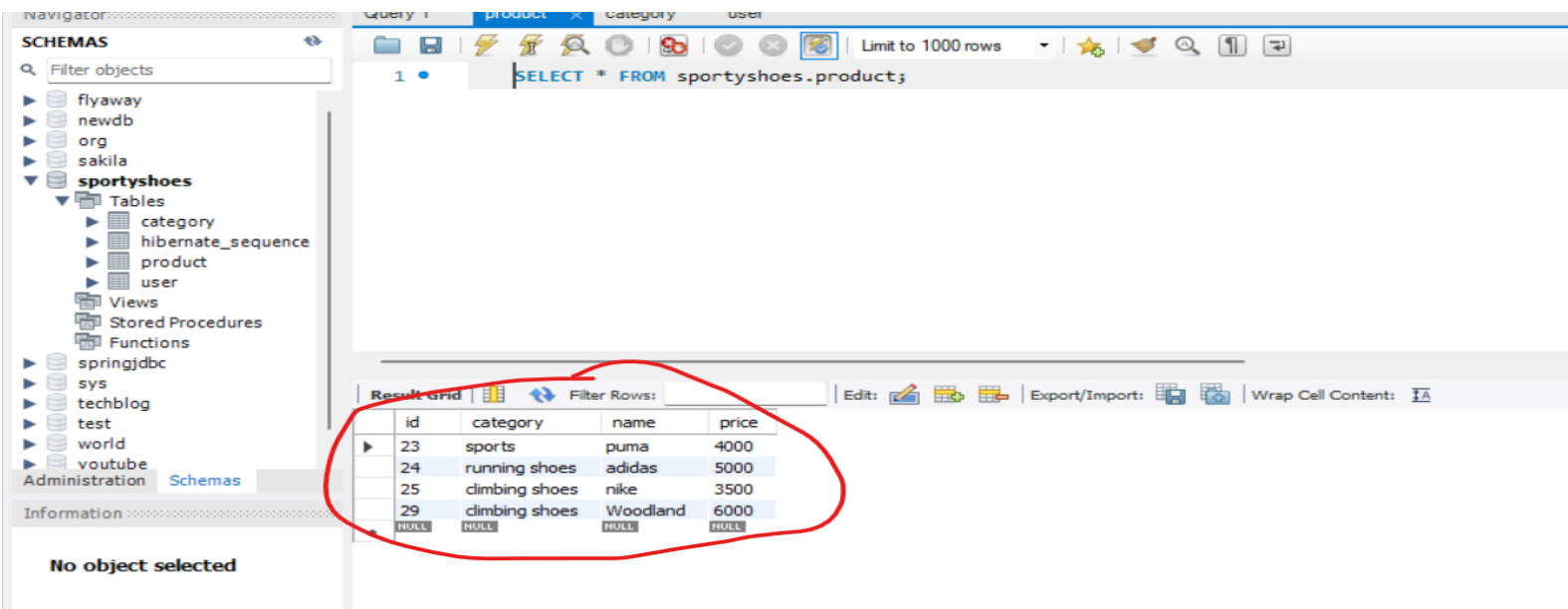
```
GET localhost:8080/product
```

Query Params

KEY	VALUE
-----	-------

Body

```
5 {
6   "price": 4000.0,
7   "category": "sports"
8 },
9 {
10  "id": 24,
11  "name": "adidas",
12  "price": 5000.0,
13  "category": "running shoes"
14 },
15 {
16  "id": 25,
17  "name": "nike",
18  "price": 3500.0,
19  "category": "climbing shoes"
20 },
21 {
22  "id": 29,
23  "name": "Woodland",
24  "price": 6000.0,
25  "category": "climbing shoes"
26 }
```



The screenshot shows the DBeaver interface with the SQL query `SELECT * FROM sportyshoes.product;` executed. The result is a table of shoe products. A red circle highlights the table data.

Query: `SELECT * FROM sportyshoes.product;`

id	category	name	price
23	sports	puma	4000
24	running shoes	adidas	5000
25	climbing shoes	nike	3500
29	climbing shoes	Woodland	6000
NULL	NULL	NULL	NULL

3.getProductById

The screenshot shows the Postman interface. On the left, the 'Collections' pane is expanded to 'Sporty-Shoes' > 'Admin' > 'Product api', where 'GET getProductById' is selected with a red checkmark. The main pane shows the request details for 'GET localhost:8080/product/29', which is circled in red. The 'Query Params' table is empty. The 'Body' tab is active, showing a JSON response in 'Pretty' format:

```
1 {
2   "id": 29,
3   "name": "Woodland",
4   "price": 6000.0,
5   "category": "climbing shoes"
6 }
```

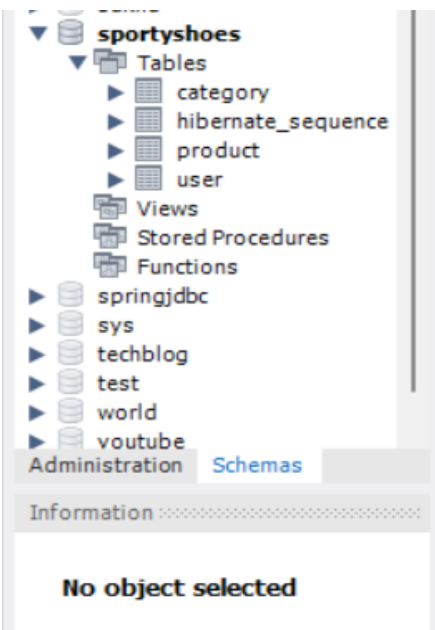
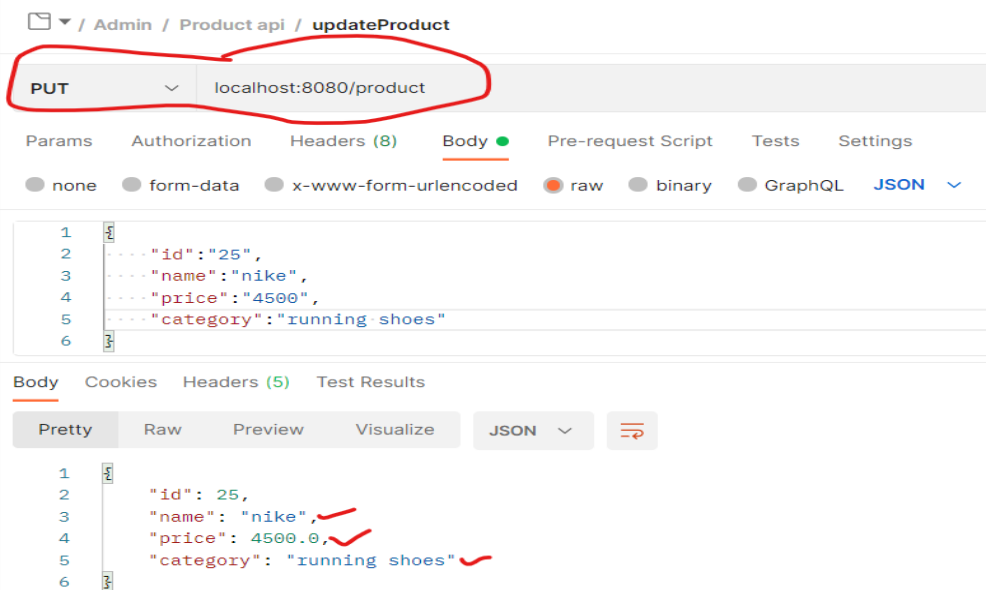
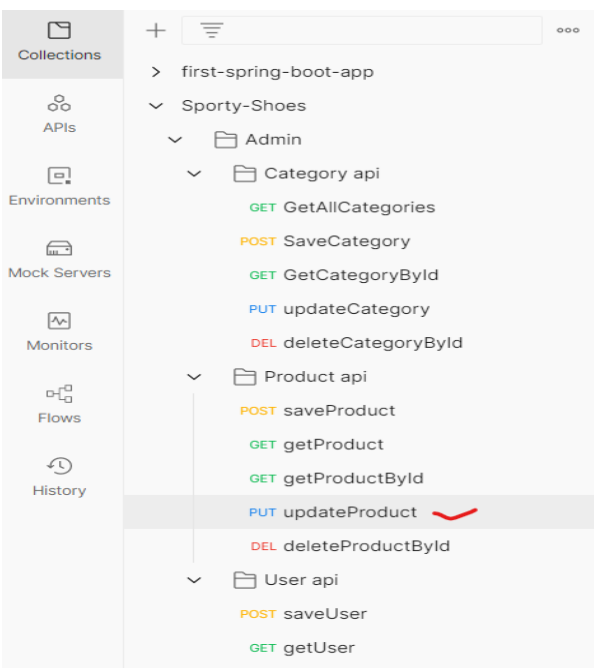
A red checkmark is next to the JSON response.

The screenshot shows the DBeaver interface. On the left, the 'Database' pane is expanded to 'sportyshoes' > 'Tables', where 'product' is selected. The main pane shows the 'Result Grid' with the following data:

	id	category	name	price
▶	23	sports	puma	4000
	24	running shoes	adidas	5000
	25	climbing shoes	nike	3500
	29	climbing shoes	Woodland	6000
•	NULL	NULL	NULL	NULL

A red checkmark is next to the row with id 29. The 'Information' pane at the bottom shows 'No object selected'.

4.updateProduct



The screenshot shows the 'Result Grid' for the 'product' table. The table has the following columns: id, category, name, and price. The data is as follows:

	id	category	name	price
▶	23	sports	puma	4000
	24	running shoes	adidas	5000
	25	running shoes	nike	4500
	29	climbing shoes	Woodland	6000
*	NULL	NULL	NULL	NULL

5.deleteProduct

The screenshot shows the Postman interface with the following details:

- Left Panel:** Collections > first-spring-boot-app > Sporty-Shoes > Admin > Category api > deleteCategoryByld (checked with a red checkmark). Product api > deleteProductByld is highlighted.
- Right Panel:** Method: DELETE, URL: localhost:8080/product/25 (circled in red).
 - Params:** Query Params table with 2 columns: KEY, VALUE. Row 1: Key, Value.
 - Body:** JSON tab selected. Content:

```
{  "id": 25,  "name": "nike",  "price": 4500.0,  "category": "running shoes"}
```

 (circled in red).

The screenshot shows the DBeaver database tool interface with the following details:

- Left Panel:** sportyshoes > Tables > product (selected).
- Right Panel:** Result Grid showing product data (circled in red).

	id	category	name	price
▶	23	sports	puma	4000
	24	running shoes	adidas	5000
	29	climbing shoes	Woodland	6000
	NULL	NULL	NULL	NULL

At the bottom, it says "No object selected".

*****User API*****

1.saveUser

The screenshot shows the Postman interface for the `saveUser` endpoint. The endpoint is a **POST** request to `localhost:8080/user`. The request body is in **JSON** format and contains the following data:

```
{  "name": "Kavya Tushar Supe",  "email": "ksupe222@gmail.com",  "password": "2017"}
```

The response is also in **JSON** format and contains the following data:

```
{  "id": 30,  "name": "Kavya Tushar Supe",  "email": "ksupe222@gmail.com",  "password": "2017"}
```

The screenshot shows the DBeaver interface for the `sportyshoes` database. The `user` table is selected, and the result grid shows the data returned by the `saveUser` API. The data is as follows:

id	email	name	password
14	tsupe222@gmail.com	Tushsar vijay Supe	12345
15	asupe222@gmail.com	Ashish vijay Supe	67890
16	vsupe222@gmail.com	Vijay kashinath supe	10 10 10
17	atsupe222@gmail.com	Aruna Tushar Supe	0987
18	vvvsupe222@gmail.com	Vidya Vijay Supe	5432
30	ksupe222@gmail.com	Kavya Tushar Supe	2017

2.getUser

Postman interface showing the GET request to `localhost:8080/user`. The response is a JSON array of user objects.

```
15 {
16   "id": 16,
17   "name": "Vijay kashinath supe",
18   "email": "vsupe222@gmail.com",
19   "password": "101010"
20 },
21 {
22   "id": 17,
23   "name": "Aruna Tushar Supe",
24   "email": "atsupe222@gmail.com",
25   "password": "0987"
26 },
27 {
28   "id": 18,
29   "name": "Vidya Vijay Supe",
30   "email": "vvsupe222@gmail.com",
31   "password": "5432"
32 },
33 {
34   "id": 30,
35   "name": "Kavya Tushar Supe",
36   "email": "ksupe222@gmail.com",
37   "password": "2017"
38 }
```

DBeaver interface showing the 'sportyshoes' database schema. The 'user' table is selected, and the 'Result Grid' shows the data.

id	email	name	password
14	tsupe222@gmail.com	Tushsar vijay Supe	12345
15	asupe222@gmail.com	Ashish vijay Supe	67890
16	vsupe222@gmail.com	Vijay kashinath supe	101010
17	atsupe222@gmail.com	Aruna Tushar Supe	0987
18	vvsupe222@gmail.com	Vidya Vijay Supe	5432
30	ksupe222@gmail.com	Kavya Tushar Supe	2017
NULL	NULL	NULL	NULL

3.getUserById

Collections

- first-spring-boot-app
 - Sporty-Shoes
 - Admin
 - Category api
 - GET GetAllCategories
 - POST SaveCategory
 - GET GetCategoryById
 - PUT updateCategory
 - DEL deleteCategoryById
 - Product api
 - POST saveProduct
 - GET getProduct
 - GET getProductById
 - PUT updateProduct
 - DEL deleteProductById
 - User api
 - POST saveUser
 - GET getUser
 - GET getUserById ✓
 - PUT updateUser
 - DEL deleteUserById

Admin / User api / getUserById

GET localhost:8080/user/30

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 2
2  "id": 30,
3  "name": "Kavya Tushar Supe",
4  "email": "ksupe222@gmail.com",
5  "password": "2017"
6 3
```

sportyshoes

- Tables
 - category
 - hibernate_sequence
 - product
 - user
- Views
- Stored Procedures
- Functions

springjdbc sys techblog test world youtube

Administration Schemas

Information

No object selected

Result Grid				
Filter Rows:				
	id	email	name	password
	14	tsupe222@gmail.com	Tushsar vijay Supe	12345
	15	asupe222@gmail.com	Ashish vijay Supe	67890
	16	vsupe222@gmail.com	Vijay kashinath supe	10 10 10
	17	atsupe222@gmail.com	Aruna Tushar Supe	0987
	18	vsupe222@gmail.com	Vidya Vijay Supe	5432
	30	ksupe222@gmail.com	Kavya Tushar Supe	2017 ✓
*	NULL	NULL	NULL	NULL

4.updateUser

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

```
{  "id": "18",  "email": "vvsupe111@gmail.com",  "name": "Vidya V supe",  "password": "202020"}
```

The response body is also shown in JSON format:

```
{  "id": 18,  "name": "Vidya V supe",  "email": "vvsupe111@gmail.com",  "password": "202020"}
```

Red annotations highlight the PUT method, the URL, and the response body.

The screenshot shows a database management tool interface with the following structure:

- sportyshoes**
 - Tables
 - category
 - hibernate_sequence
 - product
 - user
 - Views
 - Stored Procedures
 - Functions
- springjdbc
- sys
- techblog
- test
- world
- youtube
- Administration
- Schemas

Below the schema, it says "No object selected".

	id	email	name	password
	14	tsupe222@gmail.com	Tushsar vijay Supe	12345
	15	asupe222@gmail.com	Ashish vijay Supe	67890
	16	vsupe222@gmail.com	Vijay kashinath supe	101010
	17	atsupe222@gmail.com	Aruna Tushar Supe	0987
▶	18	vvsupe111@gmail.com	Vidya V supe	202020
	30	ksupe222@gmail.com	Kavya Tushar Supe	2017
	NULL	NULL	NULL	NULL

5.deleteUser

