DB Testing using Postman

Summary

Developed and tested a RESTful API using Flask in Ubuntu to perform CRUD operations on a MariaDB database. Integrated Postman for API testing, allowing full interaction with the SQL backend using GET, POST, PUT, and DELETE methods. Verified database responses using both API calls and native SQL queries.

Technologies Used:

- Backend: Python 3, Flask
- Database: MariaDB (SQL)
- Tools: Postman (API testing), Curl (Ubuntu), Terminal
- OS: Ubuntu 22.04

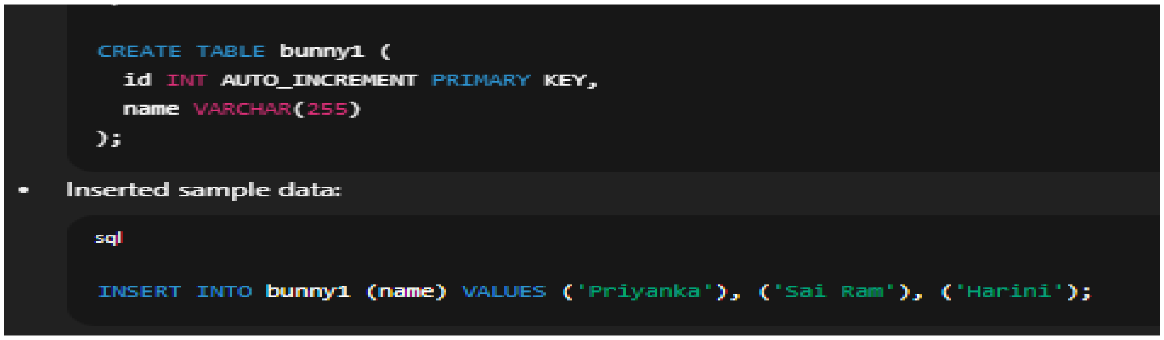
Setup & Implementation Steps:

1. **In Ubuntu**

- sudo apt install python3-pip
- pip3 install flask mysql-connector-python

2. **Database Setup (MariaDB):**

- Installed and configured MariaDB on Ubuntu.
- Created a database: db1
- Created table: bunny1 with columns:



```
CREATE TABLE bunny1 (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255)
);
```

- Inserted sample data:

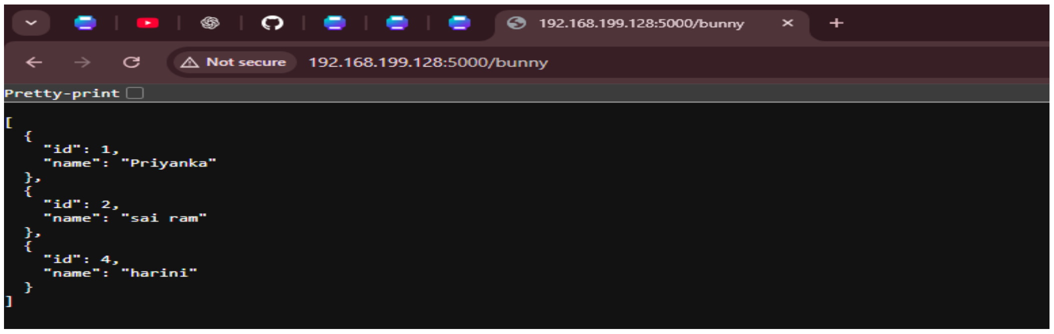
```sql
INSERT INTO bunny1 (name) VALUES ('Priyanka'), ('Sai Ram'), ('Harini');
```

3. **Flask API creation (app.py)**

```
from flask import Flask, request, jsonify
import mysql.connector
app = Flask(__name__)
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="PRIYA1203 ",
    database="db1"
)

@app.route('/bunny', methods=['GET'])
def get_bunnies():
    cursor = conn.cursor(dictionary=True)
    cursor.execute("SELECT * FROM bunny1")
    results = cursor.fetchall()
    cursor.close()
    return jsonify(results)

@app.route('/bunny', methods=['POST'])
def add_bunny():
    data = request.get_json(force=True)
    name = data.get('name')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO bunny1 (name) VALUES (%s)", (name,))
    conn.commit()
    cursor.close()
    return jsonify({"message": f"'{name}' added"}), 201

@app.route('/bunny/<int:id>', methods=['PUT'])
def update_bunny(id):
    data = request.get_json(force=True)
    new_name = data.get('name')
    cursor = conn.cursor()
    cursor.execute("UPDATE bunny1 SET name = %s WHERE id = %s",
(new_name, id))
    conn.commit()
    cursor.close()
    return jsonify({"message": f"Updated ID {id} with name '{new_name}'"}), 200

@app.route('/bunny/<int:id>', methods=['DELETE'])
def delete_bunny(id):
    cursor = conn.cursor()
    cursor.execute("DELETE FROM bunny1 WHERE id = %s", (id,))
    conn.commit()
    cursor.close()
    return jsonify({"message": f"Deleted ID {id}"}), 200

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')  # to allow external access if needed
```

check **python3 app.py** is running

- https://192.168.199.128/bunny in browser



```
[
  {
    "id": 1,
    "name": "Priyanka"
  },
  {
    "id": 2,
    "name": "sai ram"
  },
  {
    "id": 4,
    "name": "harini"
  }
]
```

4. API Testing with Postman

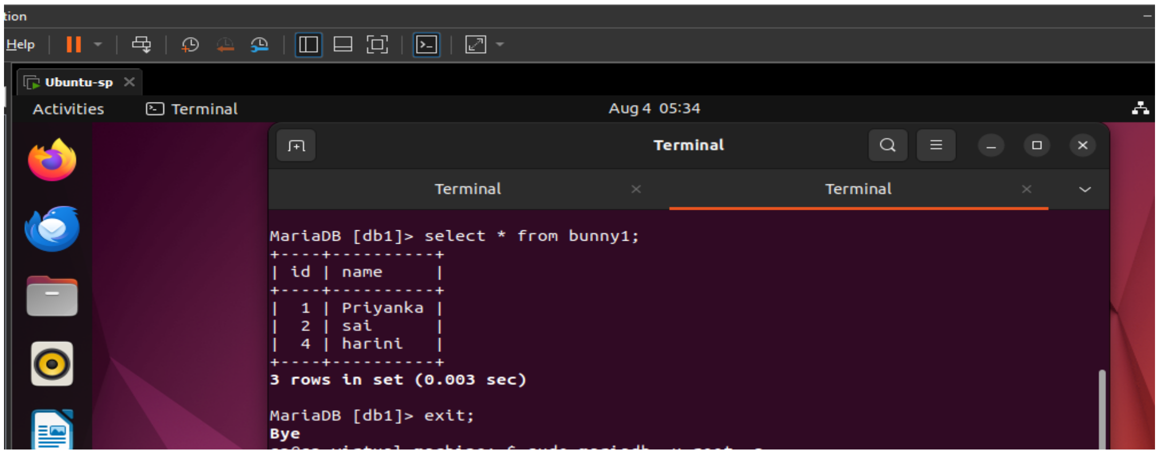
Performed the following:

Method	Endpoint	Function	Sample Body (for POST/PUT)
GET	http://192.168.199.128:5000/bunny	Retrieve all rows	–
POST	http://127.0.0.1:5000/bunny	Insert new name	{ "name": "Riya" }
PUT	http://127.0.0.1:5000/bunny/2	Update name by ID	{ "name": "Sai" }
DELETE	http://127.0.0.1:5000/bunny/3	Delete record by ID	–

Verified all actions using both:

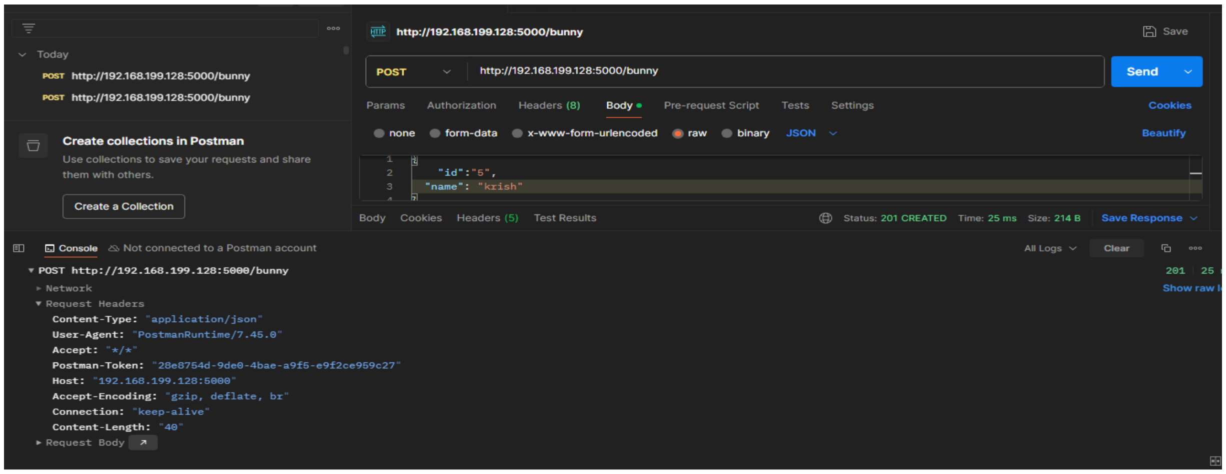
- Postman JSON response
- Direct SQL query: SELECT * FROM bunny1;

Output



```
MariaDB [db1]> select * from bunny1;
+----+----------+
| id | name     |
+----+----------+
|  1 | Priyanka |
|  2 | sai      |
|  4 | harini   |
+----+----------+
3 rows in set (0.003 sec)

MariaDB [db1]> exit;
Bye
```

After post method



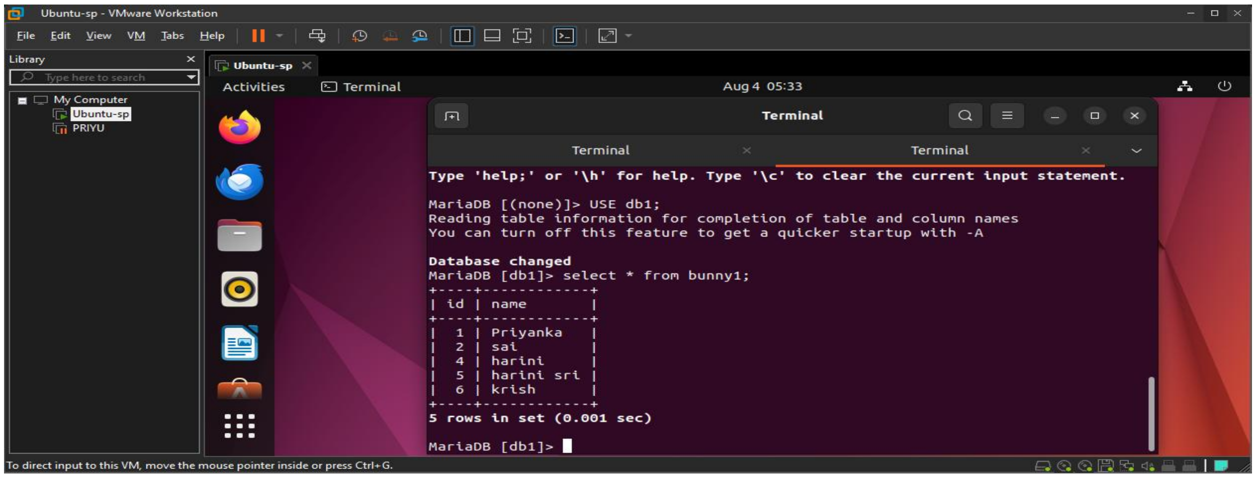
http://192.168.199.128:5000/bunny

POST http://192.168.199.128:5000/bunny

```
{
    "id":"5",
  "name": "krish"
}
```

Status: 201 CREATED Time: 25 ms Size: 214 B

```
▾ POST http://192.168.199.128:5000/bunny
  ▸ Network
  ▾ Request Headers
    Content-Type: "application/json"
    User-Agent: "PostmanRuntime/7.45.0"
    Accept: "*/*"
    Postman-Token: "28e8754d-9de0-4bae-a9f5-e9f2ce959c27"
    Host: "192.168.199.128:5000"
    Accept-Encoding: "gzip, deflate, br"
    Connection: "keep-alive"
    Content-Length: "40"
  ▸ Request Body
```



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE db1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [db1]> select * from bunny1;
+----+------------+
| id | name       |
+----+------------+
|  1 | Priyanka   |
|  2 | sai        |
|  4 | harini     |
|  5 | harini sri |
|  6 | krish      |
+----+------------+
5 rows in set (0.001 sec)

MariaDB [db1]> 
```