**API** **Assignment**

### **1. Explain the following:**

#### **A. PUT and PATCH Methods:**

* **PUT Method**:
  + The **PUT** method is used to update or create a resource at a specific URL. It is idempotent, meaning that multiple identical requests will have the same effect as a single request.
  + **Example**: If you send a **PUT** request to update user data, you are generally expected to send the complete data of the resource.
  + **Use Case**: Updating a complete record or resource.

**Example PUT request**:

PUT /users/1

{

"name": "John",

"email": "john@example.com"

}

* **PATCH Method**:
  + The **PATCH** method is used to partially update a resource. Unlike **PUT**, which requires sending the entire updated resource, **PATCH** only sends the fields that need to be updated.
  + **Example**: You can update just the user's email address while leaving other details intact.
  + **Use Case**: When you need to update only a specific field or part of a resource.

**Example PATCH request**:

PATCH /users/1

{

"email": "john\_new@example.com"

}

#### **B. Headers and Cookies:**

* **Headers**:
  + Headers are metadata sent along with HTTP requests and responses. They contain information such as the content type, authorization tokens, and client details.
  + **Common Examples**:
    - Content-Type: Specifies the format of the request body (e.g., application/json).
    - Authorization: Contains credentials or tokens used for authentication.
    - User-Agent: Provides information about the client making the request.

**Example of Headers in a request**:

GET /api/users

Host: example.com

Content-Type: application/json

Authorization: Bearer <token>

* **Cookies**:
  + Cookies are small pieces of data sent from the server and stored on the client-side. They are used to retain information about the user between requests, such as session state or login credentials.
  + **Example**: Cookies can be used to keep track of user authentication and session information.

**Example of Cookies in a request**:

GET /dashboard

Cookie: session\_id=abc12345

#### **C. Endpoint and Base URL:**

* **Base URL**:
  + The base URL is the root address for an API that provides a common part of the URL for all API endpoints. It is the starting point for constructing all URLs in the API.
  + **Example**: https://api.example.com/v1/
* **Endpoint**:
  + An endpoint refers to a specific path on the API server, which, combined with the base URL, defines the location of the resource.
  + **Example**: For GET /users, the complete endpoint would be https://api.example.com/v1/users.

#### **D. Query Parameters and Path Parameters:**

* **Query Parameters**:
  + Query parameters are added to the URL after a ? symbol and are typically used to filter or sort data. They are key-value pairs.
  + **Example**: https://api.example.com/users?name=John&age=30
  + **Use Case**: Filtering or sorting data.
* **Path Parameters**:
  + Path parameters are part of the URL path and are used to identify a specific resource or item. They are embedded directly in the URL.
  + **Example**: https://api.example.com/users/123 (where 123 is a path parameter representing a specific user ID).
  + **Use Case**: Accessing a specific resource identified by a unique ID.

#### **E. What are Error Codes? Explain all Series of Error Codes:**

* **Error Codes**:  
  Error codes are numeric values used in HTTP responses to indicate the result of the request. They are divided into several categories, based on the range of the number.
* **Error Code Series**:
  + **1xx (Informational)**:
    - These codes indicate that the request has been received and the server is processing it.
    - **Example**:  
      100 Continue: The server has received the request and is waiting for the rest of the request to be sent.
  + **2xx (Successful)**:
    - These codes indicate that the request has been successfully processed.
    - **Examples**:  
      200 OK: The request was successful, and the server has returned the requested data.  
      201 Created: The request was successful, and a new resource was created.
  + **3xx (Redirection)**:
    - These codes indicate that further action is needed to complete the request, typically a redirect.
    - **Examples**:  
      301 Moved Permanently: The requested resource has been permanently moved to a new location.  
      302 Found: The requested resource has been temporarily moved.
  + **4xx (Client Error)**:
    - These codes indicate errors made by the client (request issues).
    - **Examples**:  
      400 Bad Request: The server could not understand the request due to invalid syntax.  
      401 Unauthorized: Authentication is required to access the resource.  
      404 Not Found: The requested resource could not be found.  
      403 Forbidden: The server understands the request but refuses to authorize it.
  + **5xx (Server Error)**:
    - These codes indicate server-side errors where the request could not be completed due to issues on the server.
    - **Examples**:  
      500 Internal Server Error: The server encountered an unexpected condition.  
      502 Bad Gateway: The server received an invalid response from the upstream server.  
      503 Service Unavailable: The server is currently unable to handle the request due to a temporary overload or maintenance.

### **2. Create a new collection in Postman named "Sample APIs" using** <https://reqres.in/>

1. Open **Postman**.
2. Click on **New** and choose **Collection**.
3. Name the collection "Sample APIs" and click **Create**.
4. **Add requests for each HTTP method**:
   * **GET Request**:
     + URL: https://reqres.in/api/users/2
     + Method: **GET**
   * **POST Request**:
     + URL: https://reqres.in/api/users
     + Method: **POST**
     + Body (raw, JSON):
     + {
     + "name": "John Doe",
     + "job": "Developer"
     + }
   * **PUT Request**:
     + URL: https://reqres.in/api/users/2
     + Method: **PUT**
     + Body (raw, JSON):
     + {
     + "name": "John Doe",
     + "job": "Senior Developer"
     + }
   * **PATCH Request**:
     + URL: https://reqres.in/api/users/2
     + Method: **PATCH**
     + Body (raw, JSON):
     + {
     + "job": "Lead Developer"
     + }
   * **DELETE Request**:
     + URL: https://reqres.in/api/users/2
     + Method: **DELETE**
5. Save the requests to the "Sample APIs" collection.

### **3. What is the difference between given(), when(), and then() in Rest Assured?**

* **given()**:
  + Used to define the preconditions for the request. It is where you set up things like headers, parameters, request body, or authentication.
  + **Example**: given().header("Authorization", "Bearer token").param("id", 123)
* **when()**:
  + Used to define the actual action to be performed in the request (e.g., GET, POST, PUT, DELETE). It is used to send the request.
  + **Example**: when().get("/users/1")
* **then()**:
  + Used to define the assertions for validating the response. You can check the status code, body, headers, or other response aspects.
  + **Example**: then().statusCode(200).body("name", equalTo("John"))

**Example Rest Assured code**:

given()

.header("Authorization", "Bearer token")

.param("id", 123)

when()

.get("/users/1")

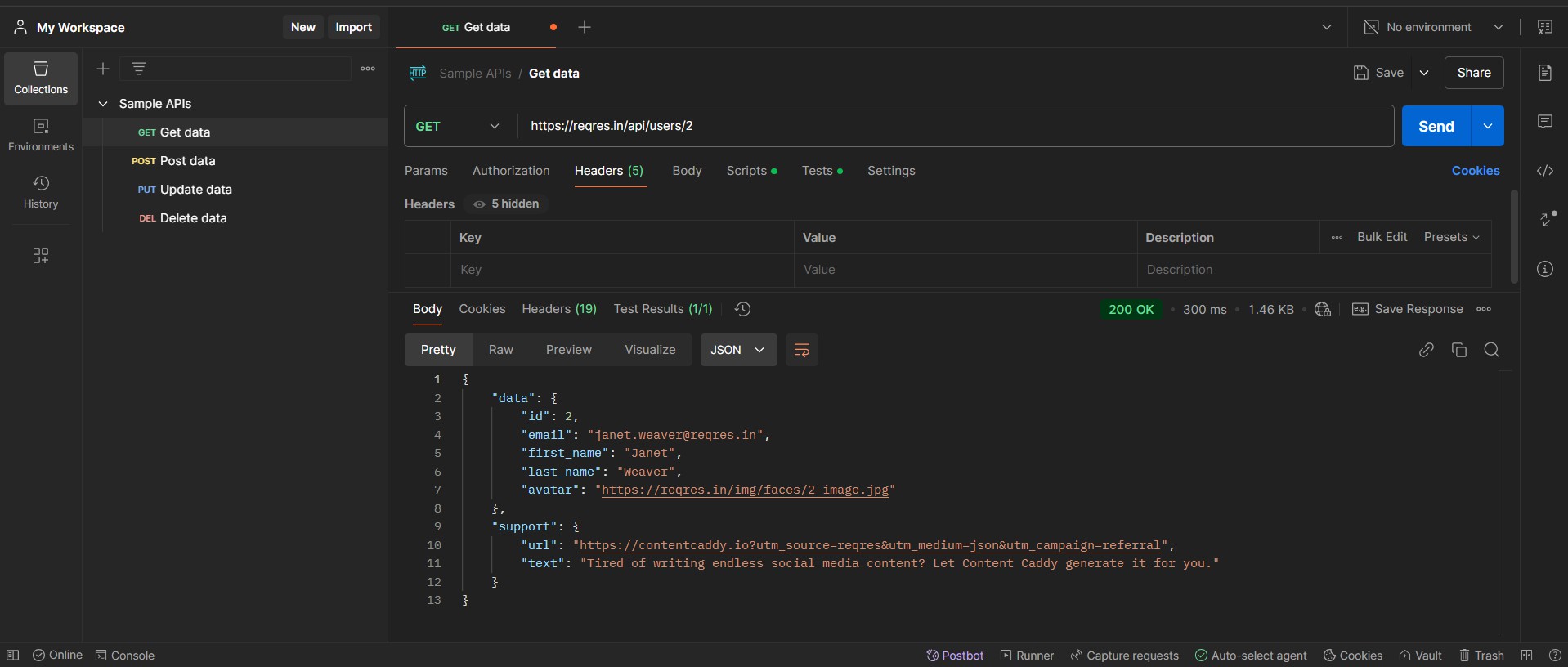
then()

.statusCode(200)

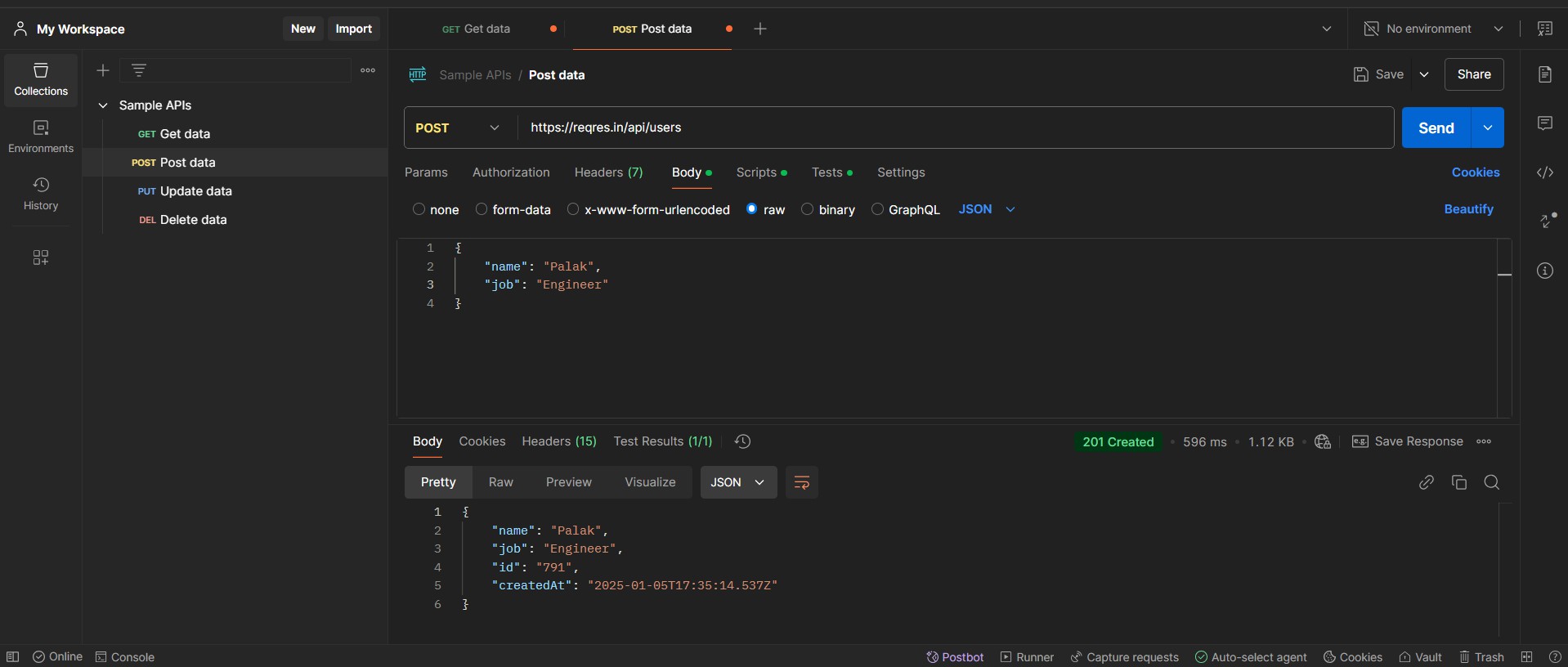
.body("name", equalTo("John"));

Q4. Create a new collection in Postman named "Sample APIs." Use https://reqres.in/ and implement GET, POST, PUT, PATCH, and DELETE operations.

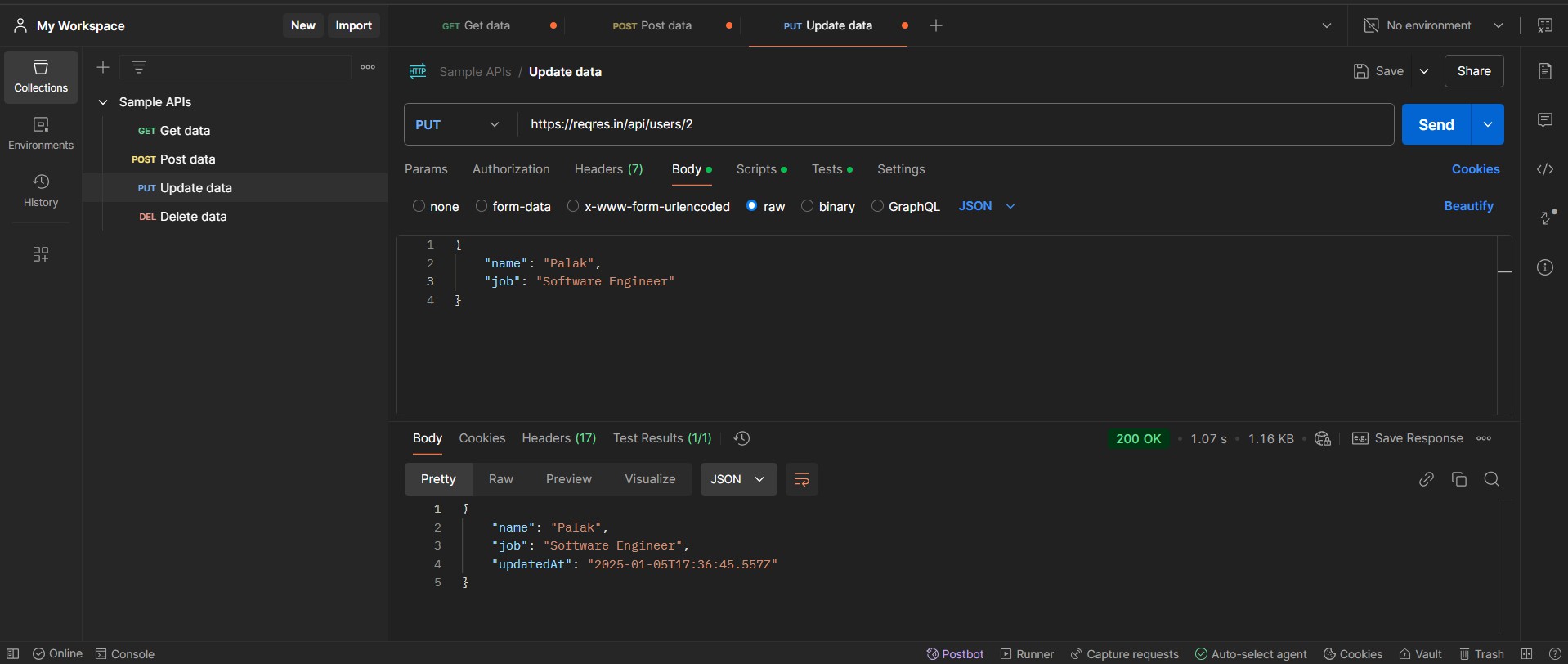
1. Use GET request to fetch the single data



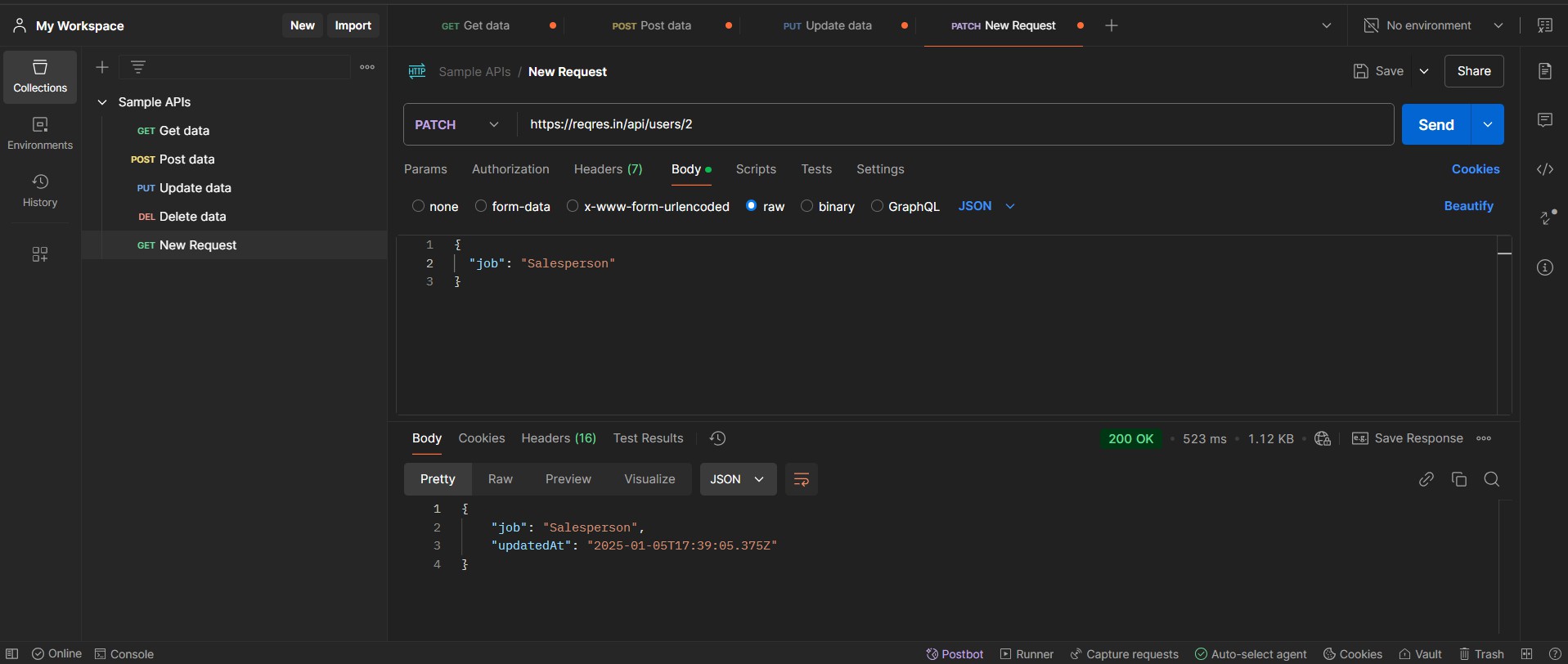
1. Use POST request to post the data



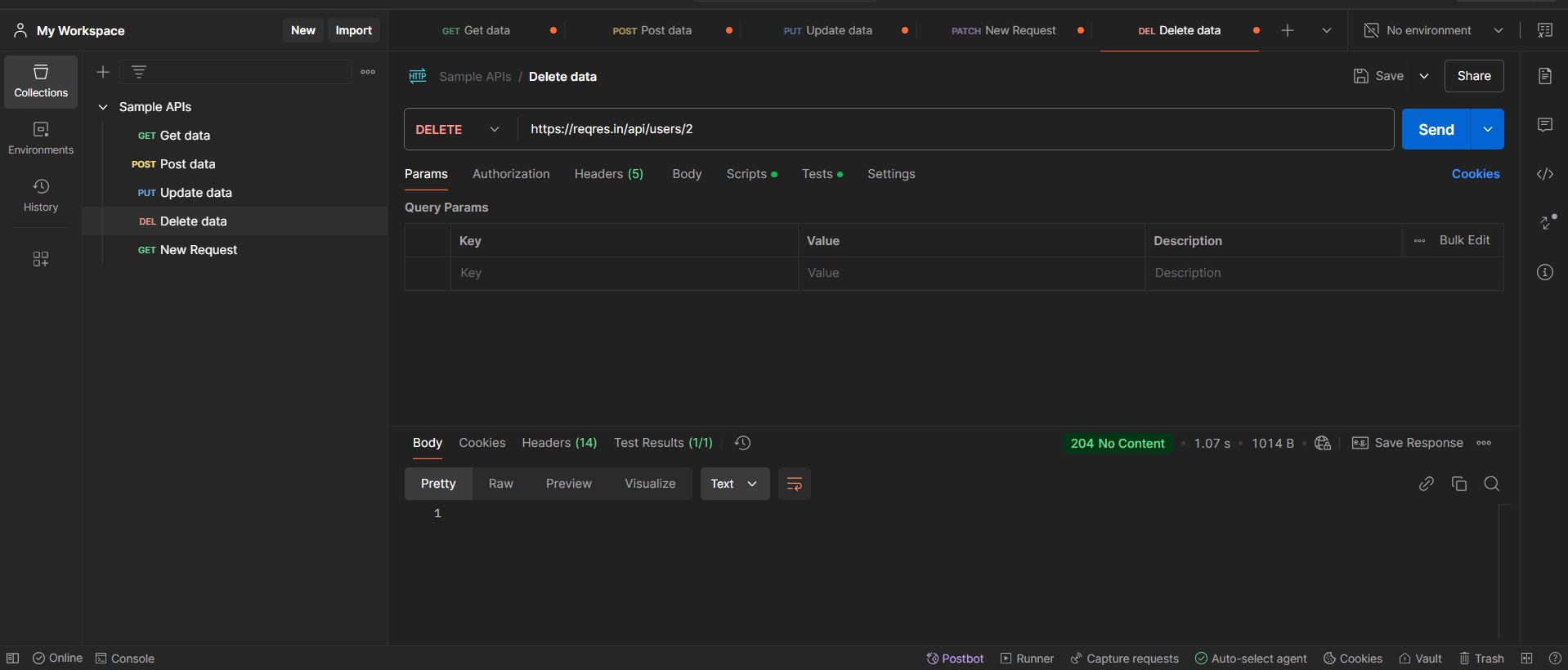
1. Use PUT request to update the existing data



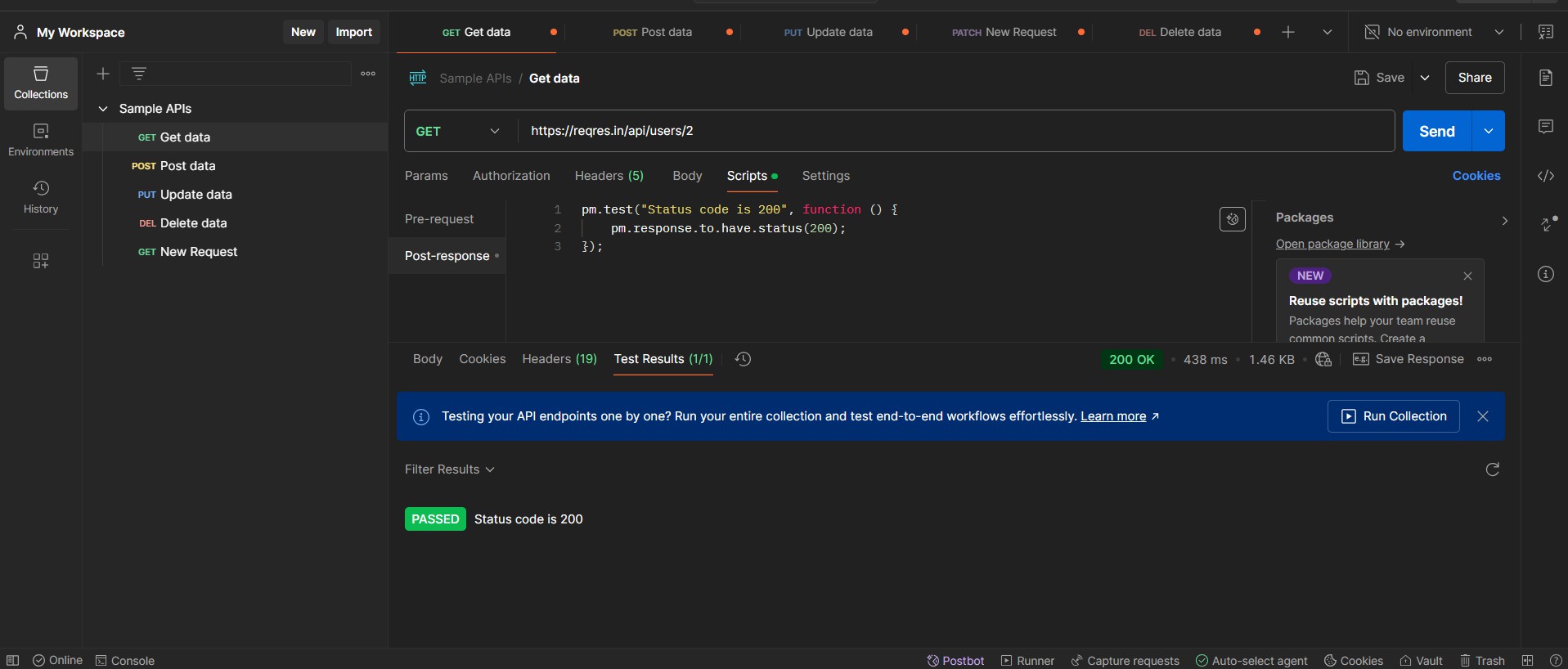
1. Use PATCH request to update the data partially



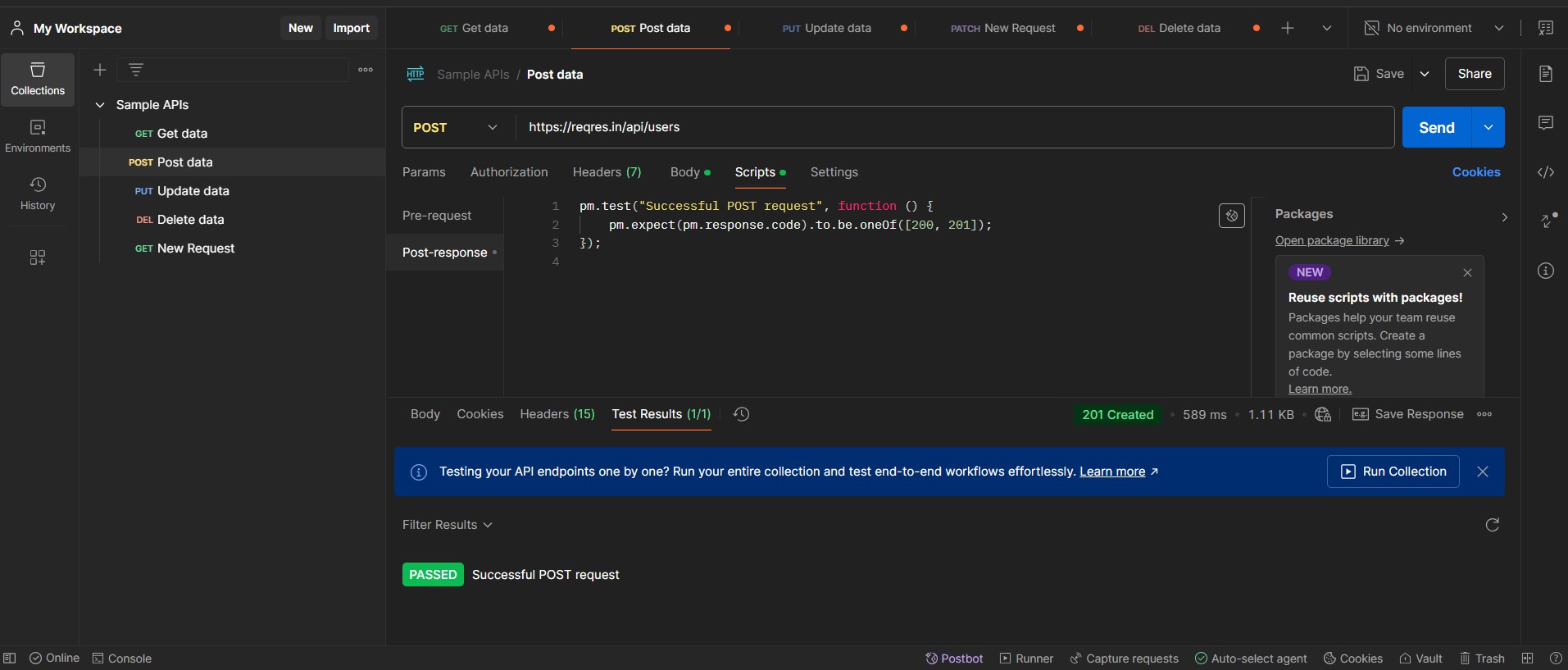
1. Use DELETE request to data



Q4 B) Assert for 200/201 response For 200 :



For 201:



C) Verify “first\_name” key value from GET response

