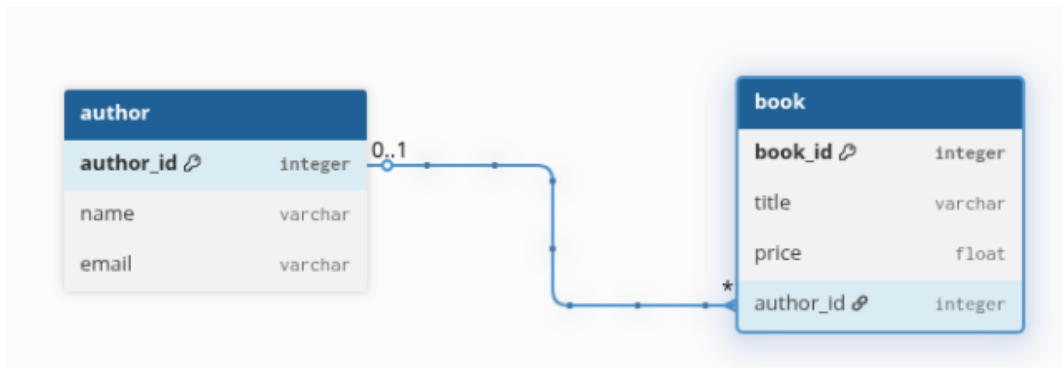


# PYTHON PROJECT: BOOK INVENTORY MANAGEMENT

## Objective:

Create a simple API to manage books and authors in a bookstore. Create tables in Python (SQLAlchemy).

## Schema:



Aliza Dangi

## Author API:

Add a new author:

The screenshot shows a REST client interface with the following details:

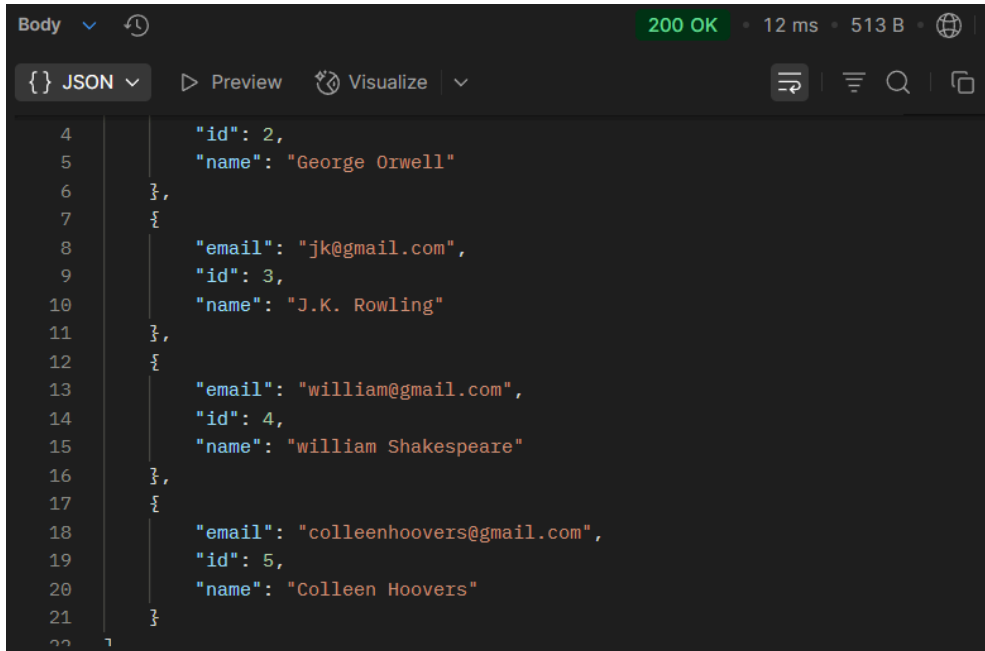
- URL:** `http://127.0.0.1:5000/authors`
- Method:** `GET`
- Response Status:** `200 OK` (24 ms, 513 B)
- Response Body (JSON):**

```
{
  "name": "Colleen Hoovers",
  "email": "colleenhoovers@gmail.com"
}
```
- Response Body (JSON Array):**

```
[
  {
    "id": 2,
    "name": "George Orwell"
  },
  {
    "email": "jk@gmail.com",
    "id": 3,
    "name": "J.K. Rowling"
  },
  {
    "email": "william@gmail.com",
    "id": 4,
    "name": "william Shakespeare"
  }
]
```

Aliza Dangi

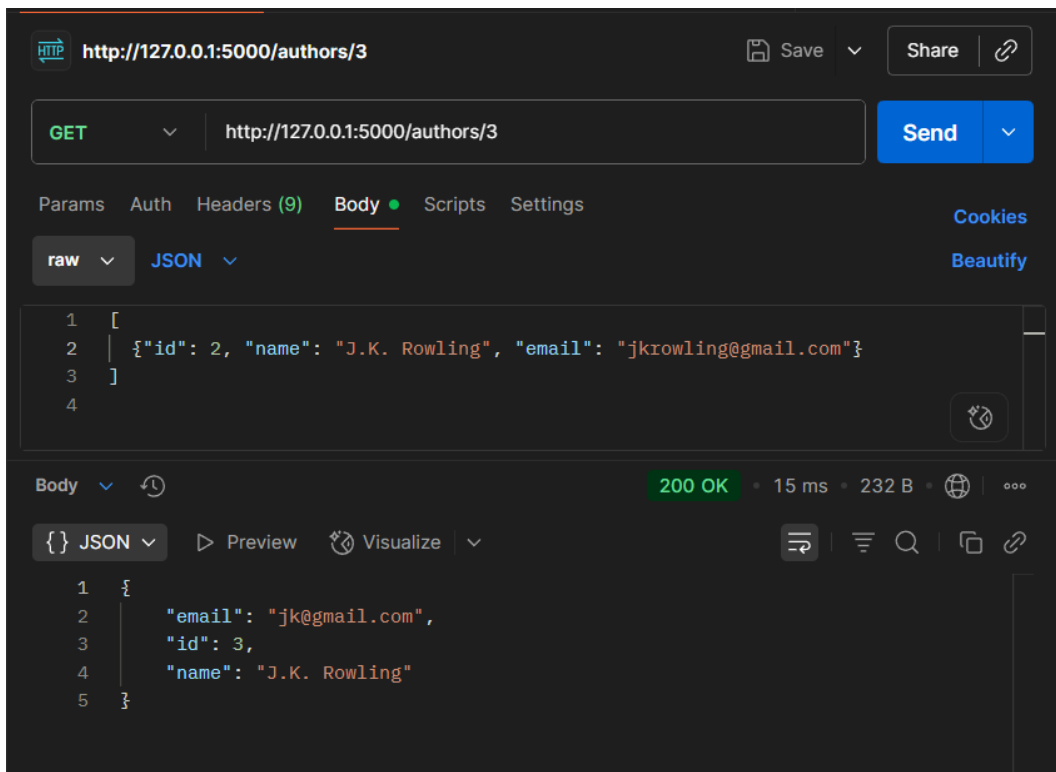
## Get All Authors:



A screenshot of a REST client interface. The top bar shows 'Body' with a dropdown arrow, a refresh icon, and a status bar indicating '200 OK', '12 ms', and '513 B'. Below the bar, there are tabs for 'JSON' (selected), 'Preview', and 'Visualize'. The main area displays a JSON array of author objects. The JSON is formatted with line numbers on the left.

```
4      "id": 2,  
5      "name": "George Orwell"  
6    },  
7    {  
8      "email": "jk@gmail.com",  
9      "id": 3,  
10     "name": "J.K. Rowling"  
11   },  
12   {  
13     "email": "william@gmail.com",  
14     "id": 4,  
15     "name": "william Shakespeare"  
16   },  
17   {  
18     "email": "colleenhoovers@gmail.com",  
19     "id": 5,  
20     "name": "Colleen Hoovers"  
21   }  
22 ]
```

## Get Author by ID:



A screenshot of a REST client interface. The top bar shows the URL 'http://127.0.0.1:5000/authors/3' with a dropdown arrow, a 'Save' button, and a 'Share' button. Below the bar, there are tabs for 'GET' (selected), 'POST', 'PUT', and 'DELETE'. The main area displays the response for the GET request. The response is a JSON array containing one object. The JSON is formatted with line numbers on the left.

```
1  [  
2    {"id": 2, "name": "J.K. Rowling", "email": "jknowling@gmail.com"}  
3  ]  
4
```

Below the main area, there is a section for 'Body' with a dropdown arrow, a refresh icon, and a status bar indicating '200 OK', '15 ms', and '232 B'. Below this, there are tabs for 'JSON' (selected), 'Preview', and 'Visualize'. The main area displays the JSON object for the author with ID 2.

```
1  {  
2    "email": "jk@gmail.com",  
3    "id": 3,  
4    "name": "J.K. Rowling"  
5  }
```

Aliza Dangi

## Update Author Email:

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:5000/authors/3`. The request body is a JSON object: `{ "email": "rowling.updated@gmail.com" }`. The response is a 200 OK status with a response time of 25 ms and a body size of 218 B. The response body is a JSON object: `{ "message": "Author email updated successfully" }`.

```
PUT http://127.0.0.1:5000/authors/3
```

Params Auth Headers (9) **Body** Scripts Settings Cookies Beautify

raw JSON

```
1 {
2   "email": "rowling.updated@gmail.com"
3 }
4
```

Body 200 OK • 25 ms • 218 B •

{ } JSON Preview Visualize

```
1 {
2   "message": "Author email updated successfully"
3 }
```

```
"email": "colleenhoovers@gmail.com",
"id": 5,
"name": "Colleen Hoovers"
},
{
  "email": "rowling.updated@gmail.com",
  "id": 3,
  "name": "J.K. Rowling"
}
```

Aliza Dangi

## Delete Author:

The screenshot shows a REST client interface with the following details:

- Request:** Method **DELETE**, URL **http://127.0.0.1:5000/authors/3**.
- Body:** A JSON object: 

```
{  "email": "rowling.updated@gmail.com"}
```
- Response:** Status **200 OK**, 32 ms, 228 B. The response body is a JSON object: 

```
{  "message": "Author and their books deleted successfully"}
```

```
[  {    "email": "george@gmail.com",    "id": 2,    "name": "George Orwell"  },  {    "email": "william@gmail.com",    "id": 4,    "name": "william Shakespeare"  },  {    "email": "colleenhoovers@gmail.com",    "id": 5,    "name": "Colleen Hoovers"  }]
```

Aliza Dangi

## Books API:

### Add a Book:

The screenshot shows a REST client interface with a dark theme. At the top, a **POST** method is selected for the URL `http://127.0.0.1:5000/books`. A **Send** button is on the right. Below the URL bar, tabs for **Params**, **Auth**, **Headers (9)**, **Body** (selected), **Scripts**, and **Settings** are visible. On the right side of the **Body** tab, there are links for **Cookies** and **Beautify**. The **Body** tab has a sub-menu with **raw** and **JSON** (selected). The main area displays the JSON body of the request:

```
1 {  
2   "title": "Animal Farm",  
3   "price": 20.5,  
4   "author_id": 2  
5 }  
6
```

Below the request body, a status bar shows **201 CREATED** in a green box, followed by response details: **14 ms**, **213 B**, a globe icon, and a menu icon. Below this, there are buttons for **JSON** (selected), **Preview**, and **Visualize**. A toolbar on the right contains icons for expand, collapse, search, copy, and share. The response body is shown in JSON format:

```
1 {  
2   "message": "Book added successfully"  
3 }
```

## Get All Books:

The screenshot displays a REST client interface with a GET request to `http://127.0.0.1:5000/books`. The response is a JSON array of two book objects, each with an author object. The top panel shows the raw JSON response, and the bottom panel shows the same response formatted and beautified.

**Request:**

```
GET http://127.0.0.1:5000/books
```

**Response (raw):**

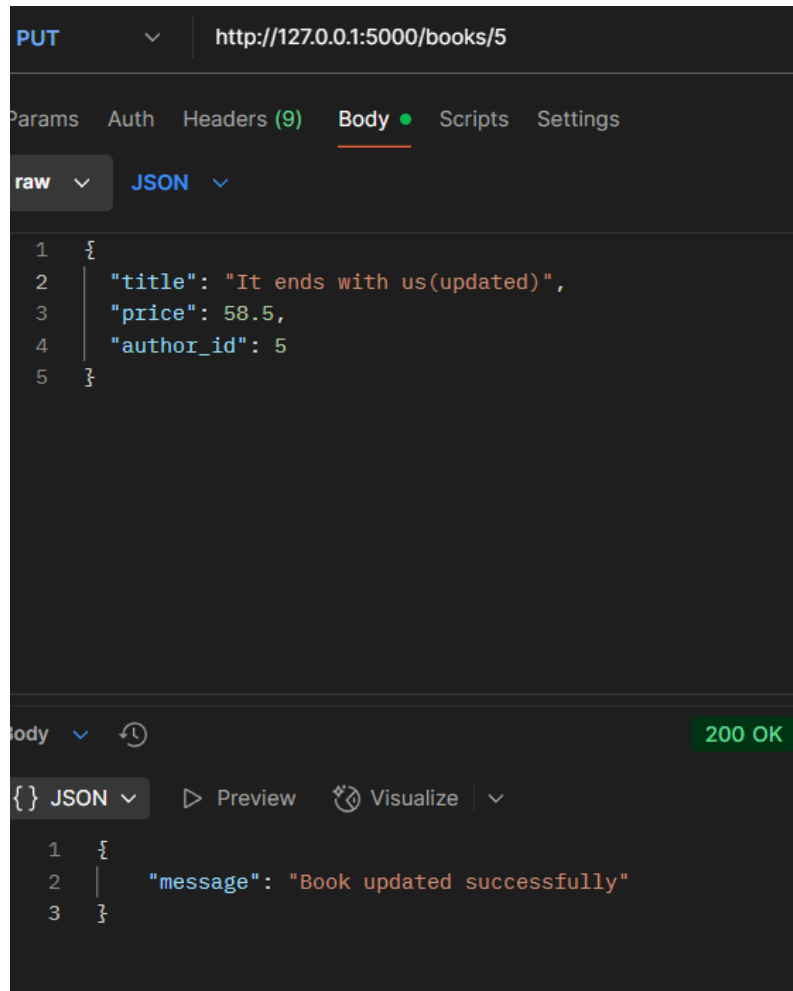
```
1 {
2   "title": "It ends with us",
3   "price": 58.5,
4   "author_id": 5
5 }
```

**Response (beautified):**

```
31 {
32   {
33     "author": {
34       "email": "george@gmail.com",
35       "id": 2,
36       "name": "George Orwell"
37     },
38     "id": 5,
39     "price": 20.5,
40     "title": "Animal Farm"
41   },
42   {
43     "author": {
44       "email": "william@gmail.com",
45       "id": 4,
46       "name": "william Shakespeare"
47     },
48     "id": 6,
```

Aliza Dangi

## Update Book:



```
    "title": "It ends with us",
  },
  {
    "author": {
      "email": "colleenhoovers@gmail.com",
      "id": 5,
      "name": "Colleen Hoovers"
    },
    "id": 5,
    "price": 58.5,
    "title": "It ends with us(updated)"
  }
}
```



Aliza Dangi

## Delete Book:

The screenshot displays a REST client interface with a DELETE request to `http://127.0.0.1:5000/books/2`. The request body is a JSON object: `{ "title": "It ends with us(updated)", "price": 58.5, "author_id": 5 }`. The response is a 200 OK status with a JSON body: `{ "message": "Book deleted successfully" }`.

**DELETE** `http://127.0.0.1:5000/books/2`

Params Auth Headers (9) **Body** Scripts Settings

raw **JSON**

```
1 {
2   "title": "It ends with us(updated)",
3   "price": 58.5,
4   "author_id": 5
5 }
```

Body `200 OK`

**{}** JSON Preview Visualize

```
1 {
2   "message": "Book deleted successfully"
3 }
```

## Dump Data to JSON

### Dump Authors:

A file got created in dumps/authors.json

The screenshot displays a REST client interface with a POST request to `http://127.0.0.1:5000/dump/authors`. The request body is a JSON object with the following structure:

```
1 {
2   "title": "It ends with us(updated)",
3   "price": 58.5,
4   "author_id": 5
5 }
```

The response status is `200 OK`. The response body is a JSON object:

```
1 {
2   "message": "Authors dumped to dumps/authors.json"
3 }
```

The interface includes tabs for Params, Auth, Headers (9), Body (selected), Scripts, and Settings. Below the request body, there are tabs for raw and JSON (selected). At the bottom, there are tabs for Body (selected), Preview, and Visualize.

## Dump Books:

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:5000/dump/books`. The request body is a JSON object with the following fields:

```
{
  "title": "It ends with us(updated)",
  "price": 58.5,
  "author_id": 5
}
```

The response status is **200 OK**. The response body is a JSON object with the following field:

```
{
  "message": "Books dumped to dumps/books.json"
}
```

The interface includes tabs for Params, Auth, Headers (9), Body (selected), Scripts, and Settings. The Body tab is further divided into raw and JSON (selected) views. The JSON view shows the response body with line numbers 1, 2, and 3.