

1. Create (Add) A Product (POST):

The screenshot shows the Postman interface for a POST request to `http://localhost:8082/api/products/addProduct`. The request body is a JSON object with the following fields:

```
1 {
2   "productId": "P21",
3   "productName": "Product 21",
4   "productPrice": 19.99,
5   "productQuantity": 100
6 }
```

The response body is displayed in the 'Body' tab, showing the same JSON object in a pretty-printed format:

```
1 {
2   "productId": "P21",
3   "productName": "Product 21",
4   "productPrice": 19.99,
5   "productQuantity": 100
6 }
```

Result Grid				
Filter Rows:				
	product_id	product_name	product_price	product_quantity
▶	P01	Product 1	19.99	100
	P02	Product 2	29.99	50
	P03	Product 3	9.99	200
	P04	Product 4	14.99	75
	P05	Product 5	39.99	150
	P06	Product 6	24.99	120
	P07	Product 7	49.99	90
	P08	Product 8	34.99	80
	P09	Product 9	0	60
	P10	Product 10	19.99	100
	P11	Product 11	29.99	50
	P12	Product 12	0	80
	P13	Product 13	49.99	90
	P14	Product 14	19.95	55
	P15	Product 15	24.99	75
	P16	Product 16	14.99	45
	P17	Product 17	39.99	110
	P18	Product 18	19.99	0
	P19	Product 19	0	70
	P20	Product 20	24.99	80
	P21	Product 21	19.99	100
*	NULL	NULL	NULL	NULL

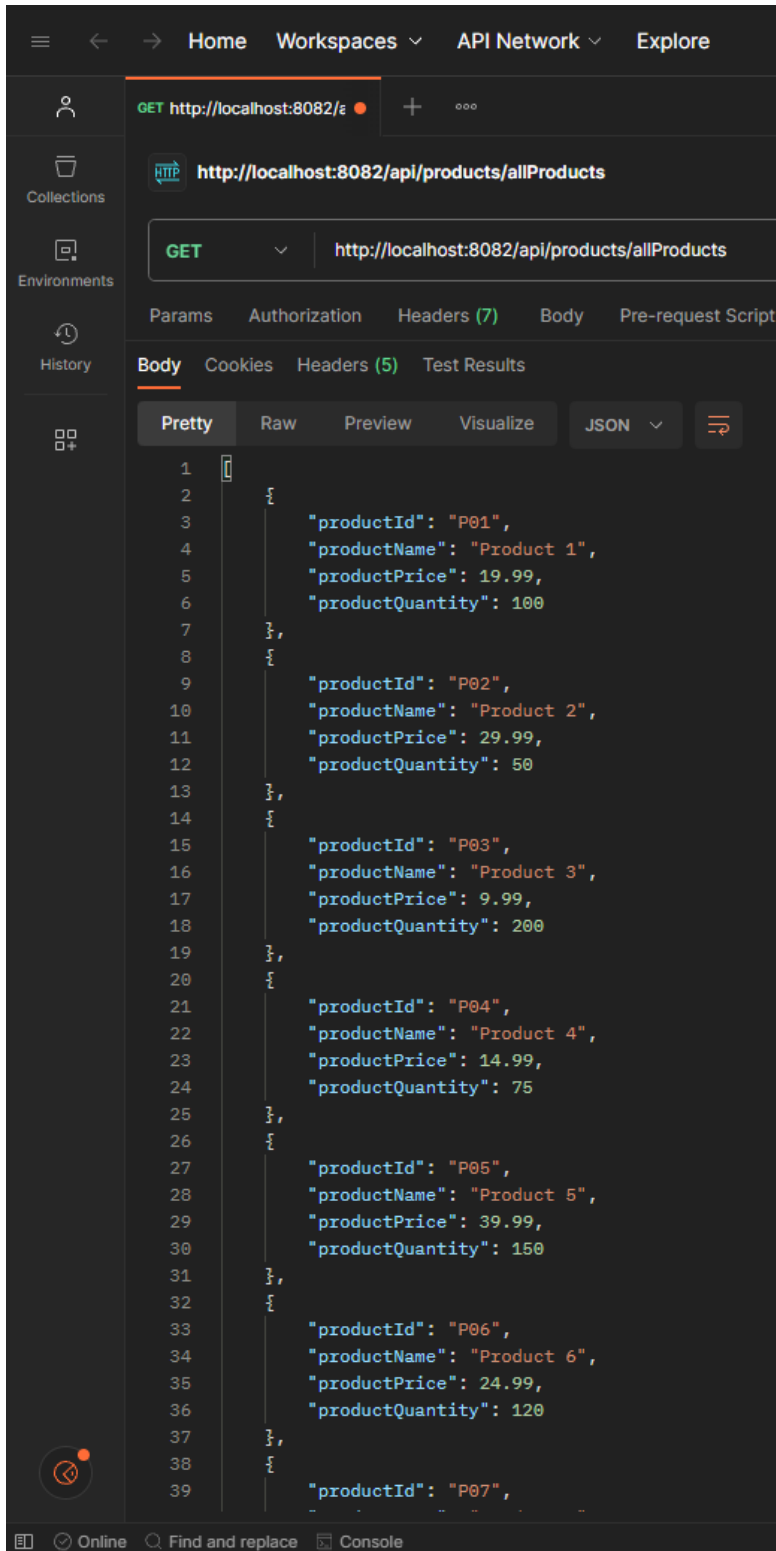
2. Retrieve a Product By productId (GET):

The screenshot shows the Postman interface with a GET request configured to `http://localhost:8082/api/products/P11`. The request is in the 'Params' tab, showing 'Query Params' with two entries, both labeled 'Key'. The 'Body' tab is selected, displaying a JSON response in 'Pretty' format:

```
1  {
2    "productId": "P11",
3    "productName": "Product 11",
4    "productPrice": 29.99,
5    "productQuantity": 50
6  }
```

The interface includes a sidebar with 'Collections', 'Environments', and 'History'. The top navigation bar shows 'Home', 'Workspaces', 'API Network', and 'Explore'.

3. Retrieve All Products (GET):



The screenshot shows the Postman interface with a GET request to `http://localhost:8082/api/products/allProducts`. The response is a JSON array of 7 product objects, displayed in the 'Body' tab with the 'Pretty' format. The products are:

- Product 1: productId "P01", productName "Product 1", productPrice 19.99, productQuantity 100
- Product 2: productId "P02", productName "Product 2", productPrice 29.99, productQuantity 50
- Product 3: productId "P03", productName "Product 3", productPrice 9.99, productQuantity 200
- Product 4: productId "P04", productName "Product 4", productPrice 14.99, productQuantity 75
- Product 5: productId "P05", productName "Product 5", productPrice 39.99, productQuantity 150
- Product 6: productId "P06", productName "Product 6", productPrice 24.99, productQuantity 120
- Product 7: productId "P07", productName "Product 7", productPrice 49.99, productQuantity 80

```
1 {
2   {
3     "productId": "P01",
4     "productName": "Product 1",
5     "productPrice": 19.99,
6     "productQuantity": 100
7   },
8   {
9     "productId": "P02",
10    "productName": "Product 2",
11    "productPrice": 29.99,
12    "productQuantity": 50
13  },
14  {
15    "productId": "P03",
16    "productName": "Product 3",
17    "productPrice": 9.99,
18    "productQuantity": 200
19  },
20  {
21    "productId": "P04",
22    "productName": "Product 4",
23    "productPrice": 14.99,
24    "productQuantity": 75
25  },
26  {
27    "productId": "P05",
28    "productName": "Product 5",
29    "productPrice": 39.99,
30    "productQuantity": 150
31  },
32  {
33    "productId": "P06",
34    "productName": "Product 6",
35    "productPrice": 24.99,
36    "productQuantity": 120
37  },
38  {
39    "productId": "P07",
40    "productName": "Product 7",
41    "productPrice": 49.99,
42    "productQuantity": 80
43  }
44 }
```

4. Update Product Name (PATCH):

The screenshot displays an API client interface with a sidebar on the left containing icons for Home, Collections, Environments, History, and a grid icon. The main panel shows a PATCH request to `http://localhost:8082/api/products/updateProductName/P01`. The request is configured with the following details:

- Method:** PATCH
- URL:** `http://localhost:8082/api/products/updateProductName/P01`
- Body Type:** form-data (selected)
- Body Content:**

Key
<input checked="" type="checkbox"/> newProductName
Key

The response is displayed in the 'Body' tab, showing a JSON object in 'Pretty' format:

```
1 {
2   "productId": "P01",
3   "productName": "Updated Product 1",
4   "productPrice": 19.99,
5   "productQuantity": 100
6 }
```

5. Update Product Price (PATCH):

The screenshot shows the Postman interface for a PATCH request. The URL is `http://localhost:8082/api/products/updateProductPrice/P01`. The request method is PATCH. The body is set to form-data. The body contains a single key-value pair: `newProductPrice` with a value of `29.99`. The response is displayed in the bottom panel, showing a JSON object with the following fields: `productId`, `productName`, `productPrice`, and `productQuantity`.

Home Workspaces More

PATCH `http://localhost:8082/api/products/updateProductPrice/P01`

HTTP `http://localhost:8082/api/products/updateProductPrice/P01`

PATCH `http://localhost:8082/api/products/updateProductPrice/P01`

Params Authorization Headers (9) Body Pre-request Script Tests

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

	Key	Value
<input checked="" type="checkbox"/>	<code>newProductPrice</code>	<code>29.99</code>
	Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "productId": "P01",
3   "productName": "Updated Product 1",
4   "productPrice": 29.99,
5   "productQuantity": 100
6 }
```

6. Update Product Quantity (PATCH):

The screenshot shows the Postman interface for a PATCH request. The URL is `http://localhost:8082/api/products/updateProductQuantity/P01`. The request is configured with the `form-data` body type. A single form field is present with the key `newProductQuantity` and the value `150`. The response is displayed in the bottom panel, showing a JSON object with product details.

Request Configuration:

- Method: PATCH
- URL: `http://localhost:8082/api/products/updateProductQuantity/P01`
- Body Type: form-data
- Body Fields:

Key	Value
<input checked="" type="checkbox"/> newProductQuantity	150

Response:

```
1 {
2   "productId": "P01",
3   "productName": "Updated Product 1",
4   "productPrice": 29.99,
5   "productQuantity": 150
6 }
```

7. Delete A Product By productId (DELETE):

The screenshot shows the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'More'. The left sidebar contains icons for 'Collections', 'Environments', 'History', and a 'Grid' icon. The main area displays a DELETE request configuration for the URL `http://localhost:8082/api/products/deleteProduct/P01`. The 'Params' tab is selected, showing a 'Query Params' table with two columns: 'Key' and 'Value'. Below the table, there are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Body' tab is selected, showing a 'Pretty' view of the request body. The 'Text' tab is also visible.

DEL `http://localhost:8082/a`

HTTP `http://localhost:8082/api/products/deleteProduct/P01`

DELETE `http://localhost:8082/api/products/deleteProduct/P01`

Params Authorization Headers (7) Body Pre-request Script

Query Params

Key	Value
Key	Value

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text

1

product_id	product_name	product_price	product_quantity
P02	Product 2	29.99	50
P03	Product 3	9.99	200
P04	Product 4	14.99	75
P05	Product 5	39.99	150
P06	Product 6	24.99	120
P07	Product 7	49.99	90
P08	Product 8	34.99	80
P09	Product 9	0	60
P10	Product 10	19.99	100
P11	Product 11	29.99	50
P12	Product 12	0	80
P13	Product 13	49.99	90
P14	Product 14	19.95	55
P15	Product 15	24.99	75
P16	Product 16	14.99	45
P17	Product 17	39.99	110
P18	Product 18	19.99	0
P19	Product 19	0	70
P20	Product 20	24.99	80
P21	Product 21	19.99	100

Ques 1. Find All Products With Price Greater Than Specified Value

The screenshot shows the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', and 'Explore'. The left sidebar contains 'Collections', 'Environments', 'History', and a 'New' button. The main area displays a GET request to `http://localhost:8082/api/products/findAllProductsWithPriceGreaterThan/30.0`. The 'Body' tab is selected, showing a JSON array of product objects. The 'Headers' tab shows 6 headers. The 'Body' tab is currently selected, showing a JSON array of product objects.

Request: GET `http://localhost:8082/api/products/findAllProductsWithPriceGreaterThan/30.0`

Body (JSON):

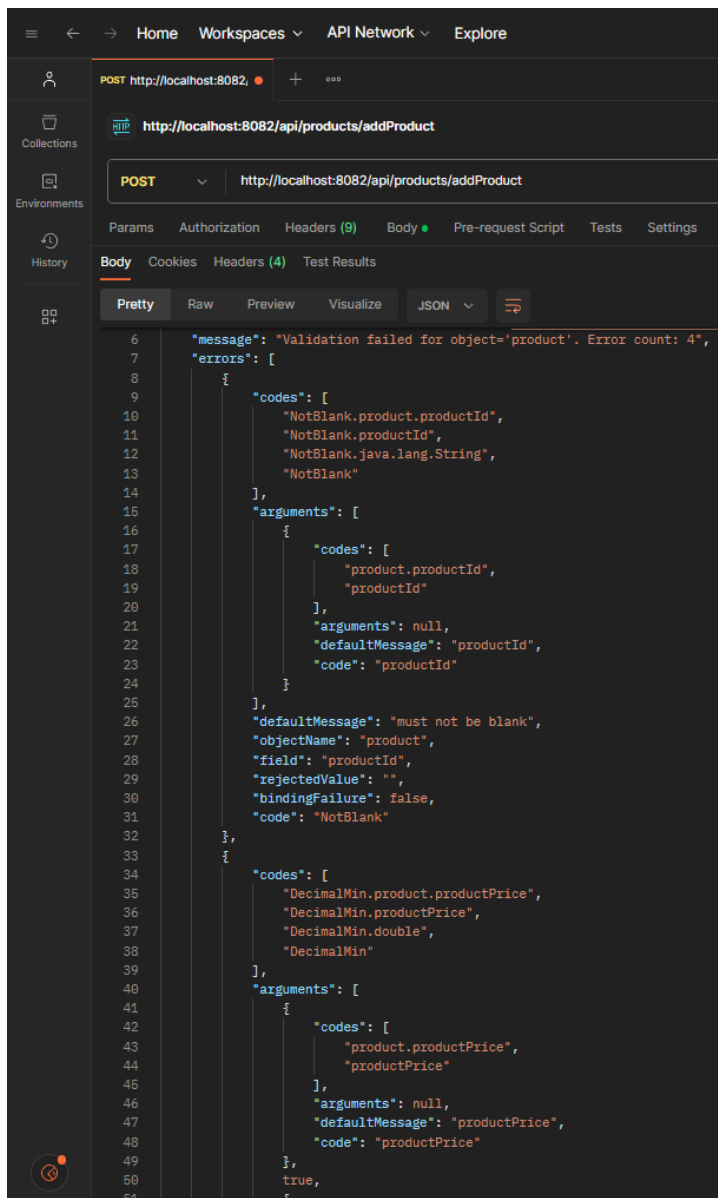
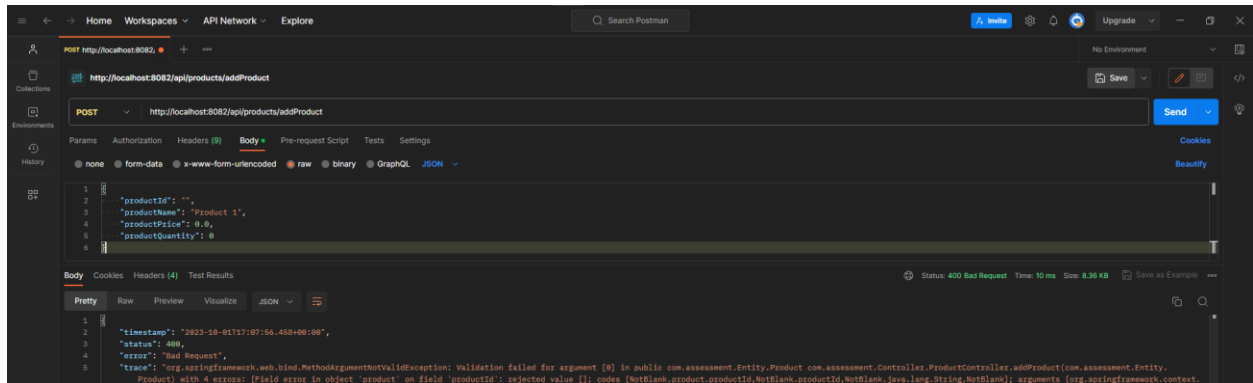
```
[
  {
    "productId": "P05",
    "productName": "Product 5",
    "productPrice": 39.99,
    "productQuantity": 150
  },
  {
    "productId": "P07",
    "productName": "Product 7",
    "productPrice": 49.99,
    "productQuantity": 90
  },
  {
    "productId": "P08",
    "productName": "Product 8",
    "productPrice": 34.99,
    "productQuantity": 80
  },
  {
    "productId": "P13",
    "productName": "Product 13",
    "productPrice": 49.99,
    "productQuantity": 90
  },
  {
    "productId": "P17",
    "productName": "Product 17",
    "productPrice": 39.99,
    "productQuantity": 110
  }
]
```


Ques 2. Find All Products With Price In Between Specified Price Range Values

The screenshot displays the Postman application interface. At the top, the navigation bar includes 'Home', 'Workspaces', 'API Network', and 'Explore'. The left sidebar contains icons for 'Collections', 'Environments', 'History', and a workspace icon. The main area shows an HTTP GET request to `http://localhost:8082/api/products/findAllProductsWithPriceInBetweenRange?minPrice=10.0&maxPrice=20.0`. The 'Body' tab is selected, showing a JSON array of product objects. The response is formatted in 'Pretty' view, with line numbers 1 through 38 on the left. The JSON data represents a list of products with their IDs, names, prices, and quantities.

```
1  {
2    "productId": "P04",
3    "productName": "Product 4",
4    "productPrice": 14.99,
5    "productQuantity": 75
6  },
7  {
8    "productId": "P10",
9    "productName": "Product 10",
10   "productPrice": 19.99,
11   "productQuantity": 100
12  },
13  {
14    "productId": "P14",
15    "productName": "Product 14",
16    "productPrice": 19.95,
17    "productQuantity": 55
18  },
19  {
20    "productId": "P16",
21    "productName": "Product 16",
22    "productPrice": 14.99,
23    "productQuantity": 45
24  },
25  {
26    "productId": "P18",
27    "productName": "Product 18",
28    "productPrice": 19.99,
29    "productQuantity": 0
30  },
31  {
32    "productId": "P21",
33    "productName": "Product 21",
34    "productPrice": 19.99,
35    "productQuantity": 100
36  },
37  }
38 }
```

Validations On ProductId Length, Product Price & Product Quantity



Home Workspaces API Network Explore

POST http://localhost:8082, + ...

http://localhost:8082/api/products/addProduct

POST http://localhost:8082/api/products/addProduct

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
50     true,
51     {
52       "arguments": null,
53       "defaultMessage": "0.1",
54       "codes": [
55         "0.1"
56       ]
57     }
58   ],
59   "defaultMessage": "Product Price Cannot Be Negative Or 0",
60   "objectName": "product",
61   "field": "productPrice",
62   "rejectedValue": 0.0,
63   "bindingFailure": false,
64   "code": "DecimalMin"
65 },
66 {
67   "codes": [
68     "Min.product.productQuantity",
69     "Min.productQuantity",
70     "Min.int",
71     "Min"
72   ],
73   "arguments": [
74     {
75       "codes": [
76         "product.productQuantity",
77         "productQuantity"
78       ],
79       "arguments": null,
80       "defaultMessage": "productQuantity",
81       "code": "productQuantity"
82     },
83     1
84   ],
85   "defaultMessage": "must be greater than or equal to 1",
86   "objectName": "product",
87   "field": "productQuantity",
88   "rejectedValue": 0,
89   "bindingFailure": false,
90   "code": "Min"
91 },
92 {
93   "codes": [
94     "Size.product.productId",
95     "Size.productId",
96     "Size.java.lang.String",
97     "Size"
98   ],
99   "arguments": [
100     {
101       "codes": [
102         "product.productId",
103         "productId"
104       ],
105       "arguments": null,
106       "defaultMessage": "productId",
107       "code": "productId"
108     },
109     9,
110     1
111   ],
112   "defaultMessage": "size must be between 1 and 9",
113   "objectName": "product",
114   "field": "productId",
115   "rejectedValue": "",
116   "bindingFailure": false,
117   "code": "Size"
118 },
119 ],
120 "path": "/api/products/addProduct"
121 }
```

Online Find and replace Console