GTP Programming, Ops, and Database Challenge (AWS RDS + Flask + FastAPI)

Cletus Nehinlalei Mangu
DevOps & Cloud Engineer at AmaliTech GH

May 5, 2025

Contents

1 Introduction

This project demonstrates the creation of a database system using AWS RDS for MySQL and Python-based APIs. It includes database modeling, SQL analytics, API design using Flask and FastAPI, and API testing with Postman.

2 Database Setup on AWS RDS

2.1 Steps

1. Launched an AWS RDS MySQL instance (version 8.0).
2. Enabled public access and set port 3306 open to my IP.
3. Connected using MySQL Workbench for schema creation and data insertion.

3 SQL Schema and Data Insertion

The database contains 4 main tables: `customers`, `products`, `orders`, and `order_items`. Below is the schema setup.

3.1 Schema Creation

Listing 1: Database Schema Script

```sql
DROP TABLE IF EXISTS order_items;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS customers;

CREATE TABLE customers (
 customer_id INT PRIMARY KEY,
 name VARCHAR(100),
 email VARCHAR(100) UNIQUE,
 country VARCHAR(50)
);
-- Products, Orders, Order Items follow...
```

3.2 Data Insertion

Listing 2: Sample Data Insert

```sql
INSERT INTO customers VALUES
(1, 'Alice Smith', 'alice@example.com', 'USA'),
(2, 'Bob Jones', 'bob@example.com', 'Canada'),
(3, 'Charlie Zhang', 'charlie@example.com', 'UK');
-- Products, Orders, and Order Items follow...
```

4 Analytical Queries

Each SQL query below extracts useful transactions from the data.

4.1 Top Customers by Spending

```sql
SELECT c.name, SUM(oi.quantity * oi.unit_price) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.name
ORDER BY total_spent DESC;
```

4.2 Monthly Sales Report

```sql
SELECT DATE_FORMAT(order_date, '%Y-%m') AS month,
       SUM(quantity * unit_price) AS total_sales
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
WHERE status IN ('Shipped', 'Delivered')
GROUP BY month
ORDER BY month;
```

4.3 Products Never Ordered

```sql
SELECT name FROM products
WHERE product_id NOT IN (
    SELECT DISTINCT product_id FROM order_items
);
```

4.4 Average Order Value by Country

```sql
SELECT c.country,
       AVG(oi.quantity * oi.unit_price) AS avg_order_value
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.country;
```

4.5 Frequent Buyers

```sql
SELECT c.name, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.name
HAVING total_orders > 1;
```

5 API Development

Two APIs were built using Flask and FastAPI, exposing the above queries as endpoints.

5.1 Flask API

- File: `flask_api/app.py` - Endpoints: `/top-customers`, `/monthly-sales`, etc. - Uses `mysql-connector-python` for DB connection.

Listing 3: Flask API Sample Code

```
@app.route('/top-customers', methods=['GET'])
def top_customers():
    cursor.execute(SQL_QUERY)
    results = cursor.fetchall()
    return jsonify(results)
```

5.2 FastAPI

- File: `fastapi_api/main.py` - Same endpoints as Flask - Auto generates docs at `/docs` using Swagger UI

Listing 4: FastAPI Sample Code

```
@app.get("/top-customers")
def get_top_customers():
    cursor.execute(SQL_QUERY)
    results = cursor.fetchall()
    return results
```

6 Testing with Postman

Each API was tested using Postman. A collection was exported as `postman_collection.json`. Screenshots of successful responses are included.

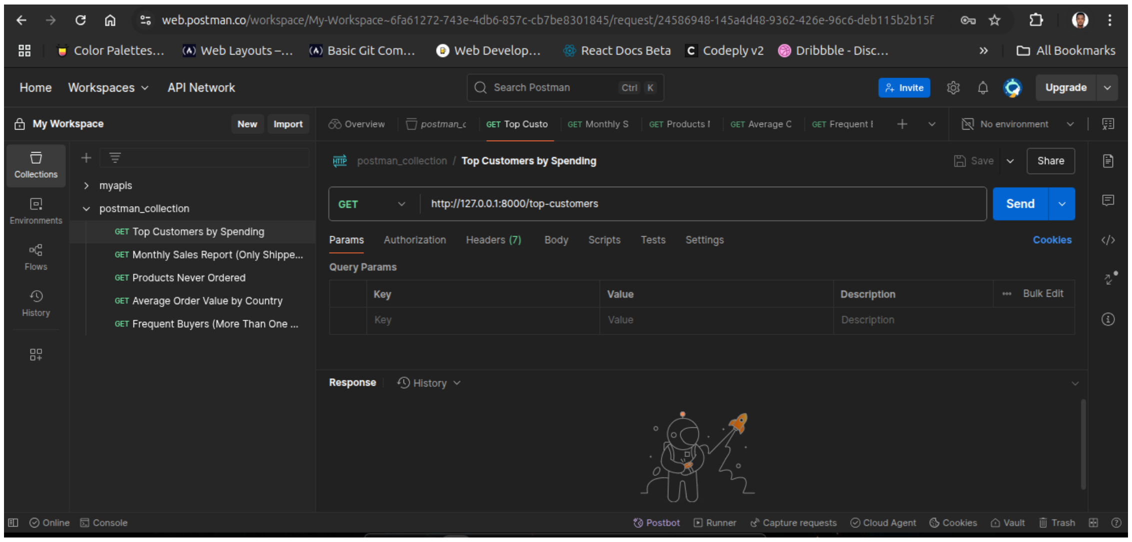
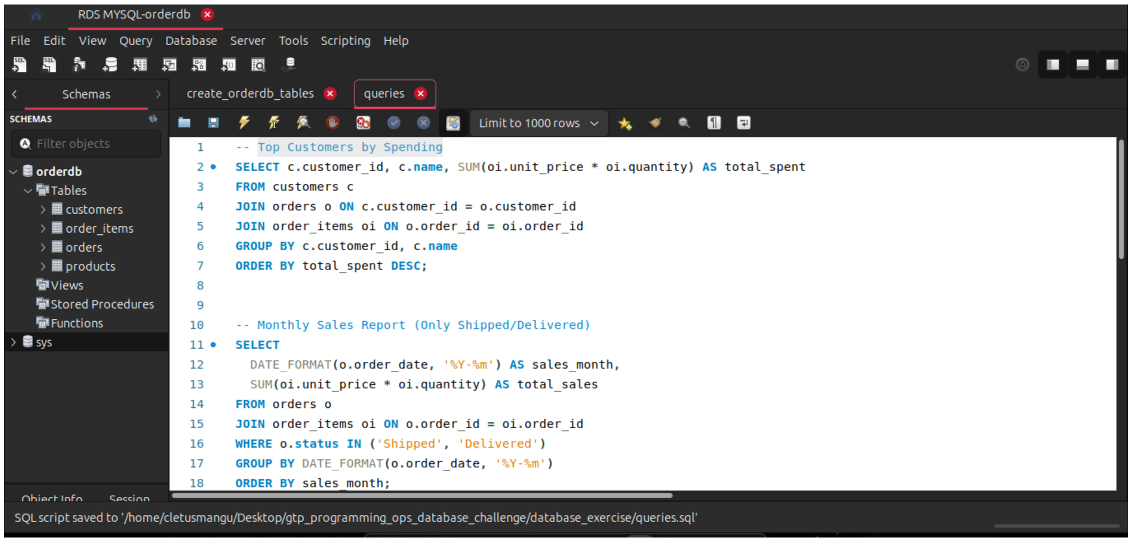

web.postman.co/workspace/My-Workspace~6fa61272-743e-4db6-857c-cb7be8301845/request/24586948-145a4d48-9362-426e-96c6-deb115b2b15f
My Workspace
Collections
Environments
Flows
History
myapis
postman_collection
GET Top Customers by Spending
GET Monthly Sales Report (Only Shippe...
GET Products Never Ordered
GET Average Order Value by Country
GET Frequent Buyers (More Than One ...
postman_collection / Top Customers by Spending
GET
http://127.0.0.1:8000/top-customers
Send
Params
Authorization
Headers (7)
Body
Scripts
Tests
Settings
Query Params
Key
Value
Description
Response

Figure 1: Postman API Documentation

7 Results and Screenshots

Below are results from SQL queries and API responses.

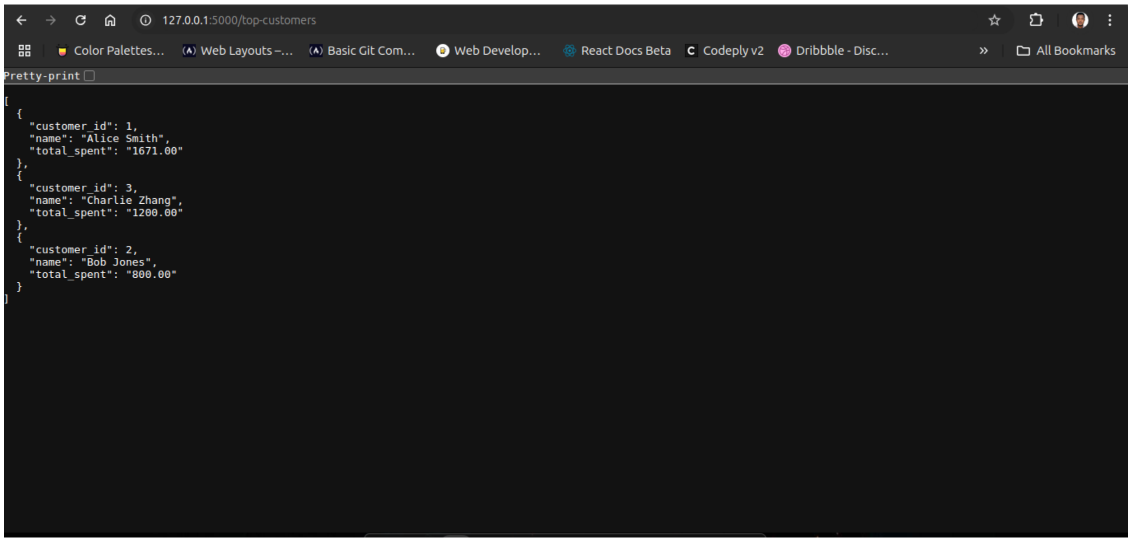


```sql
-- Top Customers by Spending
SELECT c.customer_id, c.name, SUM(oi.unit_price * oi.quantity) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC;


-- Monthly Sales Report (Only Shipped/Delivered)
SELECT
  DATE_FORMAT(o.order_date, '%Y-%m') AS sales_month,
  SUM(oi.unit_price * oi.quantity) AS total_sales
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
WHERE o.status IN ('Shipped', 'Delivered')
GROUP BY DATE_FORMAT(o.order_date, '%Y-%m')
ORDER BY sales_month;
```

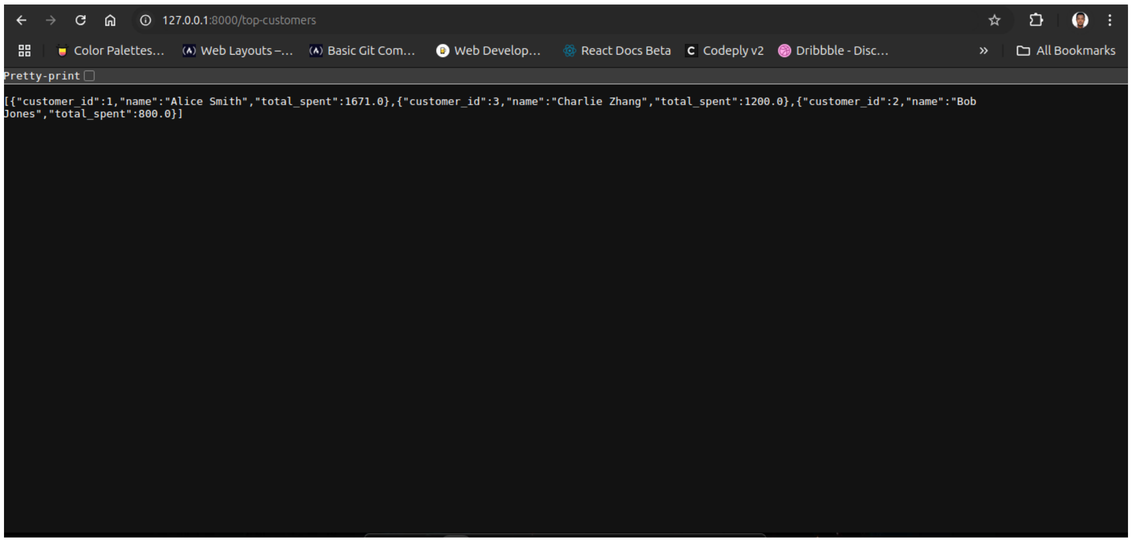
```
SQL script saved to '/home/cletusmangu/Desktop/gtp_programming_ops_database_challenge/database_exercise/queries.sql'
```

Figure 2: SQL Query Results in MySQL Workbench



```
[
  {
    "customer_id": 1,
    "name": "Alice Smith",
    "total_spent": "1671.00"
  },
  {
    "customer_id": 3,
    "name": "Charlie Zhang",
    "total_spent": "1200.00"
  },
  {
    "customer_id": 2,
    "name": "Bob Jones",
    "total_spent": "800.00"
  }
]
```

Figure 3: Flask API Response



```
[{"customer_id":1,"name":"Alice Smith","total_spent":1671.0},{"customer_id":3,"name":"Charlie Zhang","total_spent":1200.0},{"customer_id":2,"name":"Bob
Jones","total_spent":800.0}]
```

Figure 4: FastAPI Response

8 Project Structure

```
database_exercise/
|- flask_api/
|   |- app.py
|- fastapi_api/
|   |- main.py
|- postman_collection.json
|- queries.sql
|- screenshots/
|   |- workbench_queries.png
```

```
    |   |- flask_response.png
    |   |- fastapi_response.png
    |   |- postman_api_docs.png
    |- README.md
```

9 Conclusion

This project showcases an end-to-end deployment of analytics using AWS RDS and Python APIs. It covers cloud databases, SQL analysis, REST APIs, and API documentation.

10 References

- AWS RDS Documentation: `https://docs.aws.amazon.com/rds/`
- Flask Documentation: `https://flask.palletsprojects.com/`
- FastAPI Documentation: `https://fastapi.tiangolo.com/`
- Postman: `https://www.postman.com/`