6. Customize Collection Run Order and Data Driven Testing

Customize Collection Run Order

➔ By default, Postman executes requests in the order they are created in the collection. We can customize this order using the **`pm.execution.setNextRequest()`** function in Post-response to define which request to run next or to stop execution altogether.

Simple Books API

➔ The Simple Books API allows users to reserve books and perform various actions like viewing, ordering, updating, and deleting orders.

- **API Base URL:** https://simple-books-api.glitch.me
- **Books API Documentation:** https://www.postman.com/red-sunset-843133/book-api/documentation/4p0r4cu/book-api

➔ **Fork** Book API Collection or **Export** and try to import that collection to workspace and rename to **Day6_BooksAPI.**

API Authentication (Create Bearer Token) → Create Token

1. To access certain features, we need to register **our API client** and obtain a Bearer token.
2. **Endpoint: POST** https://simple-books-api.glitch.me/api-clients/
3. **Request Body (JSON) → Required**

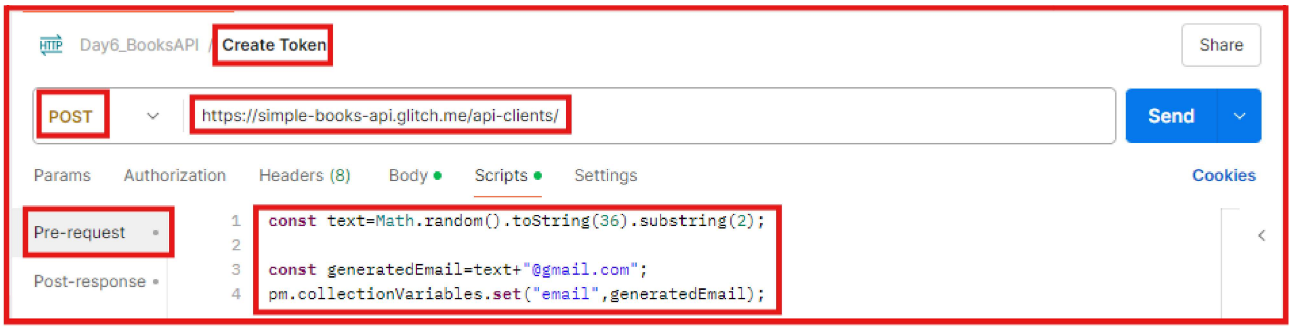
```
{
   "clientName": "KMR",
   "clientEmail": "kmrabc@example.com"
}
```

4. The response contains an access token valid for 7 days.
5. We will get **Status Code 409: "API client already registered."** if we use same clientName or clientEmail. So a different clientName or clientEmail to be used to resolve.
6. Generate random text for email instead of duplicate email in the prequest script of request and send the request.
   a. `const text = Math.random().toString(36).substring(2);`
   b. `const genratedEmail = text+"@gmail.com";`
   c. `console.log(genratedEmail );`

Note

➔ We don't need a token for **Check API Status, List Books, or Get Single Book APIs;** for Order related APIs, a Bearer Token is required.

→ To know what type of Authentication an API requires we can get information only from documentation.



Day6_BooksAPI / Create Token

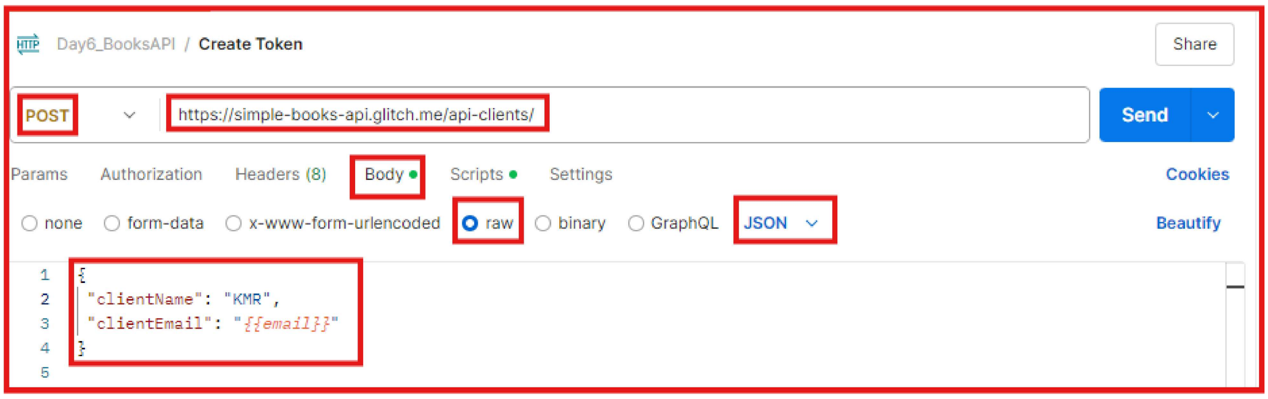
POST https://simple-books-api.glitch.me/api-clients/

Params | Authorization | Headers (8) | Body | Scripts | Settings

Pre-request

```
const text=Math.random().toString(36).substring(2);

const generatedEmail=text+"@gmail.com";
pm.collectionVariables.set("email",generatedEmail);
```

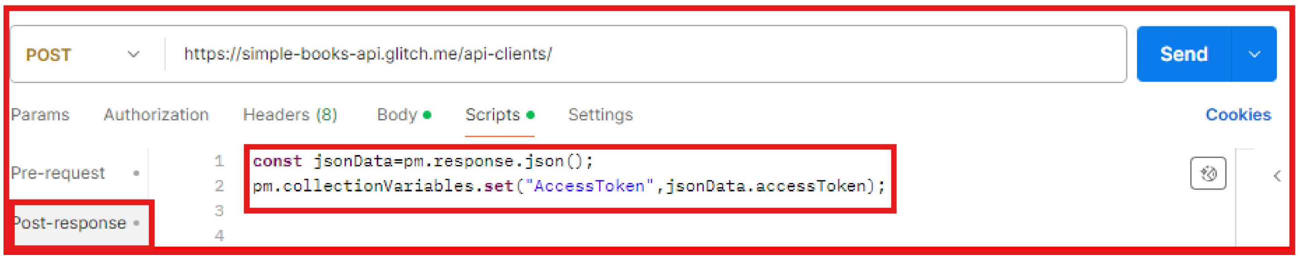


Day6_BooksAPI / Create Token

POST https://simple-books-api.glitch.me/api-clients/

```
{
 "clientName": "KMR",
 "clientEmail": "{{email}}"
}
```

7. Capture accessToken as a **collection variable** after response in Post-response script

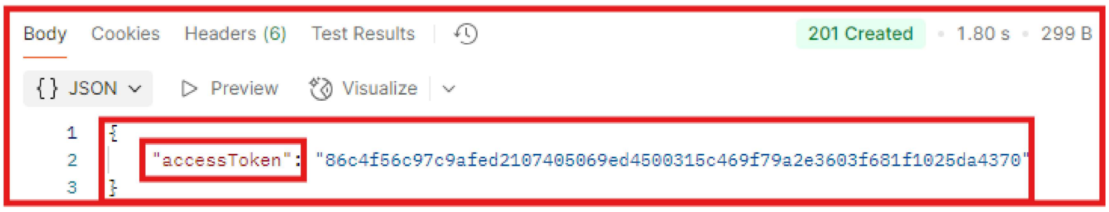


POST https://simple-books-api.glitch.me/api-clients/

Scripts → Post-response

```
const jsonData=pm.response.json();
pm.collectionVariables.set("AccessToken",jsonData.accessToken);
```

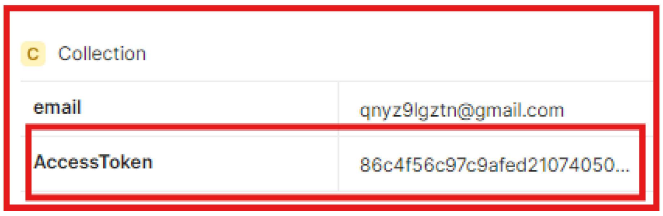
8. Below is the Response with accessToken



201 Created · 1.80 s · 299 B

```
{
 "accessToken": "86c4f56c97c9afed2107405069ed4500315c469f79a2e3603f681f1025da4370"
}
```

9. AccessToken is also created as a **collection variable** in Variables



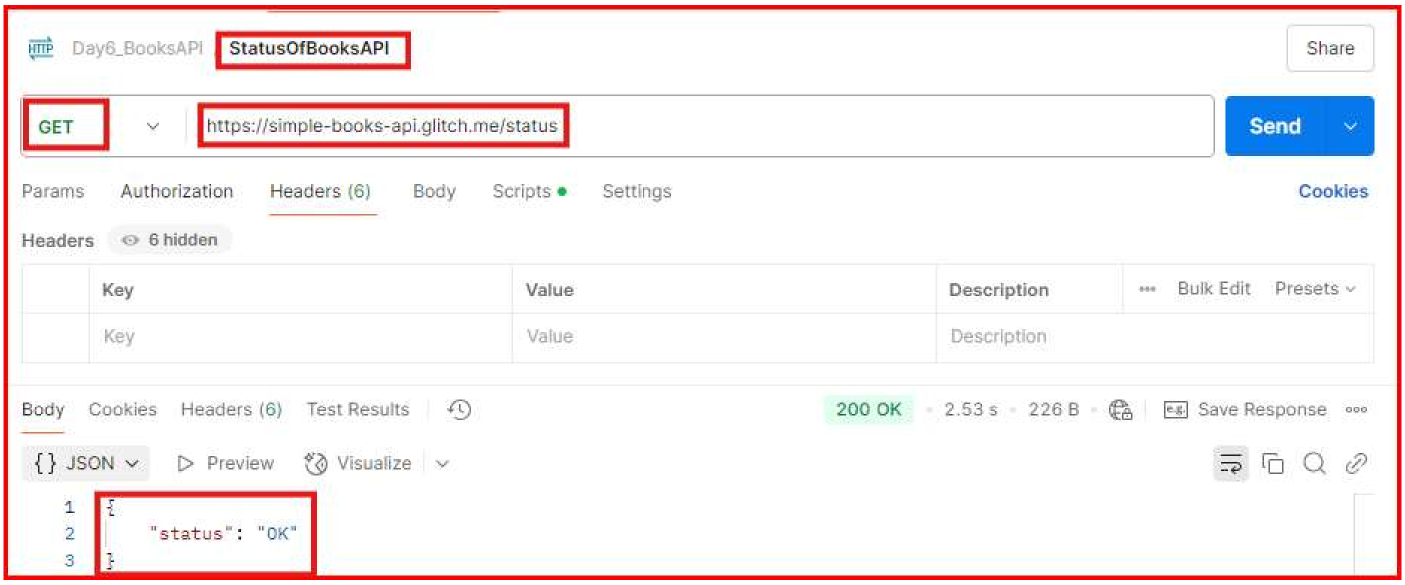
Collection

email	qnyz9lgztn@gmail.com
AccessToken	86c4f56c97c9afed21074050...

Check API Status → StatusOfBooksAPI

1. Retrieve the current status of the API.

2. **Endpoint:** GET https://simple-books-api.glitch.me/status


Day6_BooksAPI StatusOfBooksAPI

GET https://simple-books-api.glitch.me/status

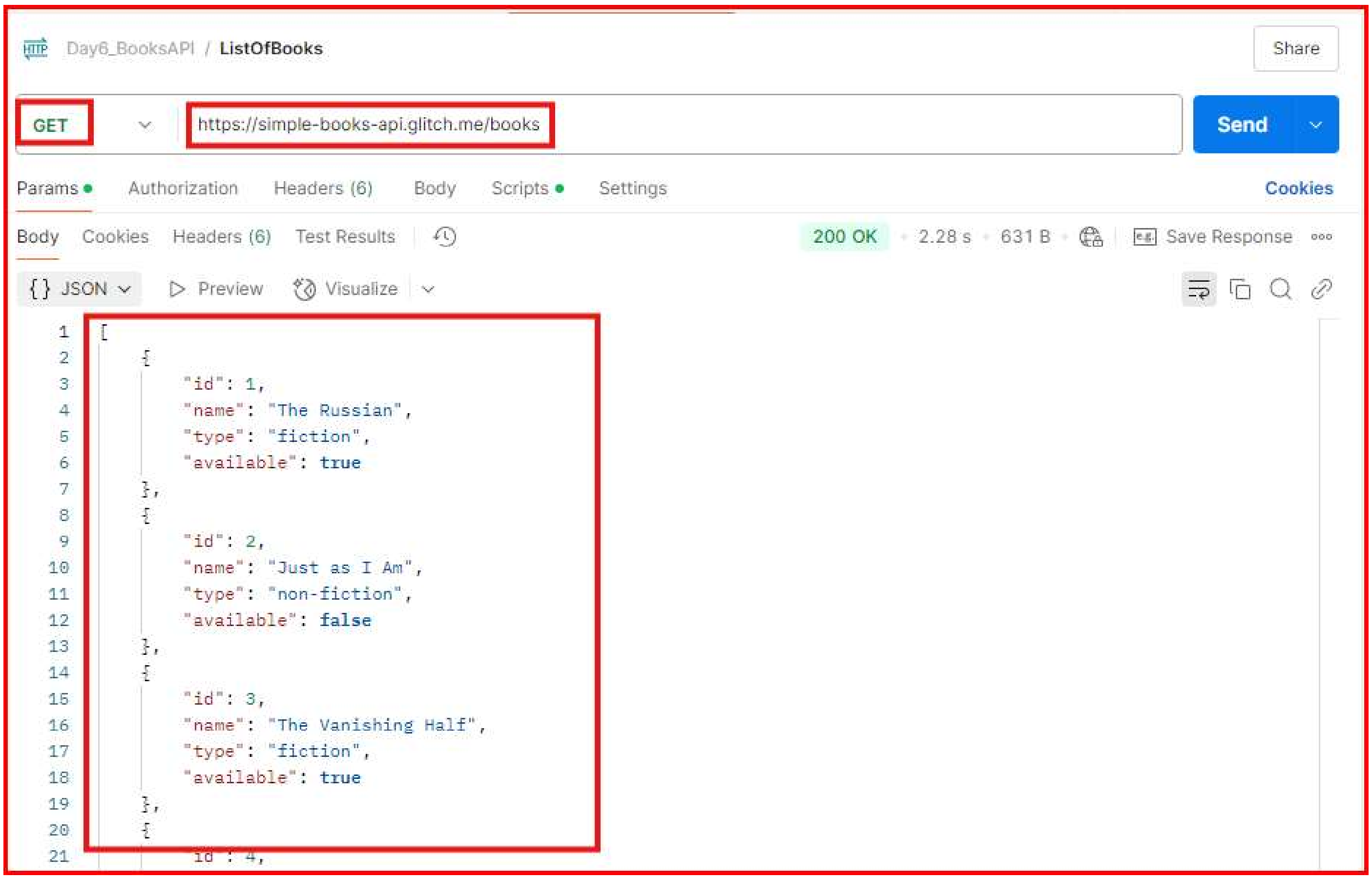
200 OK 2.53 s 226 B

```
{
    "status": "OK"
}
```

3. If Status is OK then we can access books API.

List Books → ListOfBooks

1. Get a list of available books.
2. **Endpoint:** GET https://simple-books-api.glitch.me/books

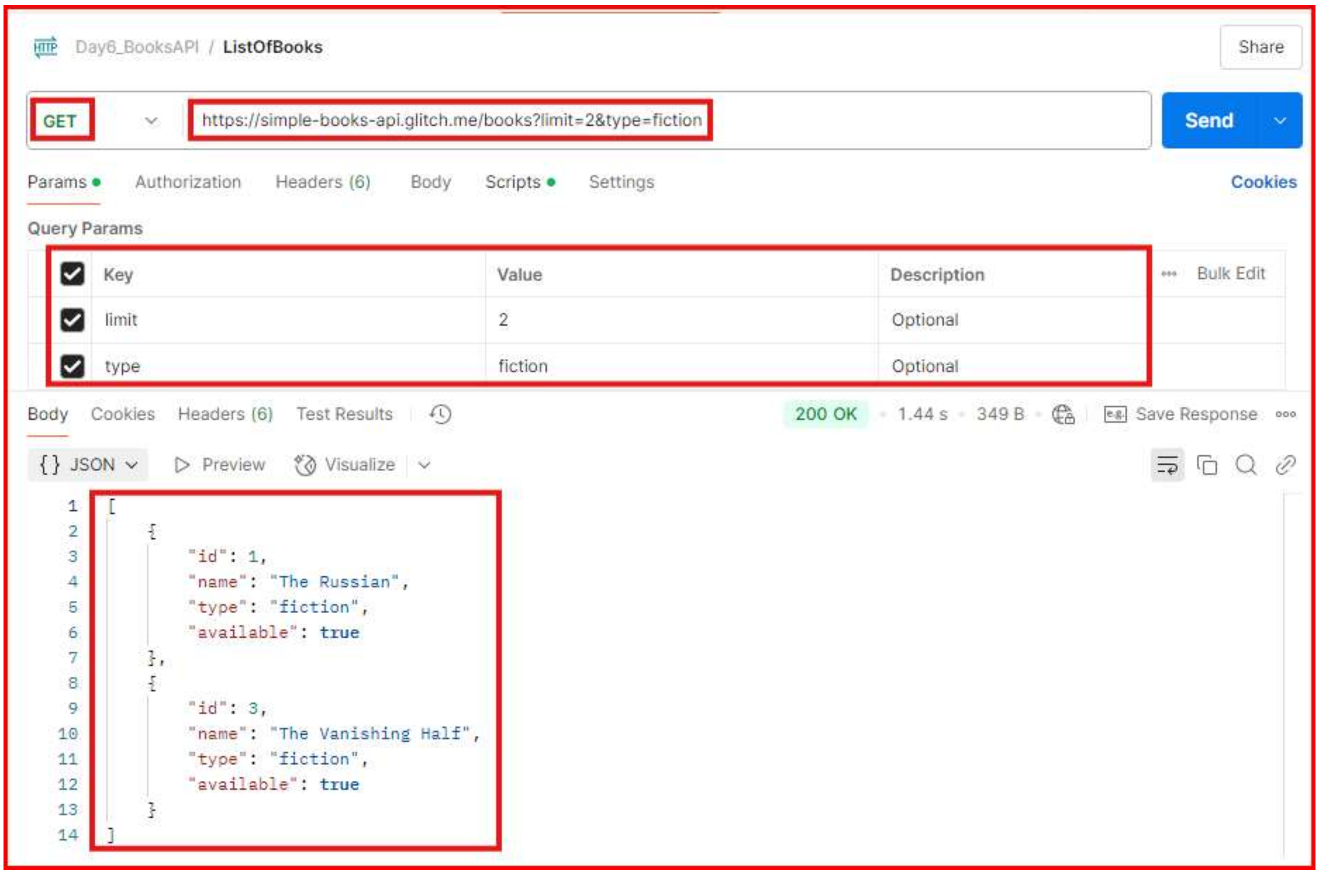

Day6_BooksAPI / ListOfBooks

GET https://simple-books-api.glitch.me/books

200 OK 2.28 s 631 B

```
[
    {
        "id": 1,
        "name": "The Russian",
        "type": "fiction",
        "available": true
    },
    {
        "id": 2,
        "name": "Just as I Am",
        "type": "non-fiction",
        "available": false
    },
    {
        "id": 3,
        "name": "The Vanishing Half",
        "type": "fiction",
        "available": true
    },
    {
        "id": 4,
```

3. **Optional Query Parameters**
   a. **type (String)** - Specify "fiction" or "non-fiction".
   b. **limit (Integer)** - A number between 1 and 20 to limit results.



GET https://simple-books-api.glitch.me/books?limit=2&type=fiction

Query Params

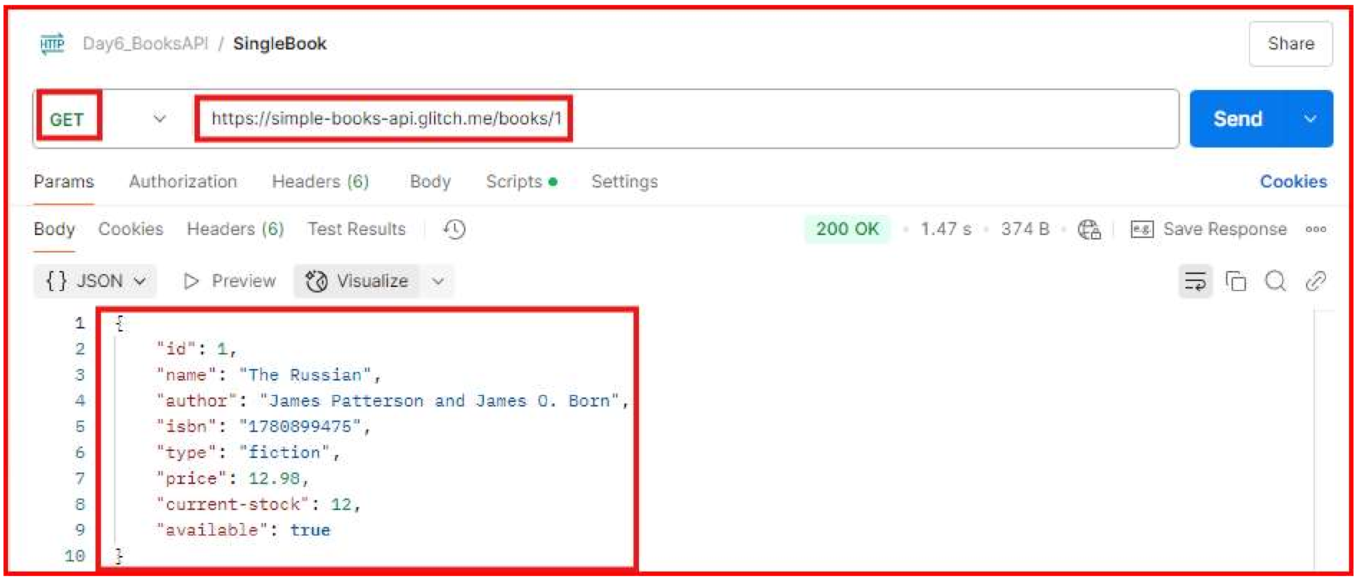
Key	Value	Description
limit	2	Optional
type	fiction	Optional

200 OK · 1.44 s · 349 B

```
[
    {
        "id": 1,
        "name": "The Russian",
        "type": "fiction",
        "available": true
    },
    {
        "id": 3,
        "name": "The Vanishing Half",
        "type": "fiction",
        "available": true
    }
]
```

Get Single Book → SingleBook

1. Retrieve detailed information about a specific book.
2. **Endpoint:** GET https://simple-books-api.glitch.me/books/**{bookId}**
   a. **Example:** GET https://simple-books-api.glitch.me/books/1 (where 1 is the bookId).



GET https://simple-books-api.glitch.me/books/1

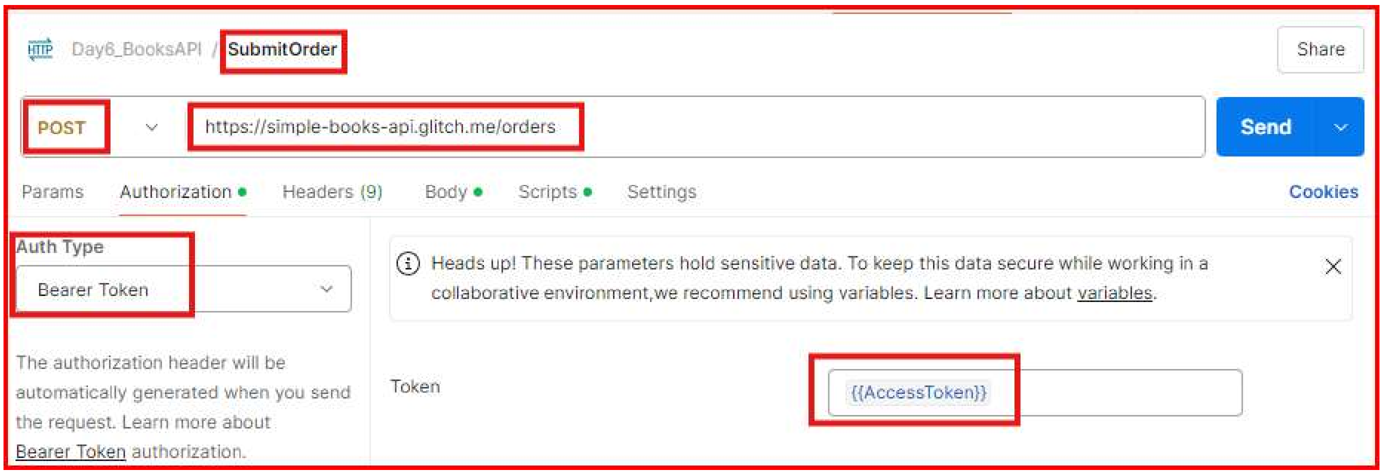
200 OK · 1.47 s · 374 B

```
{
    "id": 1,
    "name": "The Russian",
    "author": "James Patterson and James O. Born",
    "isbn": "1780899475",
    "type": "fiction",
    "price": 12.98,
    "current-stock": 12,
    "available": true
}
```

Submit an Order → SubmitOrder

1. Place a new book order. Requires **Bearer token** authentication.
2. **Endpoint: POST** https://simple-books-api.glitch.me/orders
3. **Request Body (JSON) → Required**

```
{
  "bookId" : 1,
  "customerName" : "John"
}
```



Day6_BooksAPI / SubmitOrder

POST https://simple-books-api.glitch.me/orders

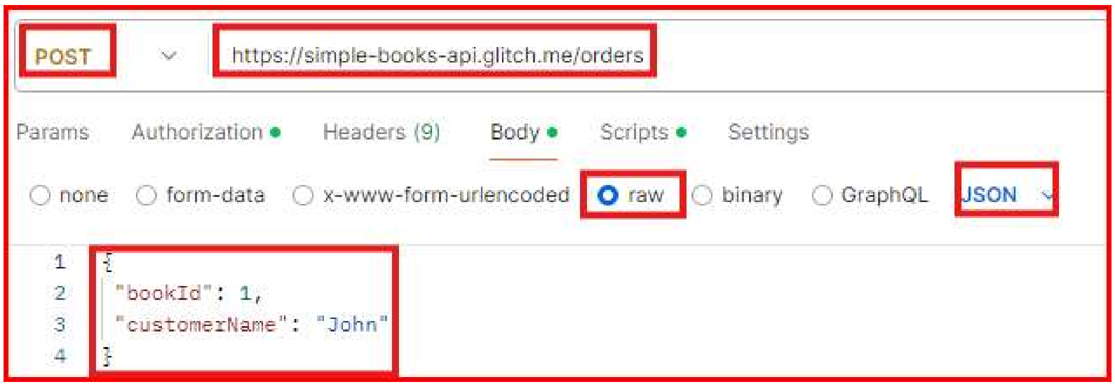
Params | Authorization ● | Headers (9) | Body ● | Scripts ● | Settings

Auth Type Bearer Token

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

Token {{AccessToken}}

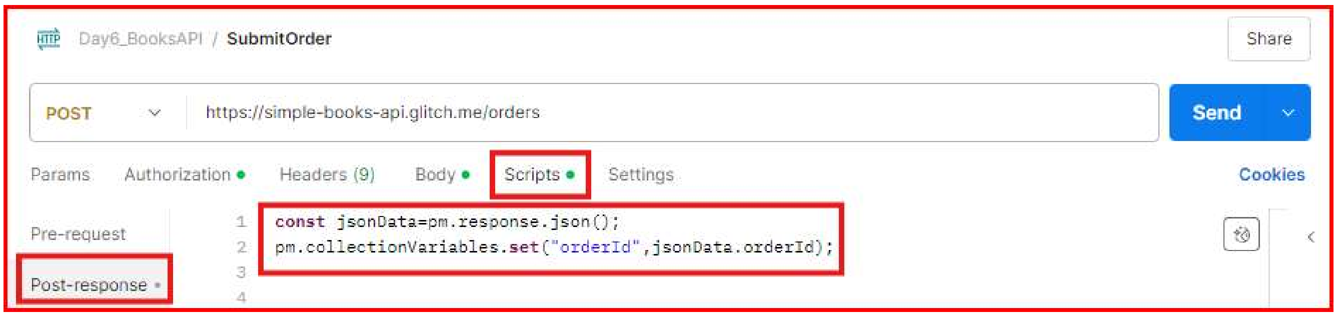


POST https://simple-books-api.glitch.me/orders

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
  "bookId": 1,
  "customerName": "John"
}
```

4. The response contains the **orderId**. We can save it as **collection or environmental variable** so that we can use for API Chaining or in another requests.

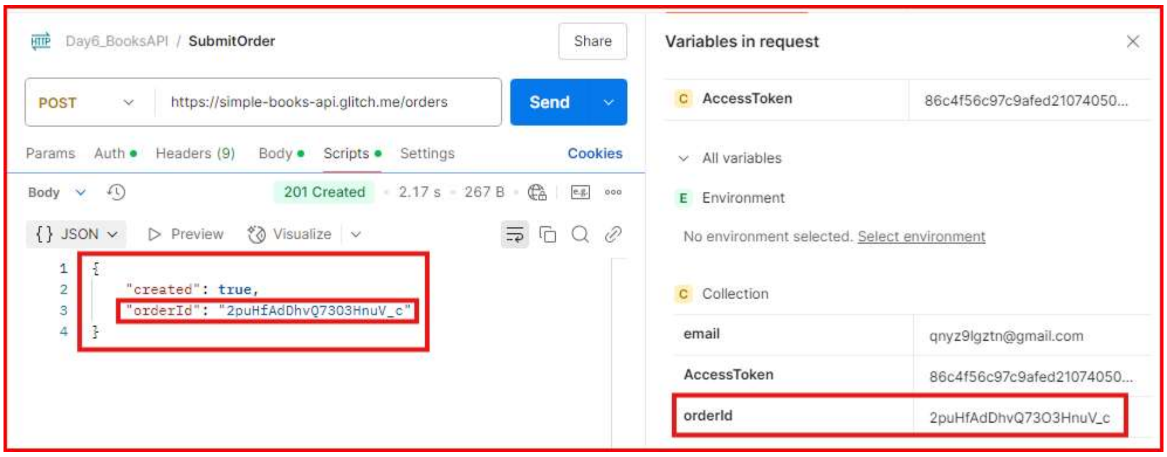


Day6_BooksAPI / SubmitOrder

POST https://simple-books-api.glitch.me/orders

Scripts ● | Pre-request | Post-response

```
const jsonData=pm.response.json();
pm.collectionVariables.set("orderId",jsonData.orderId);
```



Day6_BooksAPI / SubmitOrder

POST https://simple-books-api.glitch.me/orders

201 Created · 2.17 s · 267 B

```
{
  "created": true,
  "orderId": "2puHfAdDhvQ7303HnuV_c"
}
```

Variables in request

AccessToken	86c4f56c97c9afed21074050...

All variables

Environment

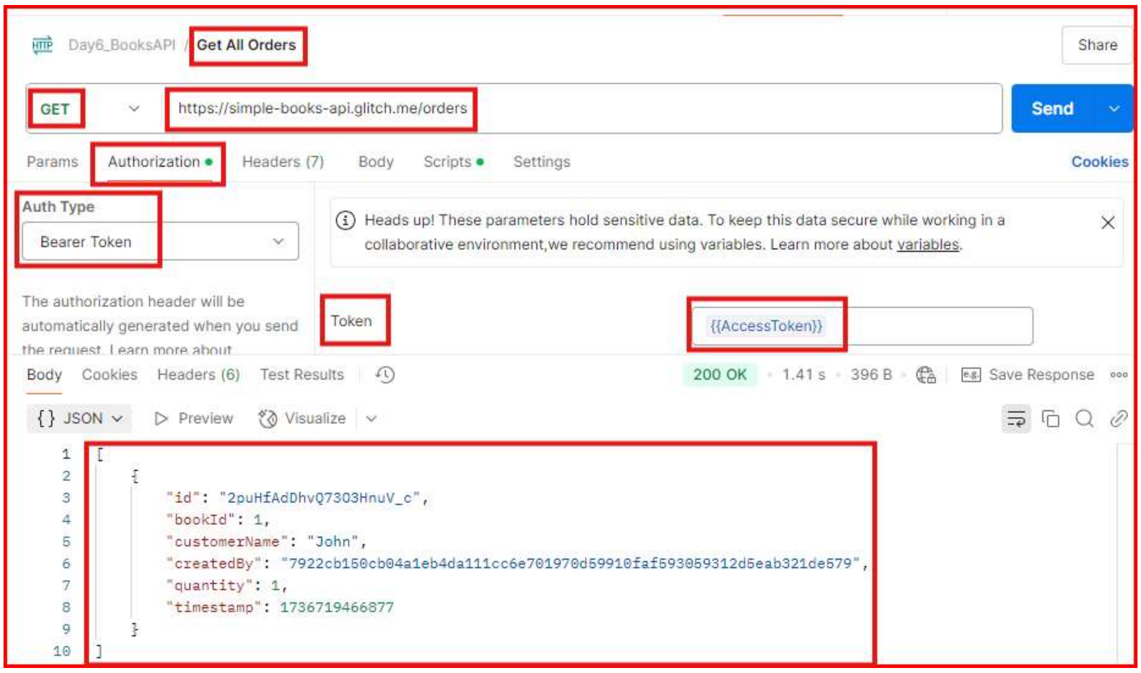
No environment selected. Select environment

Collection

email	qnyz9lgztn@gmail.com
AccessToken	86c4f56c97c9afed21074050...
orderId	2puHfAdDhvQ7303HnuV_c

Get All Orders → Get All Orders

1. View all existing orders. Requires **Bearer token** authentication.
2. **Endpoint: GET** https://simple-books-api.glitch.me/orders



GET https://simple-books-api.glitch.me/orders

Auth Type: Bearer Token

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

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.

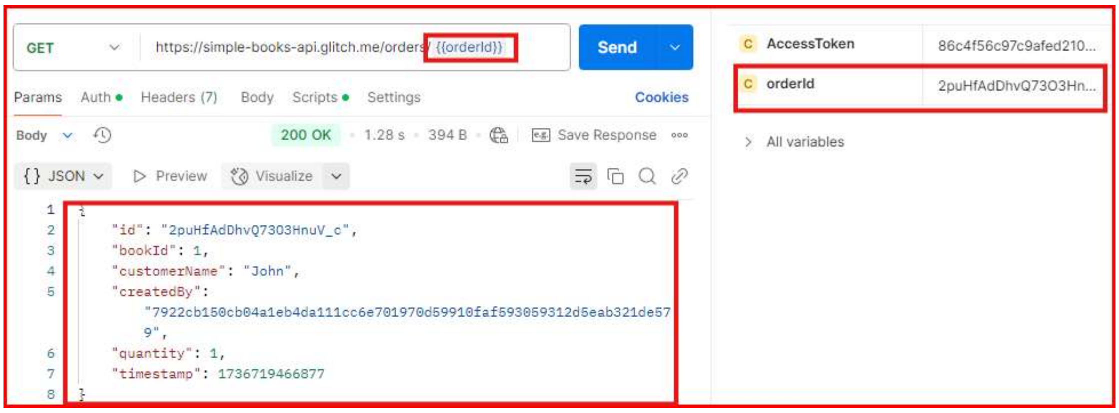
Token: {{AccessToken}}

200 OK · 1.41 s · 396 B

```
[
    {
        "id": "2puHfAdDhvQ7303HnuV_c",
        "bookId": 1,
        "customerName": "John",
        "createdBy": "7922cb150cb04a1eb4da111cc6e701970d59910faf593059312d5eab321de579",
        "quantity": 1,
        "timestamp": 1736719466877
    }
]
```

Get a Single Order → Get Single Order

1. View details of a specific order. Requires **Bearer token** authentication.
2. **Endpoint: GET** https://simple-books-api.glitch.me/orders/**{{orderId}}**



GET https://simple-books-api.glitch.me/orders/{{orderId}}

200 OK · 1.28 s · 394 B

AccessToken	86c4f56c97c9afed210...
orderId	2puHfAdDhvQ7303Hn...

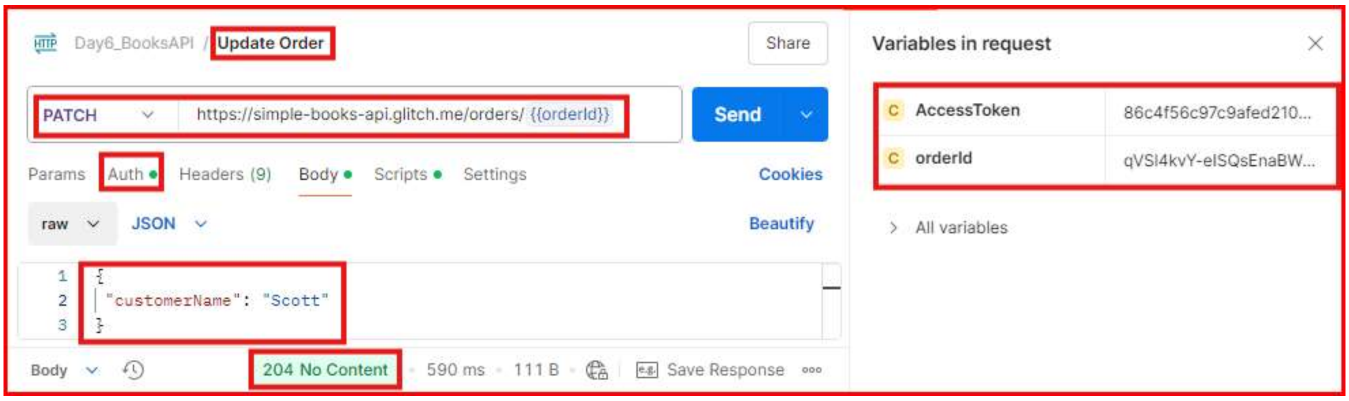
```
{
    "id": "2puHfAdDhvQ7303HnuV_c",
    "bookId": 1,
    "customerName": "John",
    "createdBy":
        "7922cb150cb04a1eb4da111cc6e701970d59910faf593059312d5eab321de57
        9",
    "quantity": 1,
    "timestamp": 1736719466877
}
```

Update an Order → Update Order

1. Modify an existing order. Requires Bearer token authentication.
2. **Endpoint: PATCH** https://simple-books-api.glitch.me/orders/**{{orderId}}**
3. **Request Body (JSON)**

```
{
  "customerName" : "John"
}
```

4. **customerName (String)** - Optional (allows updating customer name).



Day6_BooksAPI / Update Order

PATCH https://simple-books-api.glitch.me/orders/ {{orderId}}

Variables in request

AccessToken	86c4f56c97c9afed210...
orderId	qVSl4kvY-elSQsEnaBW...

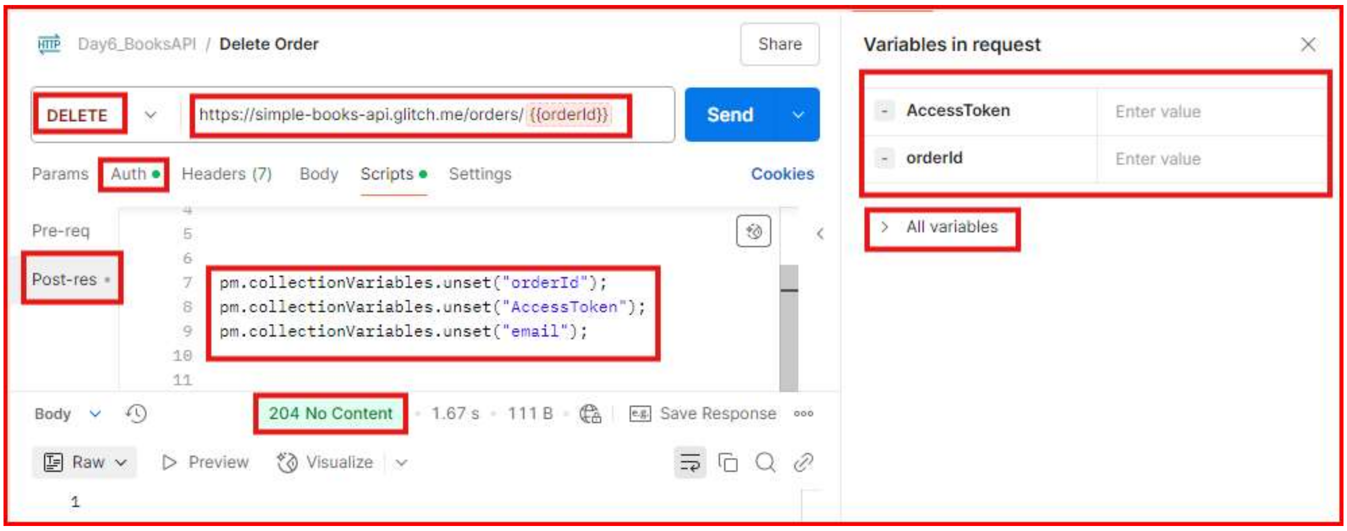
```
{
  "customerName": "Scott"
}
```

204 No Content · 590 ms · 111 B

5. **Execute GetOrders Request** to check whether updated or not

Delete an Order → Delete Order

1. Remove an existing order. Requires **Bearer token** authentication.
2. **Endpoint: DELETE** https://simple-books-api.glitch.me/orders/**{{orderId}}**



Day6_BooksAPI / Delete Order

DELETE https://simple-books-api.glitch.me/orders/ {{orderId}}

Scripts → Post-res

```
pm.collectionVariables.unset("orderId");
pm.collectionVariables.unset("AccessToken");
pm.collectionVariables.unset("email");
```

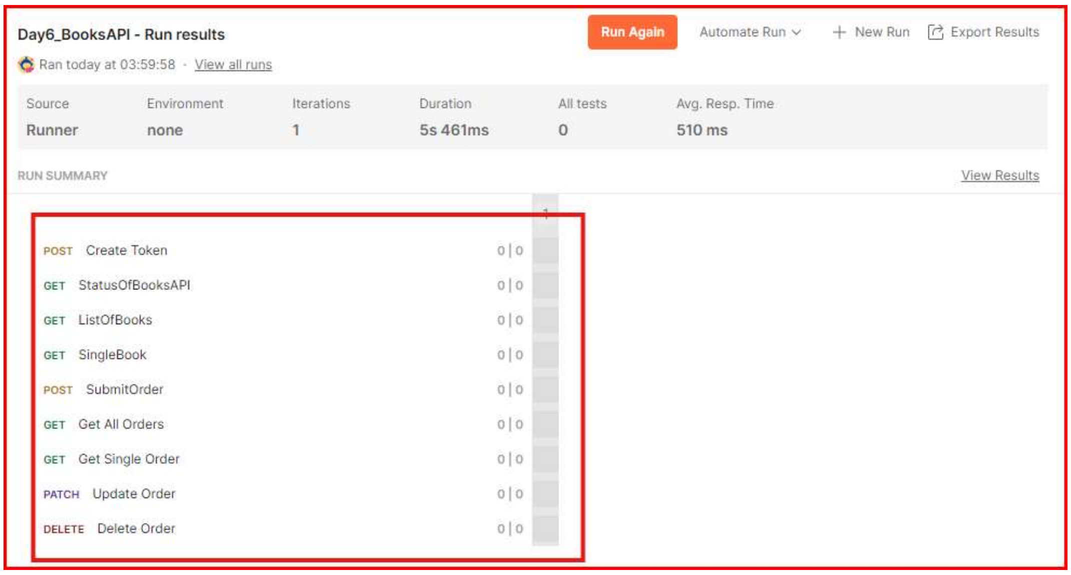
204 No Content · 1.67 s · 111 B

Variables in request

AccessToken	Enter value
orderId	Enter value

Observation

1. Execute all Requests in Collection → Run collection.
2. Upon executing all requests they are executed in the order in which they have created.



Day6_BooksAPI - Run results

Ran today at 03:59:58 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	5s 461ms	0	510 ms

RUN SUMMARY

POST	Create Token	0 | 0
GET	StatusOfBooksAPI	0 | 0
GET	ListOfBooks	0 | 0
GET	SingleBook	0 | 0
POST	SubmitOrder	0 | 0
GET	Get All Orders	0 | 0
GET	Get Single Order	0 | 0
PATCH	Update Order	0 | 0
DELETE	Delete Order	0 | 0

Note

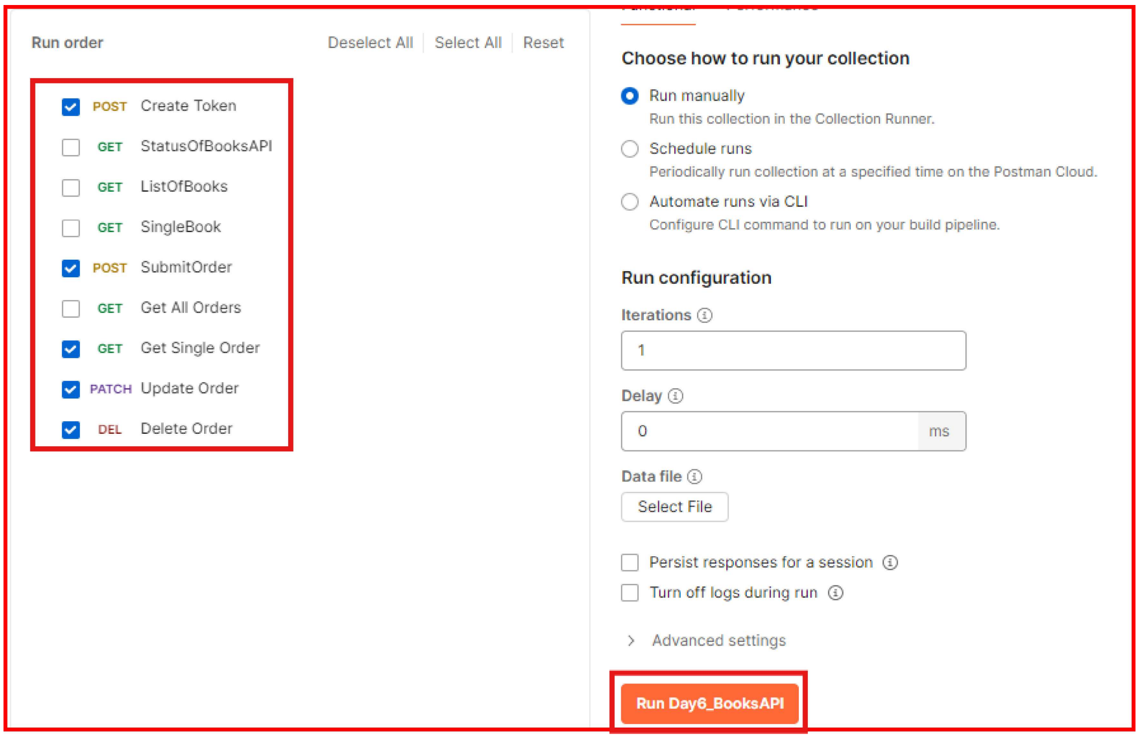
➔ In the Older versions of Postman Tool Requests in Collection are executed in Alphabetical order.

Execute Requests in our own Order

➔ Let suppose below is the order of execution required

- Create Token
- Submit Order
- Update Order
- Get Single Order
- Delete Order

➔ One way to execute is unselecting and selecting requests as below



Run order — Deselect All | Select All | Reset

- [✓] POST Create Token
- [ ] GET StatusOfBooksAPI
- [ ] GET ListOfBooks
- [ ] GET SingleBook
- [✓] POST SubmitOrder
- [ ] GET Get All Orders
- [✓] GET Get Single Order
- [✓] PATCH Update Order
- [✓] DEL Delete Order

Choose how to run your collection

- [✓ Run manually] Run this collection in the Collection Runner.
- Schedule runs — Periodically run collection at a specified time on the Postman Cloud.
- Automate runs via CLI — Configure CLI command to run on your build pipeline.

Run configuration

Iterations: 1

Delay: 0 ms

Data file: Select File

- [ ] Persist responses for a session
- [ ] Turn off logs during run

Advanced settings

Run Day6_BooksAPI

➔ Other way of executing is using **pm.execution.setNextRequest()` function** in Post-response to define which request to run next or to stop execution altogether.

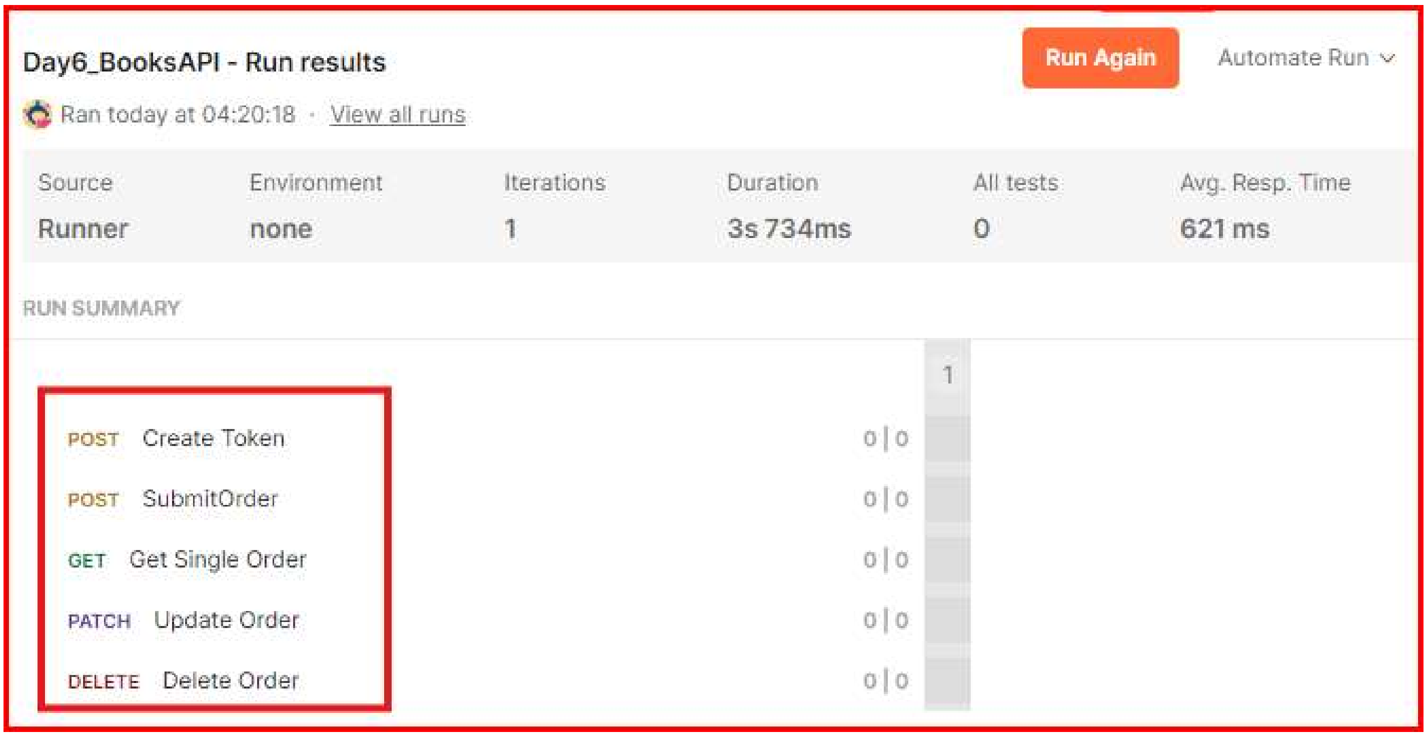
- Add in Post-response script Tab of **create Token** request
  - **pm.execution.setNextRequest("SubmitOrder");**
- Add in Post-response script Tab of **SubmitOrder** request
  - **pm.execution.setNextRequest("Get Single Order");**
- Add in Post-response script Tab of **Get Single Order** request
  - **pm.execution.setNextRequest("Update Order");**
- Add in Post-response script Tab of **Update Order** request
  - **pm.execution.setNextRequest("Delete Order");**

- Add in Post-response script Tab of **Delete Order** request
    - **pm.execution.setNextRequest(null);**

➔ **null** means simply no next request to execute

➔ If we have any request after Delete Order if we dont specify **pm.execution.setNextRequest(null);** it will be executed

➔ Now run all requests at collection level


Day6_BooksAPI - Run results

Ran today at 04:20:18 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	3s 734ms	0	621 ms

RUN SUMMARY

		1
POST Create Token	0 | 0	
POST SubmitOrder	0 | 0	
GET Get Single Order	0 | 0	
PATCH Update Order	0 | 0	
DELETE Delete Order	0 | 0	

Now we will create multiple orders by executing submit order with different sets of data, we need Data variables i.e, we will do Data Driven Testing.

Data Driven Testing

➔ Data-driven testing in Postman runs requests multiple times with different data from a CSV or JSON file to test various scenarios. Data-driven testing is done by using Data variables.

Execute Requests in Order for Data Driven Testing

➔ The Simple Books API provides endpoints to manage book orders using data-driven testing with JSON/CSV input for dynamic parameters.

➔ Let suppose below is the order of execution required for Data Driven Testing

- Submit Order
- Get Single Order
- Delete Order

➔ Create a Collection with name **Day6_BooksAPI-DataDriven** and create above requests in this collection.

➔ To run these requests, first generate a token using the "Create Token" request. Then, copy the token and use it manually in each request or set it at the collection level.

Step 1: Submit an Order

1. **Description:** Place a new book order.
2. **Endpoint: POST** https://simple-books-api.glitch.me/orders
3. **Authentication:** Bearer Token
4. **Request Body (JSON)** → **get the Data from external files**

```
{
    "bookId" : "{{BookID}}",
    "customerName" : "{{CustomerName}}"
}
```

5. **Post-response Validation**
    a. Check for status code 201.
    b. Extract and store the orderId from the response for subsequent steps.

```
pm.test("Status code is 201", () => {
    pm.response.to.have.status(201);
});
var jsonData = pm.response.json();
pm.collectionVariables.set("orderId", jsonData.orderId);
```

Step 2: Get a Single Order

1. **Description:** Retrieve the details of a specific order.
2. **Endpoint: GET** https://simple-books-api.glitch.me/orders/**{{orderId}}**
3. **Authentication:** Bearer Token
4. **Post-response Validation:**
    a. Check for status code 200.
    b. Verify the orderId in the response matches the stored variable.

```
pm.test("Status code is 200", () => {
    pm.response.to.have.status(200);
});
pm.test("Check OrderId present in the response body", () => {
    var jsonData = pm.response.json();
    pm.expect(jsonData.id).to.eql(pm.collectionVariables.get("orderId"));
});
```

Step 2: Remove an Order

5. **Description:** Delete an existing order.
6. **Endpoint: DELETE** https://simple-books-api.glitch.me/orders/**{{orderId}}**

7. **Authentication:** Bearer Token

8. **Post-response Validation:**

   a. Check for status code 204.

   b. Remove the orderId variable after successful deletion.

```
pm.test("Status code is 204", () => {
    pm.response.to.have.status(204);
});
pm.collectionVariables.unset("orderId");
```

Key Notes

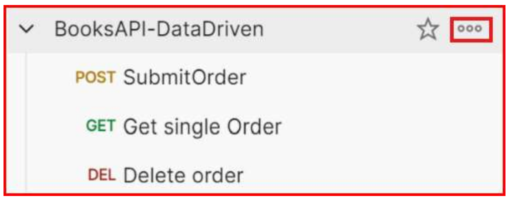
1. **Dynamic Data:** Use JSON or CSV to pass BookID and CustomerName for data-driven testing.

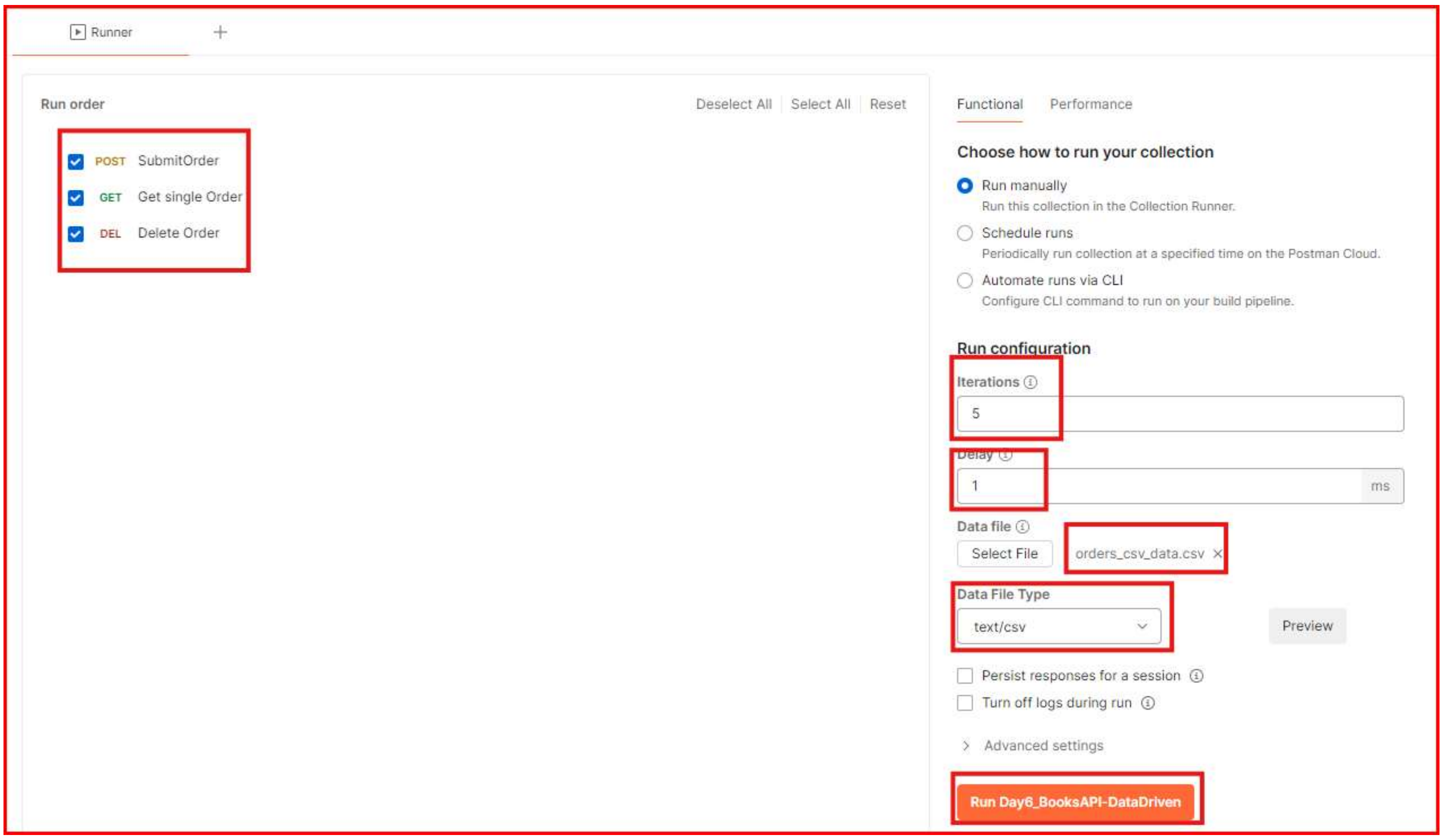
2. **Chained Testing:** Each step relies on data from the previous step. Proper variable management is critical.

3. **Postman Tests:** Automate response validation and variable handling using Postman scripts.

This approach ensures end-to-end validation of the API functionalities with reusable and dynamic test cases.

Run The collection in Postman





Choose how to run your collection

- Run manually — Run this collection in the Collection Runner. (selected)
- Schedule runs — Periodically run collection at a specified time on the Postman Cloud.
- Automate runs via CLI — Configure CLI command to run on your build pipeline.

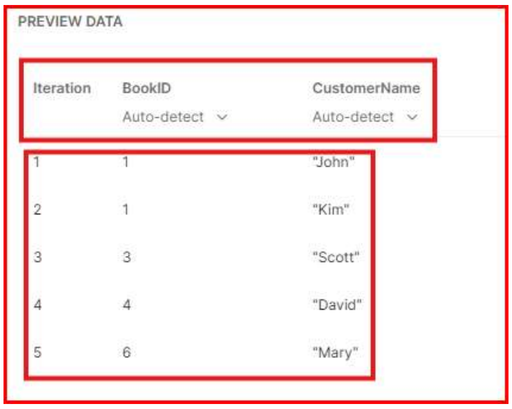
Run configuration

Iterations: 5

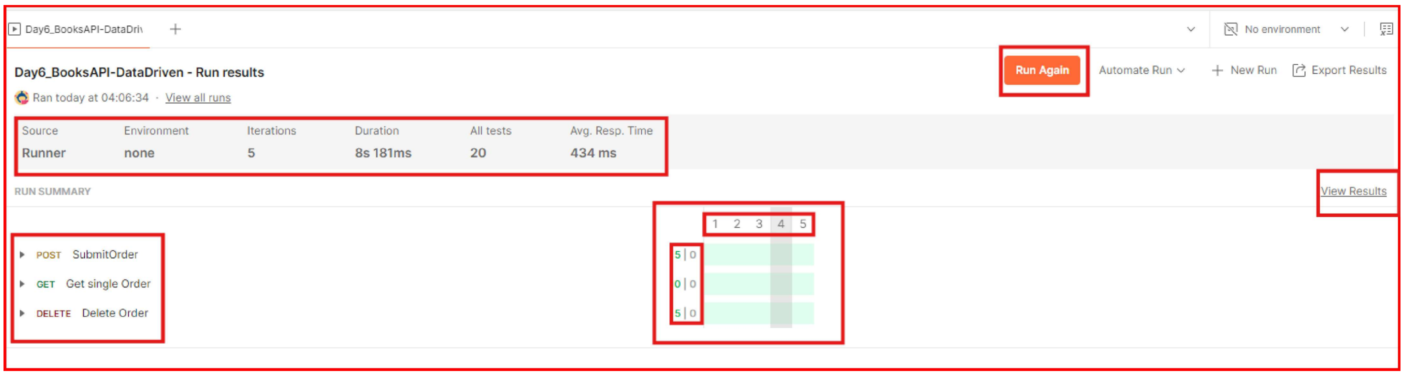
Delay: 1 ms

Data file: Select File — orders_csv_data.csv

Data File Type: text/csv


PREVIEW DATA

Iteration	BookID Auto-detect	CustomerName Auto-detect
1	1	"John"
2	1	"Kim"
3	3	"Scott"
4	4	"David"
5	6	"Mary"



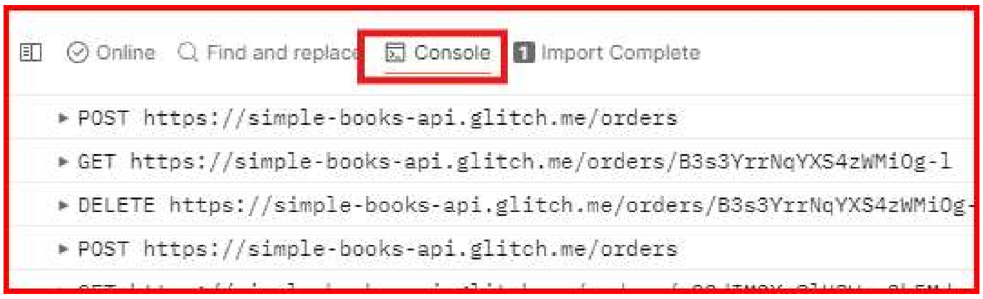
Day6_BooksAPI-DataDriven - Run results

Ran today at 04:06:34 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	5	8s 181ms	20	434 ms

RUN SUMMARY

		1 2 3 4 5
POST SubmitOrder	5 | 0	
GET Get single Order	0 | 0	
DELETE Delete Order	5 | 0	

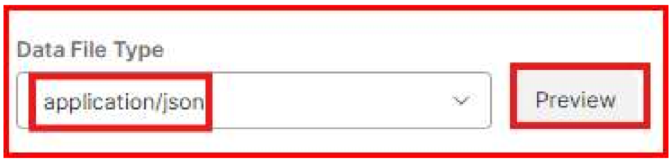


```
▸ POST https://simple-books-api.glitch.me/orders
▸ GET https://simple-books-api.glitch.me/orders/B3s3YrrNqYXS4zWMiOg-l
▸ DELETE https://simple-books-api.glitch.me/orders/B3s3YrrNqYXS4zWMiOg-
▸ POST https://simple-books-api.glitch.me/orders
```

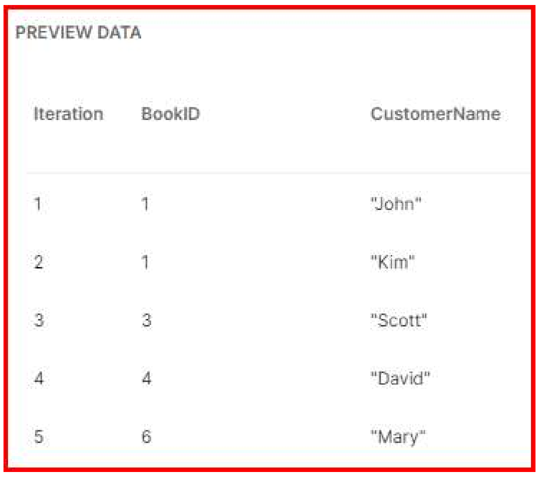
Note

➔ Column names and variable names in JSON/CSV must match

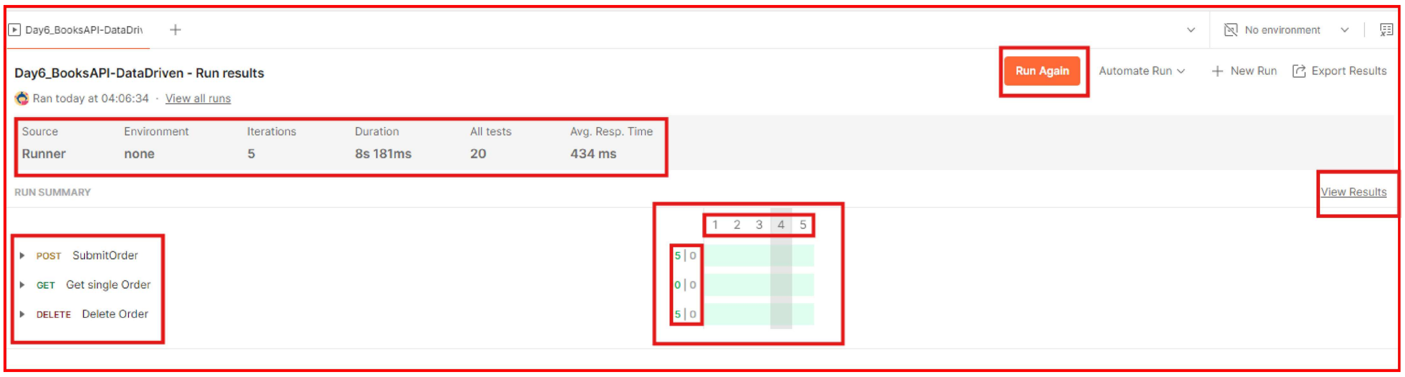
➔ Execute above collection using JSON data from JSON file. mime type is **application/json**


Data File Type

application/json


PREVIEW DATA

Iteration	BookID	CustomerName
1	1	"John"
2	1	"Kim"
3	3	"Scott"
4	4	"David"
5	6	"Mary"

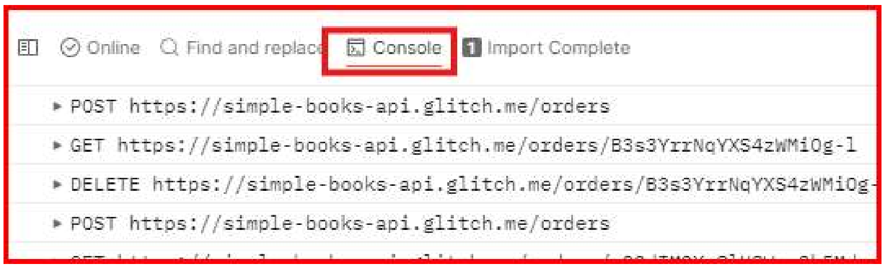


Day6_BooksAPI-DataDriven - Run results

Ran today at 04:06:34 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	5	8s 181ms	20	434 ms

POST SubmitOrder
GET Get single Order
DELETE Delete Order



```
▸ POST https://simple-books-api.glitch.me/orders
▸ GET https://simple-books-api.glitch.me/orders/B3s3YrrNqYXS4zWMiOg-l
▸ DELETE https://simple-books-api.glitch.me/orders/B3s3YrrNqYXS4zWMiOg-
▸ POST https://simple-books-api.glitch.me/orders
```

➔ For converting CSV to JSON and vice versa web bases tools are avilable

- https://data.page/json/csv
- https://csvjson.com/

Note

➔ If any request takes unpredictable time to complete we can add simple validation in post response Tab

```
const response = pm.response.json();
const status = response.status; // Assuming response has a 'status' field
if (status !== "processed") {
  // Retry same request
  console.log("Order not ready. Retrying...");
  postman.setNextRequest("Check Order Status");
} else {
  // Proceed to next request
  pm.execution.setNextRequest("Next Request Name");
}
```

➔ We can also add some delay as like below

- Create a new request named **"Wait 3 Seconds"**
  - **Get https://postman-echo.com/delay/3**
- In the Test tab of this "Wait" request
  - **pm.execution.setNextRequest("Get Single Order");**