Use endpoint from https://jsonplaceholder.typicode.com/ for API test cases:

Q1. Create all possible positive negative test cases for following endpoints

GET /posts

```
pm.test("Response of Json with body", function(){
  pm.response.to.be.ok;
  pm.response.to.be.withBody;
  pm.response.to.be.json;
});
pm.test("response should be error free", function () {
  pm.response.to.not.be.error;
  pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
pm.test("Response time is less than 200ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(200);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("All keys included", function () {
  pm.expect(pm.response.text()).to.include("userID" & "id" & "title" & "body");
```

```
});
pm.test("100 IDs", function () {
  pm.expect(pm.response.json()[0].id).eqls(1);
  pm.expect(pm.response.json()[99].id).eqls(100);
});
pm.test("Id to be ordered", function () {
  pm.expect(pm.response.json().id).to.be.ordered;
});
   GET
                    https://jsonplaceholder.typicode.com/posts
                                                  Pre-request Script
  Params
            Authorization
                           Headers (6)
                                          Body
                                                                      Tests •
                                                                                Settin
      1
          pm.test("Response of Json with body", function(){
     2
              pm.response.to.be.ok;
     3
              pm.response.to.be.withBody;
              pm.response.to.be.json;
 Body Cookies Headers (25)
                               Test Results (8/8)
          Passed
                      Skipped
                                 Failed
   All
          Response of Json with body
          response should be error free
          Status code is 200
   PASS
          Response time is less than 200ms
          Content-Type is present
          All keys included
          100 IDs
          ld to be ordered
```

GET /posts/1

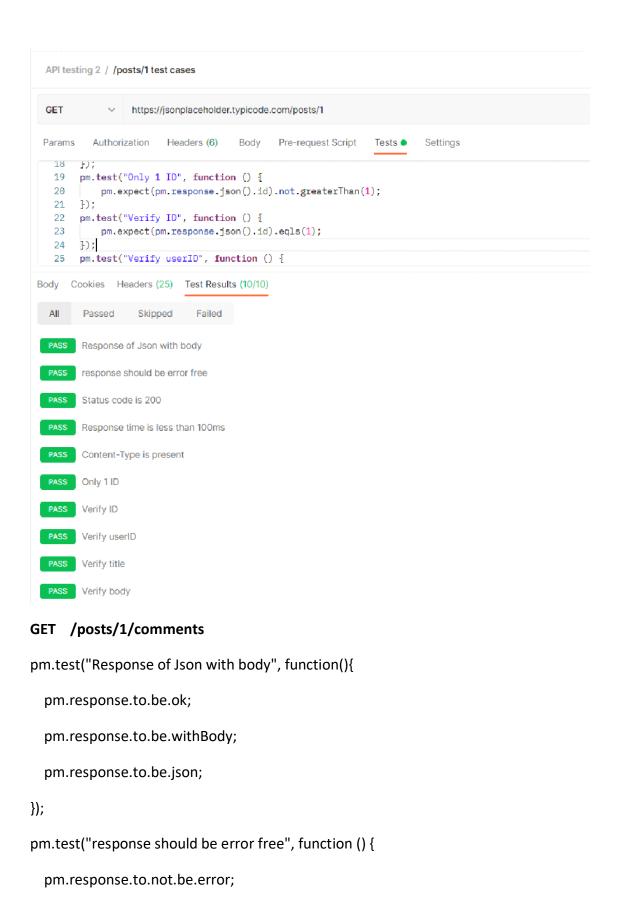
```
pm.test("Response of Json with body", function(){
  pm.response.to.be.ok;
  pm.response.to.be.withBody;
  pm.response.to.be.json;
});
pm.test("response should be error free", function () {
  pm.response.to.not.be.error;
  pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
pm.test("Response time is less than 100ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(100);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("Only 1 ID", function () {
  pm.expect(pm.response.json().id).not.greaterThan(1);
});
pm.test("Verify ID", function () {
```

```
pm.expect(pm.response.json().id).eqls(1);
});

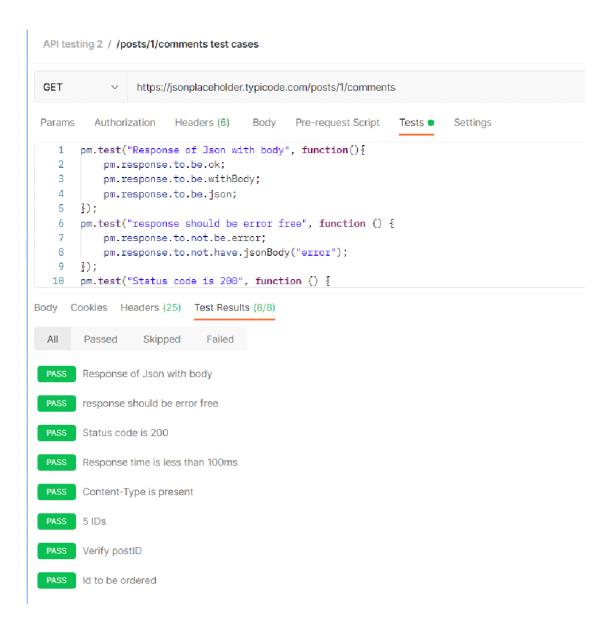
pm.test("Verify userID", function () {
    pm.expect(pm.response.json().userId).eqls(1);
});

pm.test("Verify title", function () {
    pm.expect(pm.response.json().title).eqls("sunt aut facere repellat provident occaecati excepturi optio reprehenderit");
});

pm.test("Verify body", function () {
    pm.expect(pm.response.json().body).eqls("quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto");
});
```

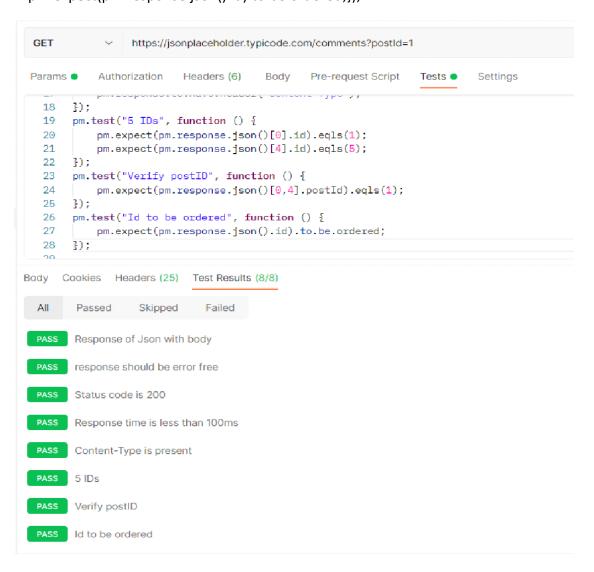


```
pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
pm.test("Response time is less than 100ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(100);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("5 IDs", function () {
  pm.expect(pm.response.json()[0].id).eqls(1);
  pm.expect(pm.response.json()[4].id).eqls(5);
});
pm.test("Verify postID", function () {
  pm.expect(pm.response.json()[0,4].postId).eqls(1);
});
pm.test("Id to be ordered", function () {
  pm.expect(pm.response.json().id).to.be.ordered;
});
```



```
pm.response.to.be.ok;
  pm.response.to.be.withBody;
  pm.response.to.be.json;
});
pm.test("response should be error free", function () {
  pm.response.to.not.be.error;
  pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
pm.test("Response time is less than 100ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(100);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("5 IDs", function () {
  pm.expect(pm.response.json()[0].id).eqls(1);
  pm.expect(pm.response.json()[4].id).eqls(5);
});
pm.test("Verify postID", function () {
  pm.expect(pm.response.json()[0,4].postId).eqls(1);
});
```

pm.test("Id to be ordered", function () {
 pm.expect(pm.response.json().id).to.be.ordered;});

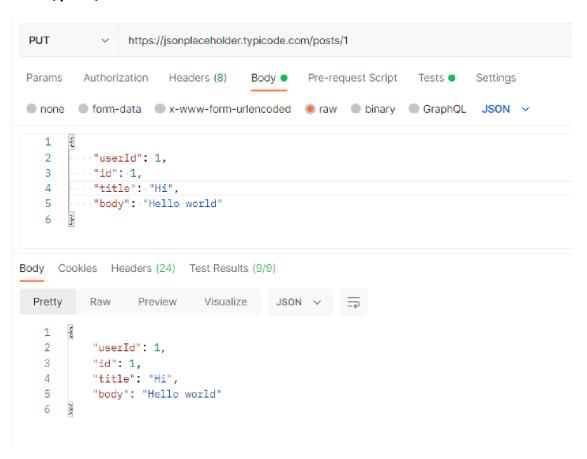


POST /posts

```
POST
                 https://jsonplaceholder.typicode.com/posts
                      Headers (8)
                                   Body •
                                           Pre-request Script
 Params
                                                                     JSON V
    1
    2
           "userId": 5,
           "id": 101,
    4
           "title": "API Testing ",
            "body": "Rest API testing using postman software"
    5
pm.test("response should be error free", function () {
  pm.response.to.not.be.error;
});
pm.test("POST request", function () {
  pm.expect(pm.response.code).to.be.oneOf([201, 202]);
});
pm.test("Response time is less than 200ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(100);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("Post status", () => {
 pm.response.to.have.status("Created");
});
pm.test("Verify data types", () => {
pm.expect(pm.response.json().userId).to.be.a("number");
pm.expect(pm.response.json().id).to.be.a("number");
```

```
pm.expect(pm.response.json().title).to.be.a("string");
pm.expect(pm.response.json().body).to.be.a("string");
});
pm.test("Non-empty fields", function () {
  pm.expect(pm.response.json().userID).not.null;
  pm.expect(pm.response.json().id).not.null;
  pm.expect(pm.response.json().title).not.null;
  pm.expect(pm.response.json().body).not.null;
});
   POST
                    https://jsonplaceholder.typicode.com/posts
            Authorization
                          Headers (8)
                                        Body •
                                                 Pre-request Script
                                                                             Settings
   Params
                                                                    Tests •
          pm.test("response should be error free", function () {
              pm.response.to.not.be.error;
      3
          pm.test("POST request", function () {
              pm.expect(pm.response.code).to.be.oneOf([201, 202]);
      5
      7
          pm.test("Response time is less than 200ms", function () {
              pm.expect(pm.response.responseTime).to.be.below(100);
      9
          });
          pm.test("Content-Type is present", function () {
  Body Cookies Headers (25) Test Results (7/7)
    All
           Passed
                     Skipped
         response should be error free
          POST request
    PASS Response time is less than 200ms
          Content-Type is present
    PASS
          Post status
    PASS
          Verify data types
         Non-empty fields
```

PUT /posts/1

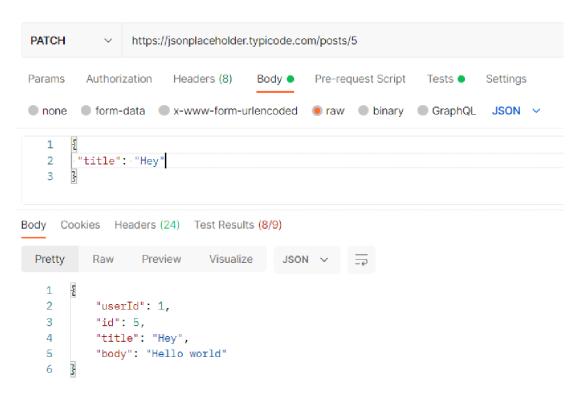


pm.test("Response of Json with body", function(){

```
pm.response.to.be.ok;
  pm.response.to.be.withBody;
  pm.response.to.be.json;
});
pm.test("response should be error free", function () {
  pm.response.to.not.be.error;
  pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
pm.test("Response time is less than 100ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(100);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("Verify userId", function () {
  pm.expect(pm.response.json().userId).eqls(1);
});
pm.test("Verify ID", function () {
  pm.expect(pm.response.json().id).eqls(1);
});
pm.test("Verify New title", function () {
```

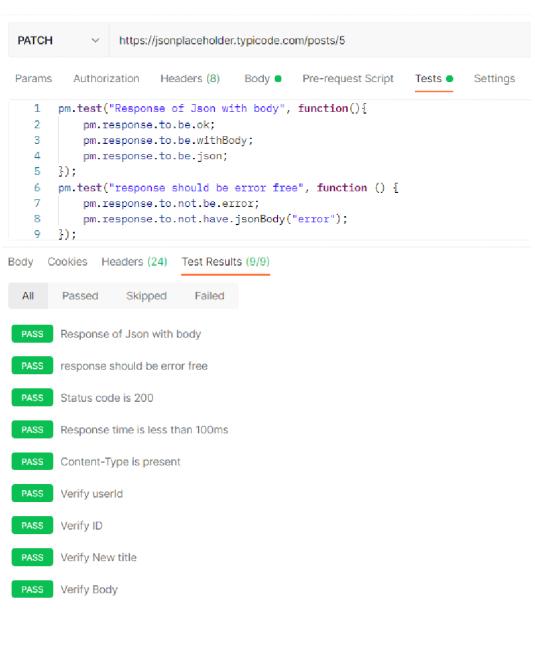
```
pm.expect(pm.response.json().title).eqls("Hi");
});
pm.test("Verify New Body", function () {
  pm.expect(pm.response.json().body).eqls("Hello world");
});
 PUT
                  https://jsonplaceholder.typicode.com/posts/1
           Authorization
                          Headers (8)
 Params
                                        Body •
                                                  Pre-request Script
                                                                      Tests •
                                                                                Settings
        pm.test("Response of Json with body", function(){
    2
            pm.response.to.be.ok;
    3
            pm.response.to.be.withBody;
    4
            pm.response.to.be.json;
    5
       });
        pm.test("response should be error free", function () {
Body Cookies Headers (24) Test Results (9/9)
  ΑII
         Passed
                               Failed
                    Skipped
        Response of Json with body
  PASS
        response should be error free
        Status code is 200
        Response time is less than 100ms
        Content-Type is present
  PASS
        Verify userId
        Verify ID
       Verify New title
       Verify New Body
```

PATCH /posts/1



```
pm.test("Response of Json with body", function(){
    pm.response.to.be.ok;
    pm.response.to.be.withBody;
    pm.response.to.be.json;
```

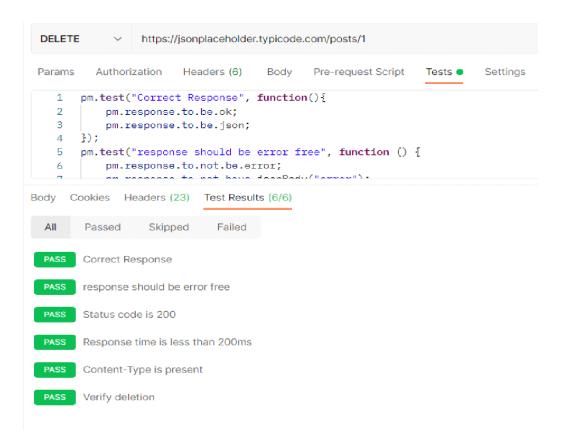
```
});
pm.test("response should be error free", function () {
  pm.response.to.not.be.error;
  pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
pm.test("Response time is less than 100ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(100);
});
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
pm.test("Verify userId", function () {
  pm.expect(pm.response.json().userId).eqls(1);
});
pm.test("Verify ID", function () {
  pm.expect(pm.response.json().id).eqls(5);
});
pm.test("Verify New title", function () {
  pm.expect(pm.response.json().title).eqls("Hey");
});
pm.test("Verify Body", function () {
  pm.expect(pm.response.json().body).eqls("Hello world");
});
```



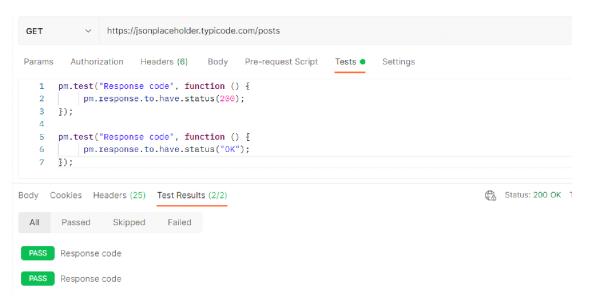
DELETE /posts/1

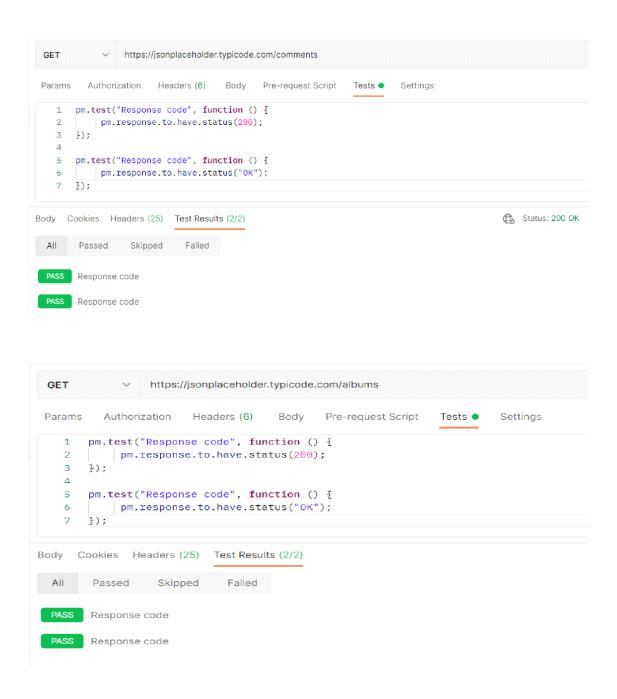
```
pm.test("Correct Response", function(){
    pm.response.to.be.ok;
    pm.response.to.be.json;
});
pm.test("response should be error free", function () {
```

```
pm.response.to.not.be.error;
pm.response.to.not.have.jsonBody("error");
});
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});
pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");
});
pm.test("Verify deletion", function () {
    pm.expect(pm.response.json().id).not.exist; });
```



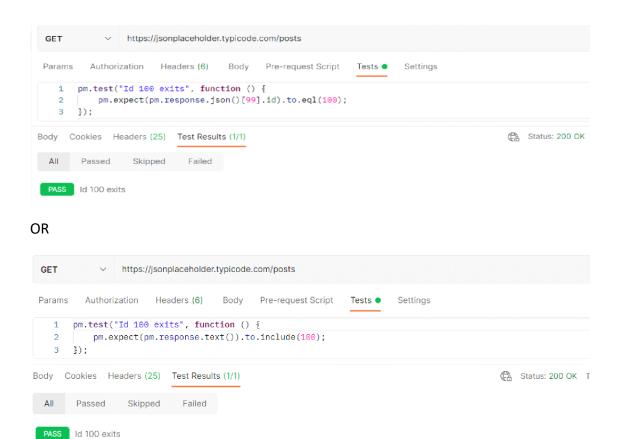
Q2. Assert response code and value from the response body.





Q3. Verify for id 100 exists in response data

https://jsonplaceholder.typicode.com/posts



Q4. Verify email id for from given response whose id is 5

https://jsonplaceholder.typicode.com/comments



Q5. Verify title from a given endpoint for tenth response

https://jsonplaceholder.typicode.com/albums

