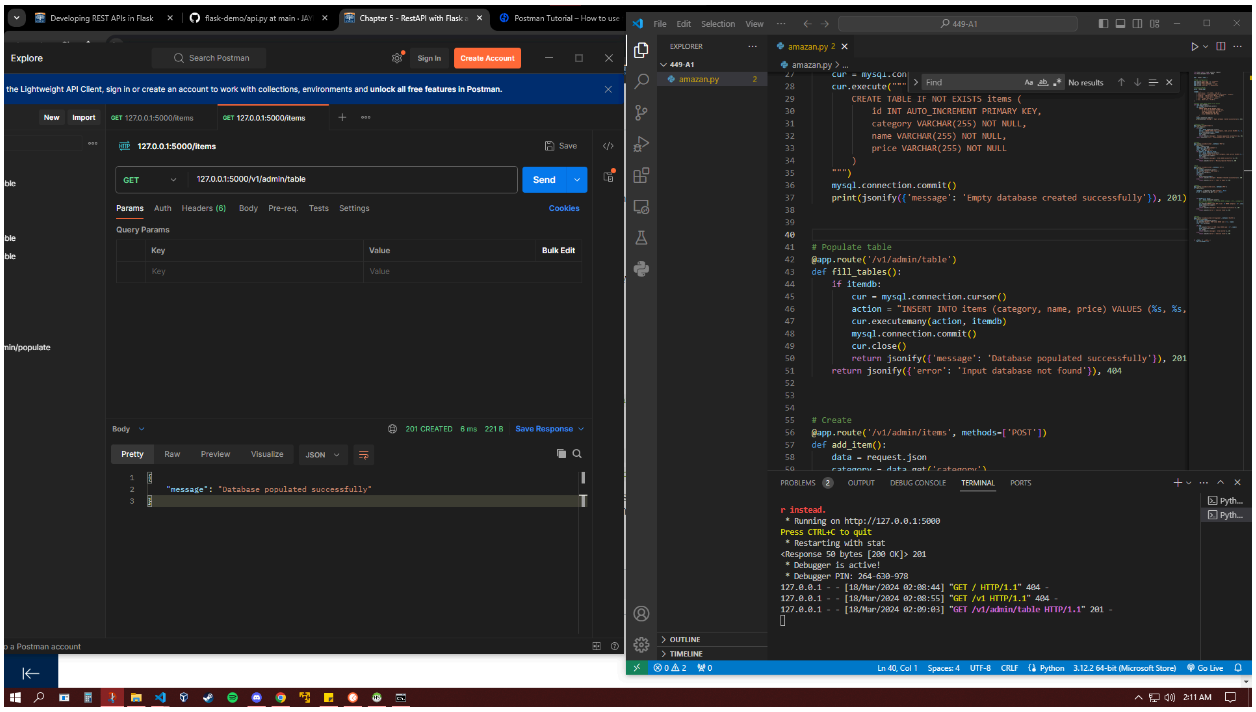
Mike Thai

Report

Topic: E-commerce



GET 127.0.0.1:5000/v1/admin/table

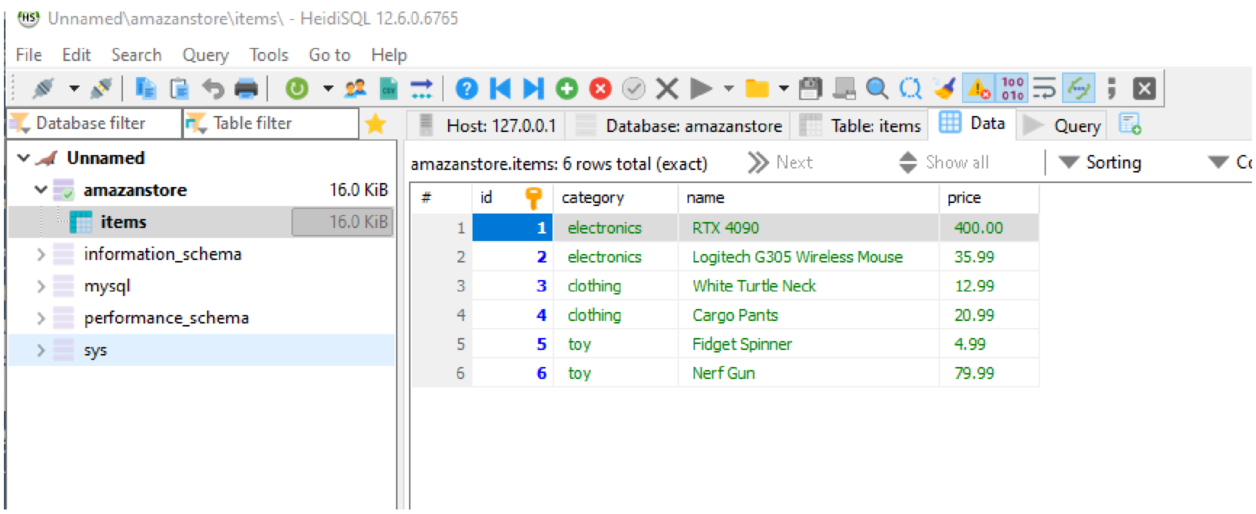
201 CREATED 6 ms 221 B

```
{
    "message": "Database populated successfully"
}
```

```python
    cur = mysql.con
    cur.execute("""
        CREATE TABLE IF NOT EXISTS items (
            id INT AUTO_INCREMENT PRIMARY KEY,
            category VARCHAR(255) NOT NULL,
            name VARCHAR(255) NOT NULL,
            price VARCHAR(255) NOT NULL
        )
    """)
    mysql.connection.commit()
    print(jsonify({'message': 'Empty database created successfully'}), 201)



# Populate table
@app.route('/v1/admin/table')
def fill_tables():
    if itemdb:
        cur = mysql.connection.cursor()
        action = "INSERT INTO items (category, name, price) VALUES (%s, %s,
        cur.executemany(action, itemdb)
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Database populated successfully'}), 201
    return jsonify({'error': 'Input database not found'}), 404



# Create
@app.route('/v1/admin/items', methods=['POST'])
def add_item():
    data = request.json
    category = data.get('category')
```

```
r instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
127.0.0.1 - - [18/Mar/2024 02:08:44] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 02:08:55] "GET /v1 HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 02:09:03] "GET /v1/admin/table HTTP/1.1" 201 -
```

I defined an item database within the source code and then created an endpoint to pass that database to populate the table, instead of repeating the Create endpoint over again for the initial table.

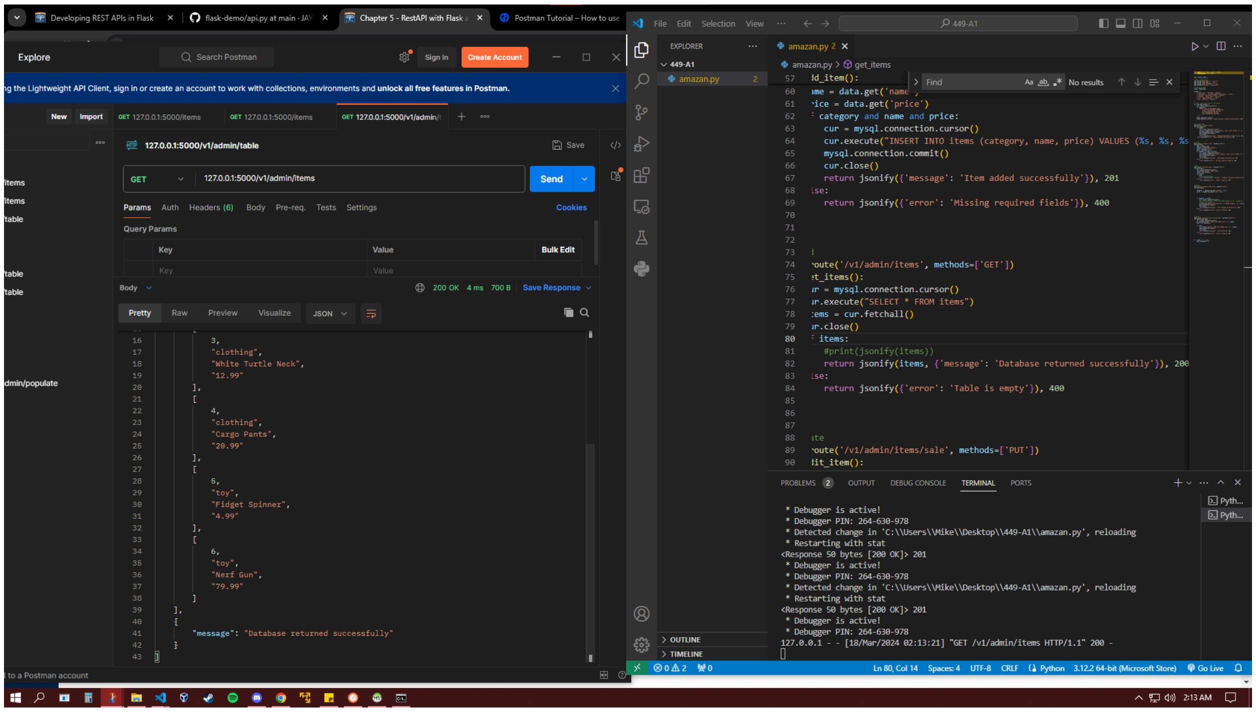


Unnamed\amazanstore\items\ - HeidiSQL 12.6.0.6765

amazanstore.items: 6 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	400.00
2	2	electronics	Logitech G305 Wireless Mouse	35.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	5	toy	Fidget Spinner	4.99
6	6	toy	Nerf Gun	79.99

Amazanstore is the database name and items is the table holding the available items for the shop.



GET 127.0.0.1:5000/v1/admin/items

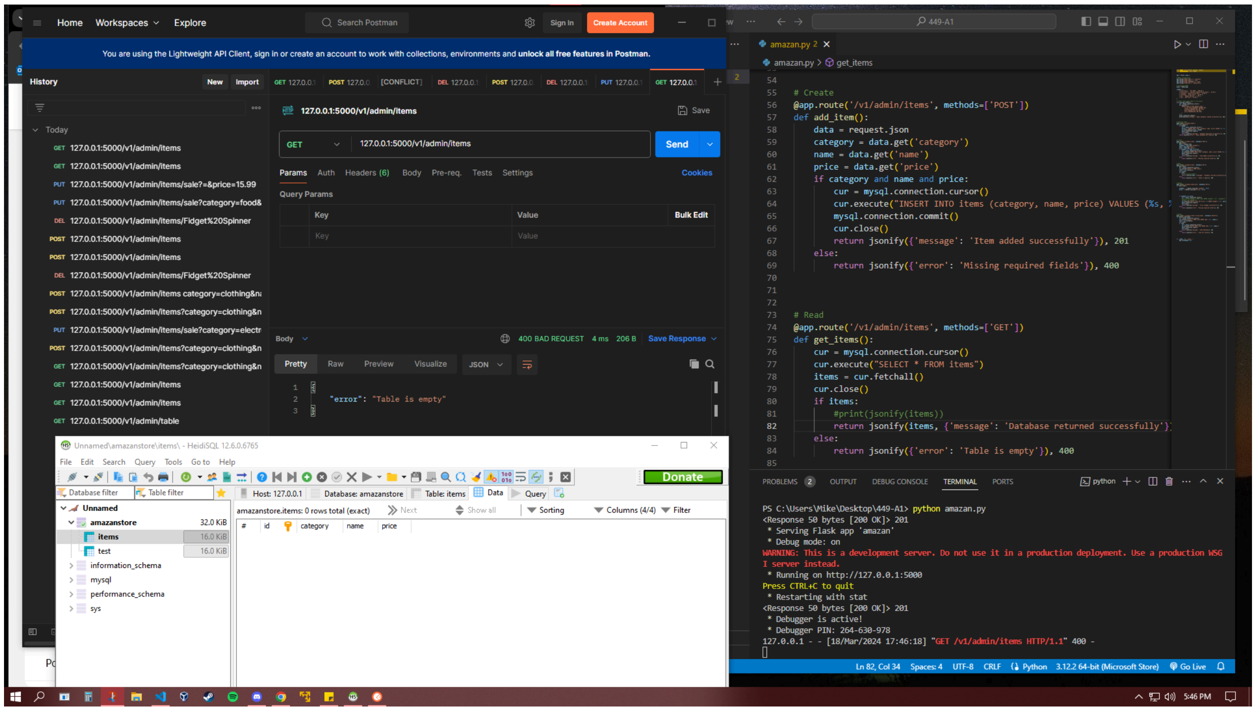
200 OK 4 ms 700 B

```
            3,
            "clothing",
            "White Turtle Neck",
            "12.99"
        ],
        [
            4,
            "clothing",
            "Cargo Pants",
            "20.99"
        ],
        [
            5,
            "toy",
            "Fidget Spinner",
            "4.99"
        ],
        [
            6,
            "toy",
            "Nerf Gun",
            "79.99"
        ]
    ],
    {
        "message": "Database returned successfully"
    }
]
```

```
ld_item():
ame = data.get('name
ice = data.get('price')
 category and name and price:
   cur = mysql.connection.cursor()
   cur.execute("INSERT INTO items (category, name, price) VALUES (%s, %s, %s
   mysql.connection.commit()
   cur.close()
   return jsonify({'message': 'Item added successfully'}), 201
lse:
   return jsonify({'error': 'Missing required fields'}), 400




route('/v1/admin/items', methods=['GET'])
et_items():
ur = mysql.connection.cursor()
ur.execute("SELECT * FROM items")
tems = cur.fetchall()
ur.close()
 items:
   #print(jsonify(items))
   return jsonify(items, {'message': 'Database returned successfully'}), 200
lse:
   return jsonify({'error': 'Table is empty'}), 400



ate
route('/v1/admin/items/sale', methods=['PUT'])
dit_item():
```

```
 * Debugger is active!
 * Debugger PIN: 264-630-978
 * Detected change in 'C:\\Users\\Mike\\Desktop\\449-A1\\amazan.py', reloading
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
 * Detected change in 'C:\\Users\\Mike\\Desktop\\449-A1\\amazan.py', reloading
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
127.0.0.1 - - [18/Mar/2024 02:13:21] "GET /v1/admin/items HTTP/1.1" 200 -
```

Here we used the Read endpoint to gather all the items from the items table.



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History — Today

```
GET   127.0.0.1:5000/v1/admin/items
GET   127.0.0.1:5000/v1/admin/items
PUT   127.0.0.1:5000/v1/admin/items/sale?=&price=15.99
PUT   127.0.0.1:5000/v1/admin/items/sale?category=food&
DEL   127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST  127.0.0.1:5000/v1/admin/items
POST  127.0.0.1:5000/v1/admin/items
DEL   127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST  127.0.0.1:5000/v1/admin/items category=clothing&n
POST  127.0.0.1:5000/v1/admin/items?category=clothing&n
PUT   127.0.0.1:5000/v1/admin/items/sale?category=electr
POST  127.0.0.1:5000/v1/admin/items?category=clothing&n
GET   127.0.0.1:5000/v1/admin/items?category=clothing&n
GET   127.0.0.1:5000/v1/admin/items
GET   127.0.0.1:5000/v1/admin/items
GET   127.0.0.1:5000/v1/admin/table
```

GET 127.0.0.1:5000/v1/admin/items

400 BAD REQUEST 4 ms 206 B

```
{
    "error": "Table is empty"
}
```

Unnamed\amazanstore\items\ - HeidiSQL 12.6.0.6765

Host: 127.0.0.1 | Database: amazanstore | Table: items

amazanstore.items: 0 rows total (exact)

#	id	category	name	price

amazanstore	32.0 KiB
items	16.0 KiB
test	16.0 KiB
information_schema	
mysql	
performance_schema	
sys	

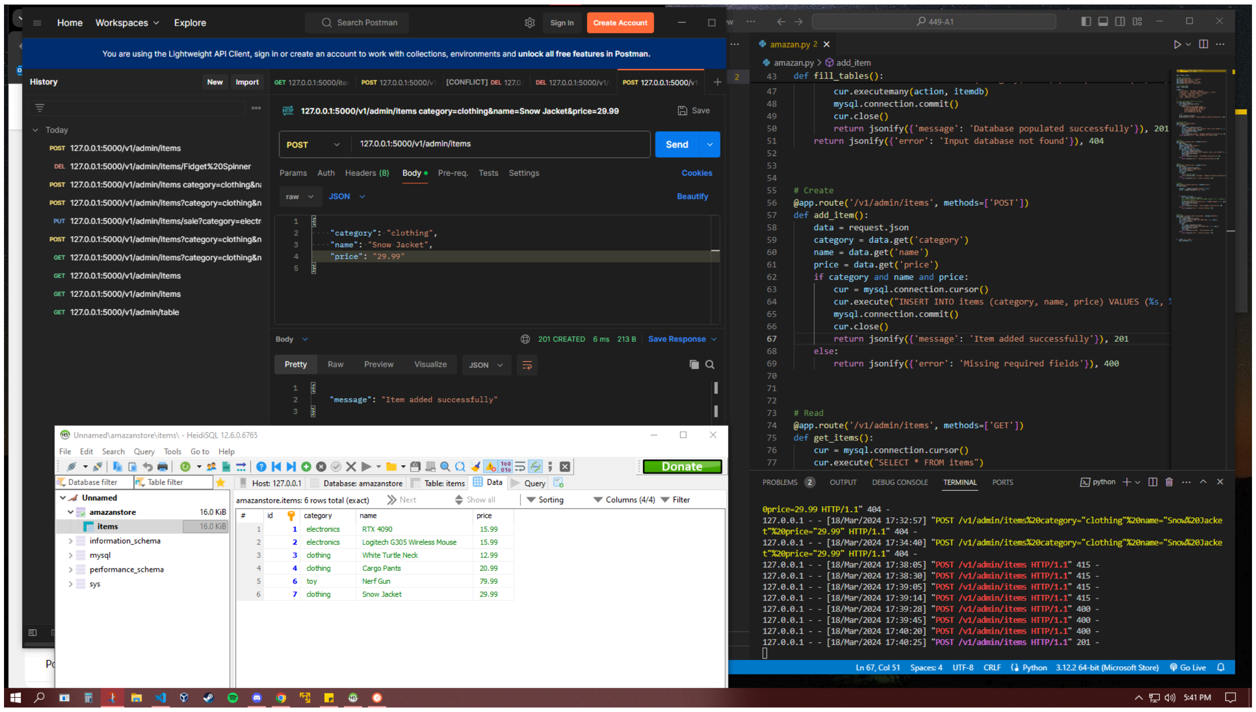
amazan.py > get_items

```python

# Create
@app.route('/v1/admin/items', methods=['POST'])
def add_item():
    data = request.json
    category = data.get('category')
    name = data.get('name')
    price = data.get('price')
    if category and name and price:
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO items (category, name, price) VALUES (%s, %
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Item added successfully'}), 201
    else:
        return jsonify({'error': 'Missing required fields'}), 400



# Read
@app.route('/v1/admin/items', methods=['GET'])
def get_items():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM items")
    items = cur.fetchall()
    cur.close()
    if items:
        #print(jsonify(items))
        return jsonify(items, {'message': 'Database returned successfully'})
    else:
        return jsonify({'error': 'Table is empty'}), 400

```

```
PS C:\Users\Mike\Desktop\449-A1> python amazan.py
<Response 50 bytes [200 OK]> 201
 * Serving Flask app 'amazan'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSG
I server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
127.0.0.1 - - [18/Mar/2024 17:46:18] "GET /v1/admin/items HTTP/1.1" 400 -
```

Ln 82, Col 34 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit (Microsoft Store)

We tested the same endpoint but with an empty table this time, we changed the initial table to test to temporarily hold those items.



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

```
POST 127.0.0.1:5000/v1/admin/items
DEL 127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST 127.0.0.1:5000/v1/admin/items category=clothing&n
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
PUT 127.0.0.1:5000/v1/admin/items/sale?category=electr
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
GET 127.0.0.1:5000/v1/admin/items?category=clothing&n
GET 127.0.0.1:5000/v1/admin/items
GET 127.0.0.1:5000/v1/admin/items
GET 127.0.0.1:5000/v1/admin/table
```

127.0.0.1:5000/v1/admin/items category=clothing&name=Snow Jacket&price=29.99

POST 127.0.0.1:5000/v1/admin/items

```json
{
    "category": "clothing",
    "name": "Snow Jacket",
    "price": "29.99"
}
```

201 CREATED 6 ms 213 B

```json
{
    "message": "Item added successfully"
}
```

amazan.py > add_item

```python
def fill_tables():
            cur.executemany(action, itemdb)
            mysql.connection.commit()
            cur.close()
            return jsonify({'message': 'Database populated successfully'}), 201
        return jsonify({'error': 'Input database not found'}), 404



# Create
@app.route('/v1/admin/items', methods=['POST'])
def add_item():
    data = request.json
    category = data.get('category')
    name = data.get('name')
    price = data.get('price')
    if category and name and price:
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO items (category, name, price) VALUES (%s, %
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Item added successfully'}), 201
    else:
        return jsonify({'error': 'Missing required fields'}), 400



# Read
@app.route('/v1/admin/items', methods=['GET'])
def get_items():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM items")
```

```
0price=29.99 HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:32:57] "POST /v1/admin/items%20category="clothing"%20name="Snow%20Jacke
t"%20price="29.99" HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:34:40] "POST /v1/admin/items%20category="clothing"%20name="Snow%20Jacke
t"%20price="29.99" HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:38:05] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:38:30] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:05] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:14] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:28] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:39:45] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:20] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:25] "POST /v1/admin/items HTTP/1.1" 201 -
```

Ln 67, Col 51 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit (Microsoft Store)

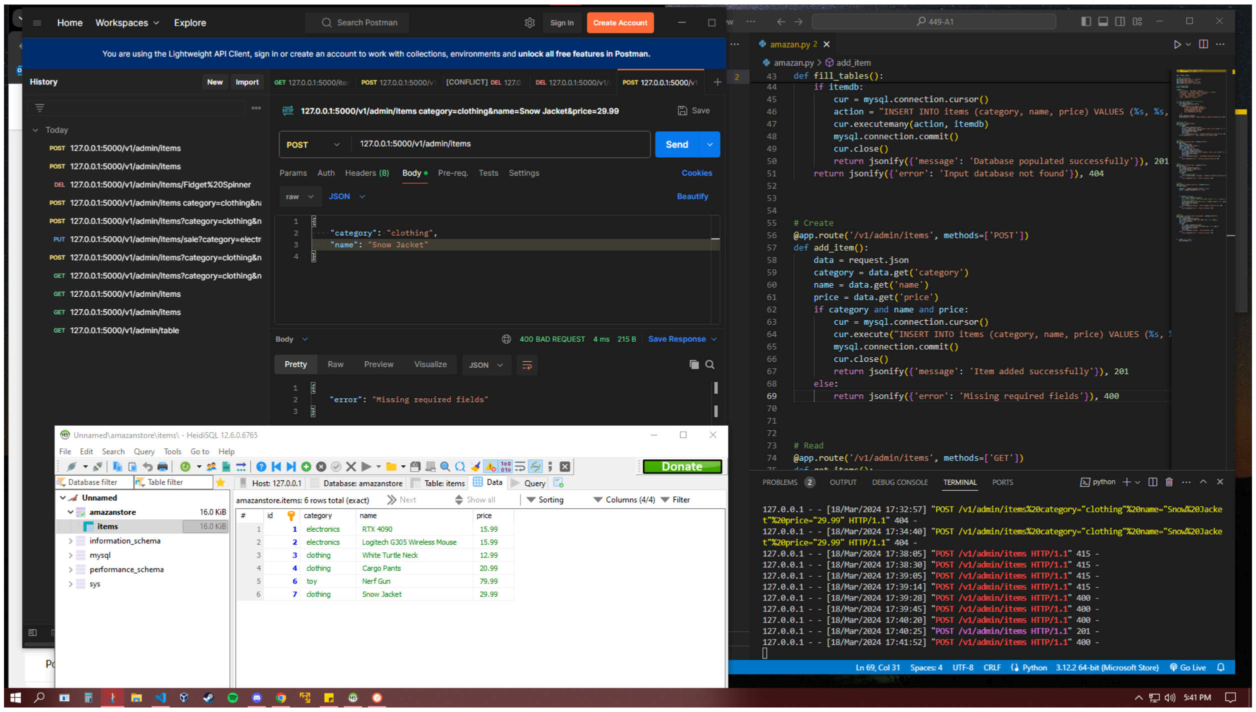
Unnamed\amazanstore\items\ - HeidiSQL 12.6.0.6765

Host: 127.0.0.1 Database: amazanstore Table: items

amazanstore.items: 6 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	15.99
2	2	electronics	Logitech G305 Wireless Mouse	15.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	6	toy	Nerf Gun	79.99
6	7	clothing	Snow Jacket	29.99

We tested the Create endpoint by using the same url but this time include some value in the body tab of the Postman application.



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

```
POST 127.0.0.1:5000/v1/admin/items
POST 127.0.0.1:5000/v1/admin/items
DEL 127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST 127.0.0.1:5000/v1/admin/items category=clothing&n
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
PUT 127.0.0.1:5000/v1/admin/items/sale?category=electr
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
GET 127.0.0.1:5000/v1/admin/items?category=clothing&n
GET 127.0.0.1:5000/v1/admin/items
GET 127.0.0.1:5000/v1/admin/items
GET 127.0.0.1:5000/v1/admin/table
```

127.0.0.1:5000/v1/admin/items category=clothing&name=Snow Jacket&price=29.99

POST 127.0.0.1:5000/v1/admin/items

```json
{
    "category": "clothing",
    "name": "Snow Jacket"
}
```

400 BAD REQUEST 4 ms 215 B

```json
{
    "error": "Missing required fields"
}
```

amazan.py > add_item

```python
def fill_tables():
    if itemdb:
        cur = mysql.connection.cursor()
        action = "INSERT INTO items (category, name, price) VALUES (%s, %s,
        cur.executemany(action, itemdb)
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Database populated successfully'}), 201
    return jsonify({'error': 'Input database not found'}), 404



# Create
@app.route('/v1/admin/items', methods=['POST'])
def add_item():
    data = request.json
    category = data.get('category')
    name = data.get('name')
    price = data.get('price')
    if category and name and price:
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO items (category, name, price) VALUES (%s, %
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Item added successfully'}), 201
    else:
        return jsonify({'error': 'Missing required fields'}), 400



# Read
@app.route('/v1/admin/items', methods=['GET'])
```

```
127.0.0.1 - - [18/Mar/2024 17:32:57] "POST /v1/admin/items%20category="clothing"%20name="Snow%20Jacke
t"%20price="29.99" HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:34:40] "POST /v1/admin/items%20category="clothing"%20name="Snow%20Jacke
t"%20price="29.99" HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:38:05] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:38:30] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:05] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:14] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:28] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:39:45] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:20] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:25] "POST /v1/admin/items HTTP/1.1" 201 -
127.0.0.1 - - [18/Mar/2024 17:41:52] "POST /v1/admin/items HTTP/1.1" 400 -
```

Ln 69, Col 31 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit (Microsoft Store)

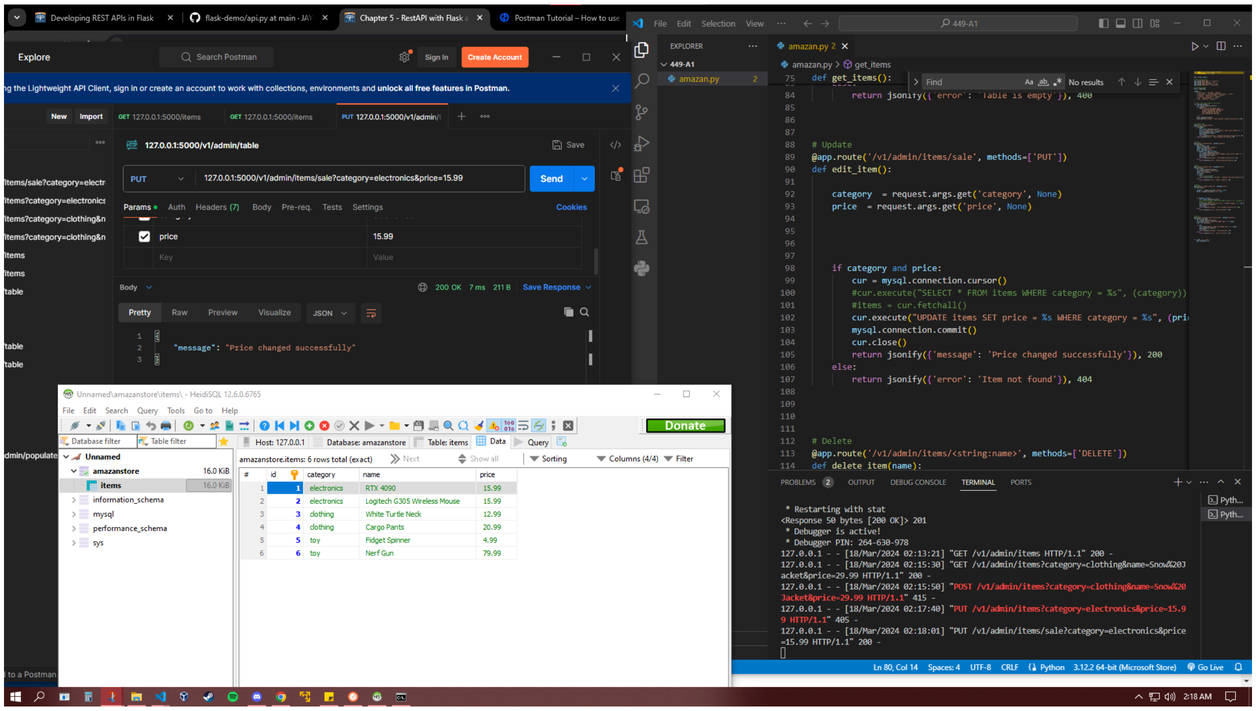
Unnamed\amazanstore\items\ - HeidiSQL 12.6.0.6765

Host: 127.0.0.1 Database: amazanstore Table: items

amazanstore.items: 6 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	15.99
2	2	electronics	Logitech G305 Wireless Mouse	15.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	6	toy	Nerf Gun	79.99
6	7	clothing	Snow Jacket	29.99

We can see the error message by not including all the required fields.



PUT 127.0.0.1:5000/v1/admin/items/sale?category=electronics&price=15.99

	Key	Value
✓	price	15.99

200 OK 7 ms 211 B

```
{
    "message": "Price changed successfully"
}
```

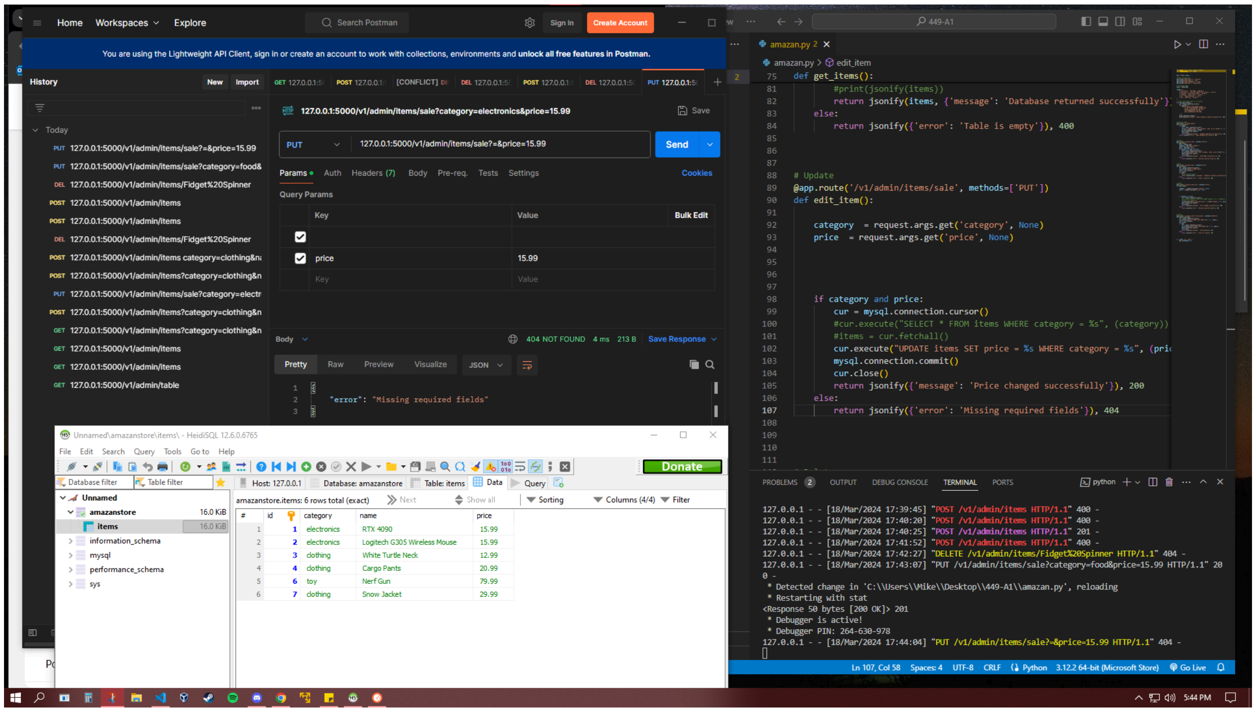
amazanstore.items: 6 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	15.99
2	2	electronics	Logitech G305 Wireless Mouse	15.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	5	toy	Fidget Spinner	4.99
6	6	toy	Nerf Gun	79.99

```python
def get_items():
            return jsonify({'error': 'Table is empty'}), 400



# Update
@app.route('/v1/admin/items/sale', methods=['PUT'])
def edit_item():

    category  = request.args.get('category', None)
    price  = request.args.get('price', None)




    if category and price:
        cur = mysql.connection.cursor()
        #cur.execute("SELECT * FROM items WHERE category = %s", (category))
        #items = cur.fetchall()
        cur.execute("UPDATE items SET price = %s WHERE category = %s", (pri
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Price changed successfully'}), 200
    else:
        return jsonify({'error': 'Item not found'}), 404




# Delete
@app.route('/v1/admin/items/<string:name>', methods=['DELETE'])
def delete item(name):
```

```
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
127.0.0.1 - - [18/Mar/2024 02:13:21] "GET /v1/admin/items HTTP/1.1" 200 -
127.0.0.1 - - [18/Mar/2024 02:15:30] "GET /v1/admin/items?category=clothing&name=Snow%20J
acket&price=29.99 HTTP/1.1" 200 -
127.0.0.1 - - [18/Mar/2024 02:15:50] "POST /v1/admin/items?category=clothing&name=Snow%20
Jacket&price=29.99 HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 02:17:40] "PUT /v1/admin/items?category=electronics&price=15.9
9 HTTP/1.1" 405 -
127.0.0.1 - - [18/Mar/2024 02:18:01] "PUT /v1/admin/items/sale?category=electronics&price
=15.99 HTTP/1.1" 200 -
```

Here we used the Update endpoint to change the prices of all electronics items to represent a sale going on.



History

```
PUT  127.0.0.1:5000/v1/admin/items/sale?=&price=15.99
PUT  127.0.0.1:5000/v1/admin/items/sale?category=food&
DEL  127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST 127.0.0.1:5000/v1/admin/items
POST 127.0.0.1:5000/v1/admin/items
DEL  127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST 127.0.0.1:5000/v1/admin/items category=clothing&n
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
PUT  127.0.0.1:5000/v1/admin/items/sale?category=electr
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
GET  127.0.0.1:5000/v1/admin/items?category=clothing&n
GET  127.0.0.1:5000/v1/admin/items
GET  127.0.0.1:5000/v1/admin/items
GET  127.0.0.1:5000/v1/admin/table
```

PUT 127.0.0.1:5000/v1/admin/items/sale?=&price=15.99

	Key	Value
✓		
✓	price	15.99

404 NOT FOUND 4 ms 213 B

```
{
    "error": "Missing required fields"
}
```

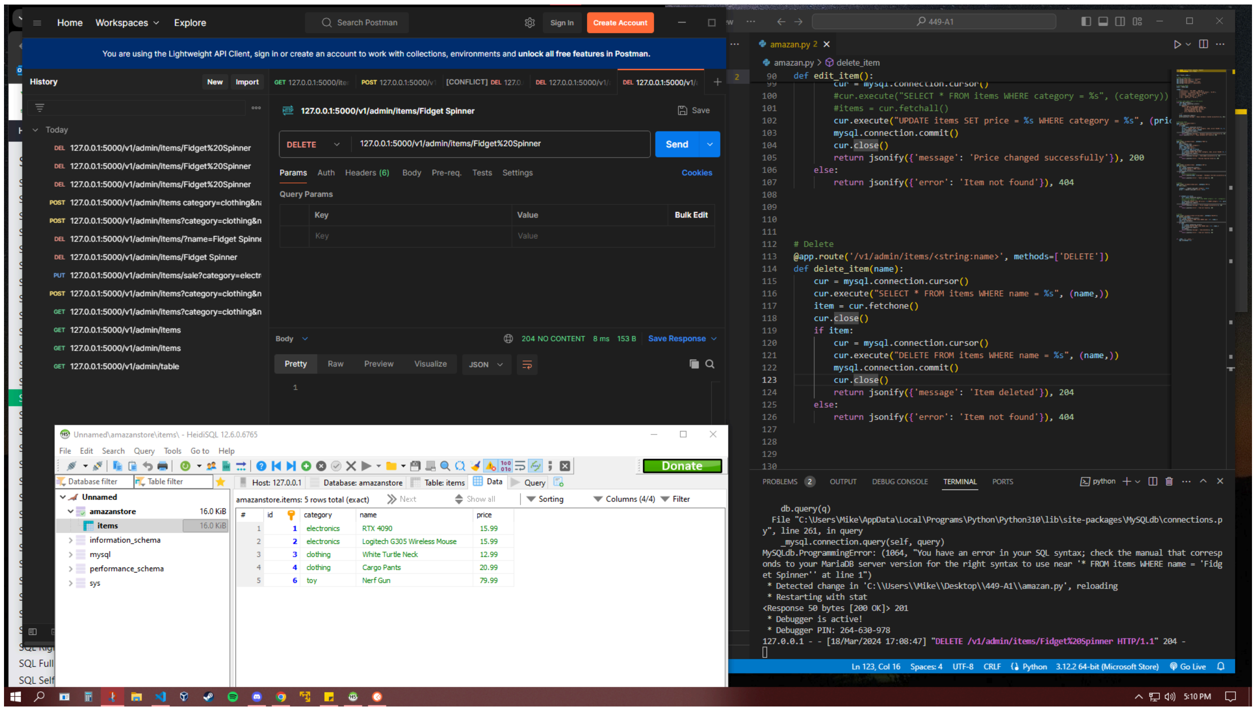
amazanstore.items: 6 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	15.99
2	2	electronics	Logitech G305 Wireless Mouse	15.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	6	toy	Nerf Gun	79.99
6	7	clothing	Snow Jacket	29.99

```python
def get_items():
            #print(jsonify(items))
            return jsonify(items, {'message': 'Database returned successfully'}
        else:
            return jsonify({'error': 'Table is empty'}), 400



# Update
@app.route('/v1/admin/items/sale', methods=['PUT'])
def edit_item():

    category  = request.args.get('category', None)
    price  = request.args.get('price', None)




    if category and price:
        cur = mysql.connection.cursor()
        #cur.execute("SELECT * FROM items WHERE category = %s", (category))
        #items = cur.fetchall()
        cur.execute("UPDATE items SET price = %s WHERE category = %s", (pri
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Price changed successfully'}), 200
    else:
        return jsonify({'error': 'Missing required fields'}), 404
```

```
127.0.0.1 - - [18/Mar/2024 17:39:45] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:20] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:25] "POST /v1/admin/items HTTP/1.1" 201 -
127.0.0.1 - - [18/Mar/2024 17:41:52] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:42:27] "DELETE /v1/admin/items/Fidget%20Spinner HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:43:07] "PUT /v1/admin/items/sale?category=food&price=15.99 HTTP/1.1" 20
0 -
 * Detected change in 'C:\\Users\\Mike\\Desktop\\449-A1\\amazan.py', reloading
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
127.0.0.1 - - [18/Mar/2024 17:44:04] "PUT /v1/admin/items/sale?=&price=15.99 HTTP/1.1" 404 -
```

This shows the error message by not including which category to have a sale for.



```
DELETE    127.0.0.1:5000/v1/admin/items/Fidget%20Spinner

204 NO CONTENT   8 ms   153 B
```

History (Today):

```
DEL   127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
DEL   127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
DEL   127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST  127.0.0.1:5000/v1/admin/items category=clothing&n
POST  127.0.0.1:5000/v1/admin/items?category=clothing&n
DEL   127.0.0.1:5000/v1/admin/items/?name=Fidget Spinne
DEL   127.0.0.1:5000/v1/admin/items/Fidget Spinner
PUT   127.0.0.1:5000/v1/admin/items/sale?category=electr
POST  127.0.0.1:5000/v1/admin/items?category=clothing&n
GET   127.0.0.1:5000/v1/admin/items?category=clothing&n
GET   127.0.0.1:5000/v1/admin/items
GET   127.0.0.1:5000/v1/admin/items
GET   127.0.0.1:5000/v1/admin/table
```

Host: 127.0.0.1 Database: amazanstore Table: items

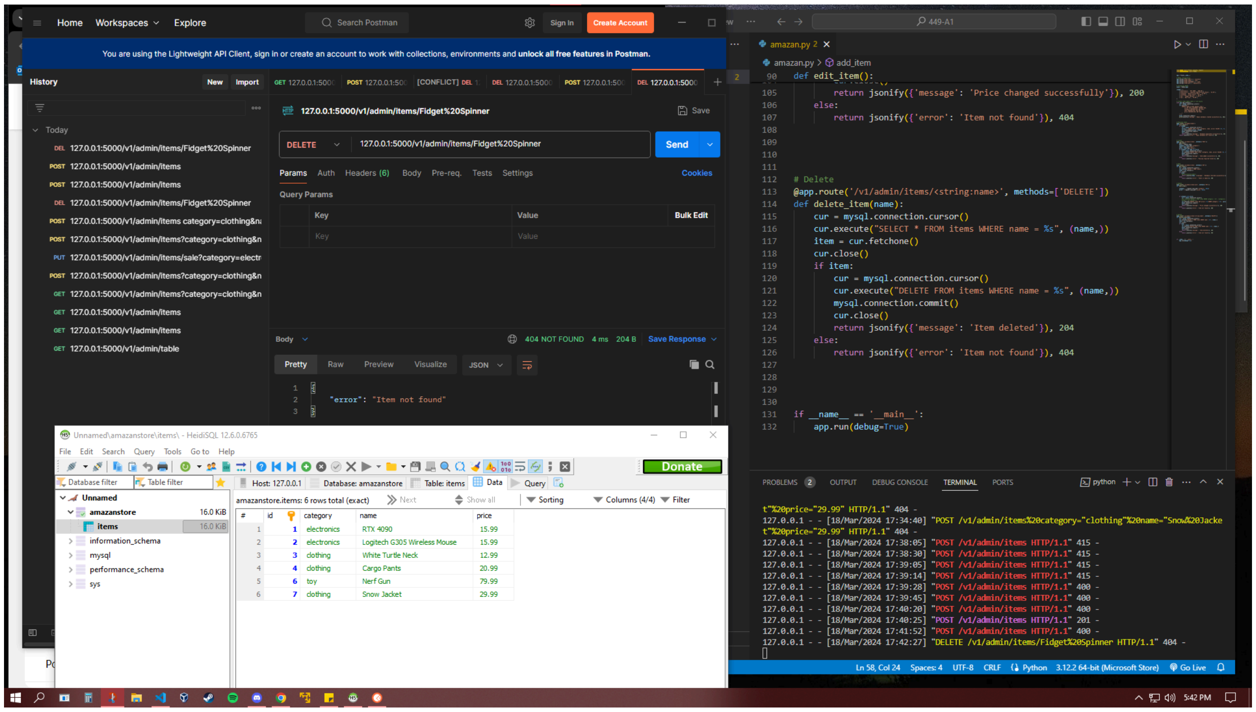
amazanstore.items: 5 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	15.99
2	2	electronics	Logitech G305 Wireless Mouse	15.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	6	toy	Nerf Gun	79.99

```python
def edit_item():
            #cur.execute("SELECT * FROM items WHERE category = %s", (category))
            #items = cur.fetchall()
            cur.execute("UPDATE items SET price = %s WHERE category = %s", (pri
            mysql.connection.commit()
            cur.close()
            return jsonify({'message': 'Price changed successfully'}), 200
        else:
            return jsonify({'error': 'Item not found'}), 404




# Delete
@app.route('/v1/admin/items/<string:name>', methods=['DELETE'])
def delete_item(name):
        cur = mysql.connection.cursor()
        cur.execute("SELECT * FROM items WHERE name = %s", (name,))
        item = cur.fetchone()
        cur.close()
        if item:
            cur = mysql.connection.cursor()
            cur.execute("DELETE FROM items WHERE name = %s", (name,))
            mysql.connection.commit()
            cur.close()
            return jsonify({'message': 'Item deleted'}), 204
        else:
            return jsonify({'error': 'Item not found'}), 404
```

```
    db.query(q)
  File "C:\Users\Mike\AppData\Local\Programs\Python\Python310\lib\site-packages\MySQLdb\connections.p
y", line 261, in query
    _mysql.connection.query(self, query)
MySQLdb.ProgrammingError: (1064, "You have an error in your SQL syntax; check the manual that corresp
onds to your MariaDB server version for the right syntax to use near '* FROM items WHERE name = 'Fidg
et Spinner'' at line 1")
 * Detected change in 'C:\\Users\\Mike\\Desktop\\449-A1\\amazan.py', reloading
 * Restarting with stat
<Response 50 bytes [200 OK]> 201
 * Debugger is active!
 * Debugger PIN: 264-630-978
127.0.0.1 - - [18/Mar/2024 17:08:47] "DELETE /v1/admin/items/Fidget%20Spinner HTTP/1.1" 204 -
```

We demonstrated the Delete endpoint by deleting Fidget Spinner from the shop and no comment was returned except the status code as you can see on the right terminal.


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.
History
New
Import
Today

```
DEL  127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST 127.0.0.1:5000/v1/admin/items
POST 127.0.0.1:5000/v1/admin/items
DEL  127.0.0.1:5000/v1/admin/items/Fidget%20Spinner
POST 127.0.0.1:5000/v1/admin/items category=clothing&na
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
PUT  127.0.0.1:5000/v1/admin/items/sale?category=electr
POST 127.0.0.1:5000/v1/admin/items?category=clothing&n
GET  127.0.0.1:5000/v1/admin/items?category=clothing&n
GET  127.0.0.1:5000/v1/admin/items
GET  127.0.0.1:5000/v1/admin/items
GET  127.0.0.1:5000/v1/admin/table
```

127.0.0.1:5000/v1/admin/items/Fidget%20Spinner

DELETE 127.0.0.1:5000/v1/admin/items/Fidget%20Spinner

404 NOT FOUND 4 ms 204 B

```
{
    "error": "Item not found"
}
```

```
def edit_item():
            return jsonify({'message': 'Price changed successfully'}), 200
        else:
            return jsonify({'error': 'Item not found'}), 404



# Delete
@app.route('/v1/admin/items/<string:name>', methods=['DELETE'])
def delete_item(name):
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM items WHERE name = %s", (name,))
    item = cur.fetchone()
    cur.close()
    if item:
        cur = mysql.connection.cursor()
        cur.execute("DELETE FROM items WHERE name = %s", (name,))
        mysql.connection.commit()
        cur.close()
        return jsonify({'message': 'Item deleted'}), 204
    else:
        return jsonify({'error': 'Item not found'}), 404




if __name__ == '__main__':
    app.run(debug=True)
```

```
t"%20price="29.99" HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:34:40] "POST /v1/admin/items%20category="clothing"%20name="Snow%20Jacke
t"%20price="29.99" HTTP/1.1" 404 -
127.0.0.1 - - [18/Mar/2024 17:38:05] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:38:30] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:05] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:14] "POST /v1/admin/items HTTP/1.1" 415 -
127.0.0.1 - - [18/Mar/2024 17:39:28] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:39:45] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:20] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:40:25] "POST /v1/admin/items HTTP/1.1" 201 -
127.0.0.1 - - [18/Mar/2024 17:41:52] "POST /v1/admin/items HTTP/1.1" 400 -
127.0.0.1 - - [18/Mar/2024 17:42:27] "DELETE /v1/admin/items/Fidget%20Spinner HTTP/1.1" 404 -
```

Ln 58, Col 24 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit (Microsoft Store)

Unnamed\amazanstore\items\ - HeidiSQL 12.6.0.6765

amazanstore.items: 6 rows total (exact)

#	id	category	name	price
1	1	electronics	RTX 4090	15.99
2	2	electronics	Logitech G305 Wireless Mouse	15.99
3	3	clothing	White Turtle Neck	12.99
4	4	clothing	Cargo Pants	20.99
5	6	toy	Nerf Gun	79.99
6	7	clothing	Snow Jacket	29.99

Finally, we tried to delete an item that is not in the table and resulted in item not found with status code 404.