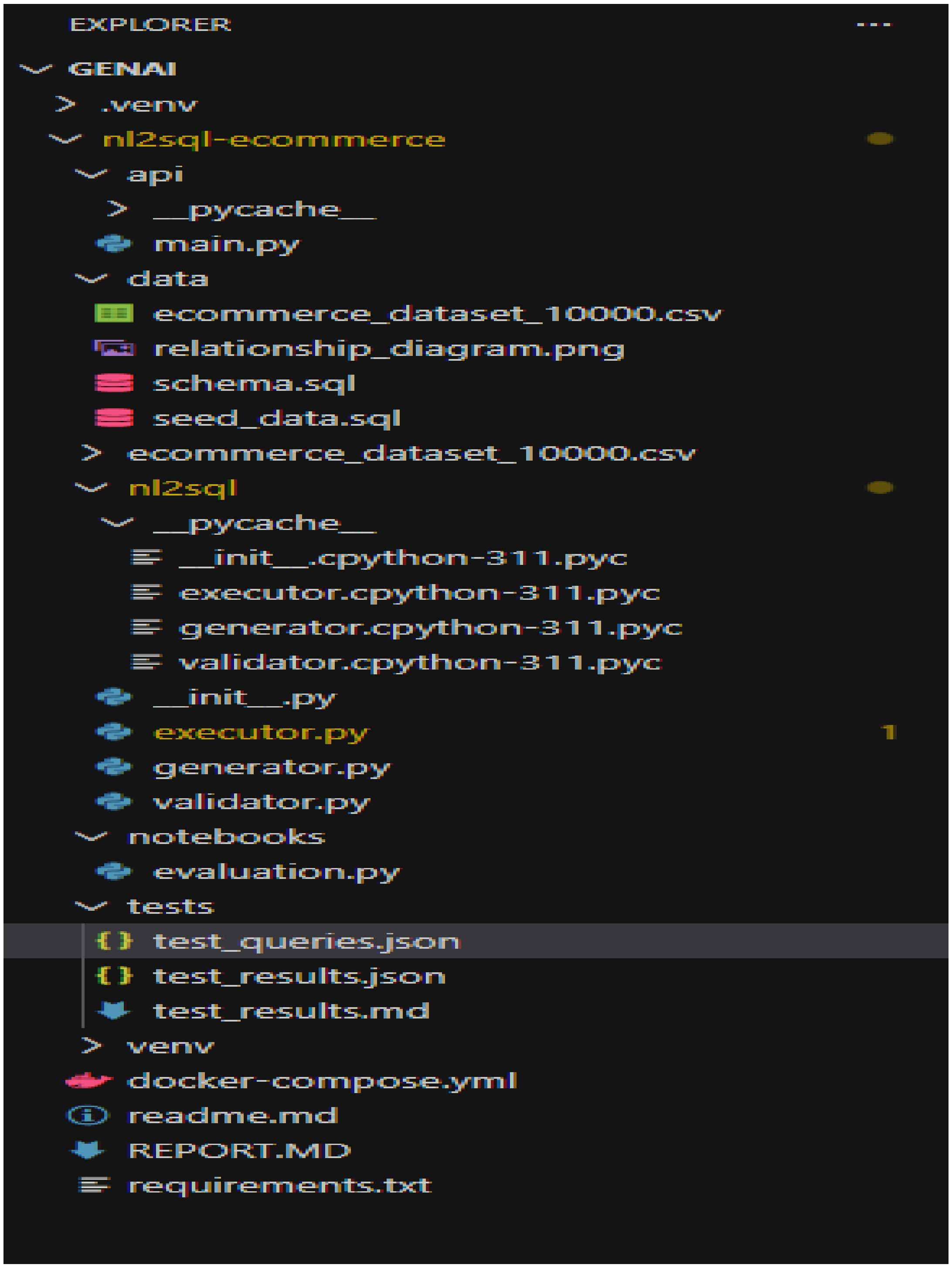
10-Screenshot Demo Checklist-

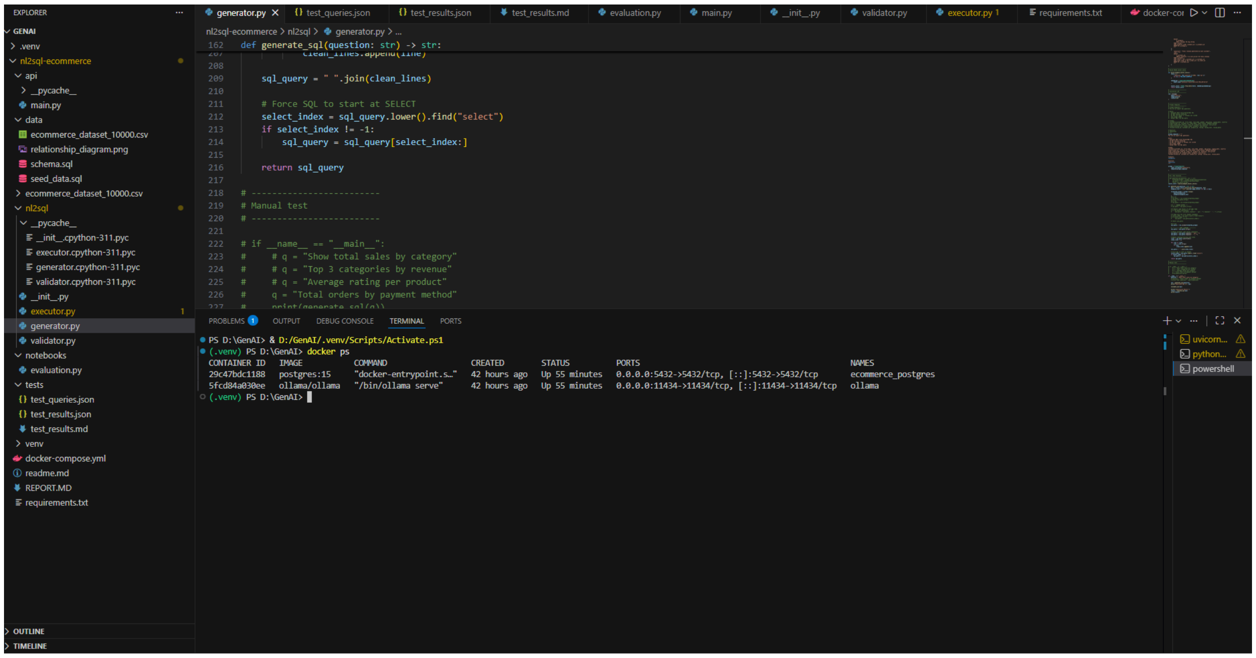
```
demo/
├── 01_project_structure.png
├── 02_docker_running.png
├── 03_swagger_home.png
├── 04_schema_endpoint.png
        ├── nl2sql endpoints
        ├── validator endpoints
        ├── schema endpoints
├── 05_easy_query.png
├── 06_medium_query.png
├── 07_hard_query.png
├── 08_safety_block.png
├── 09_evaluation_results.png
├── 10_neo4j_diagram.png
```

1. Screenshot 1 — Project Structure



```
EXPLORER                                        ...
∨ GENAI
  > .venv
  ∨ nl2sql-ecommerce                                 ●
    ∨ api
      > __pycache__
      main.py
    ∨ data
      ecommerce_dataset_10000.csv
      relationship_diagram.png
      schema.sql
      seed_data.sql
    > ecommerce_dataset_10000.csv
    ∨ nl2sql                                         ●
      ∨ __pycache__
        __init__.cpython-311.pyc
        executor.cpython-311.pyc
        generator.cpython-311.pyc
        validator.cpython-311.pyc
      __init__.py
      executor.py                                    1
      generator.py
      validator.py
    ∨ notebooks
      evaluation.py
    ∨ tests
      test_queries.json
      test_results.json
      test_results.md
    > venv
    docker-compose.yml
    readme.md
    REPORT.MD
    requirements.txt
```

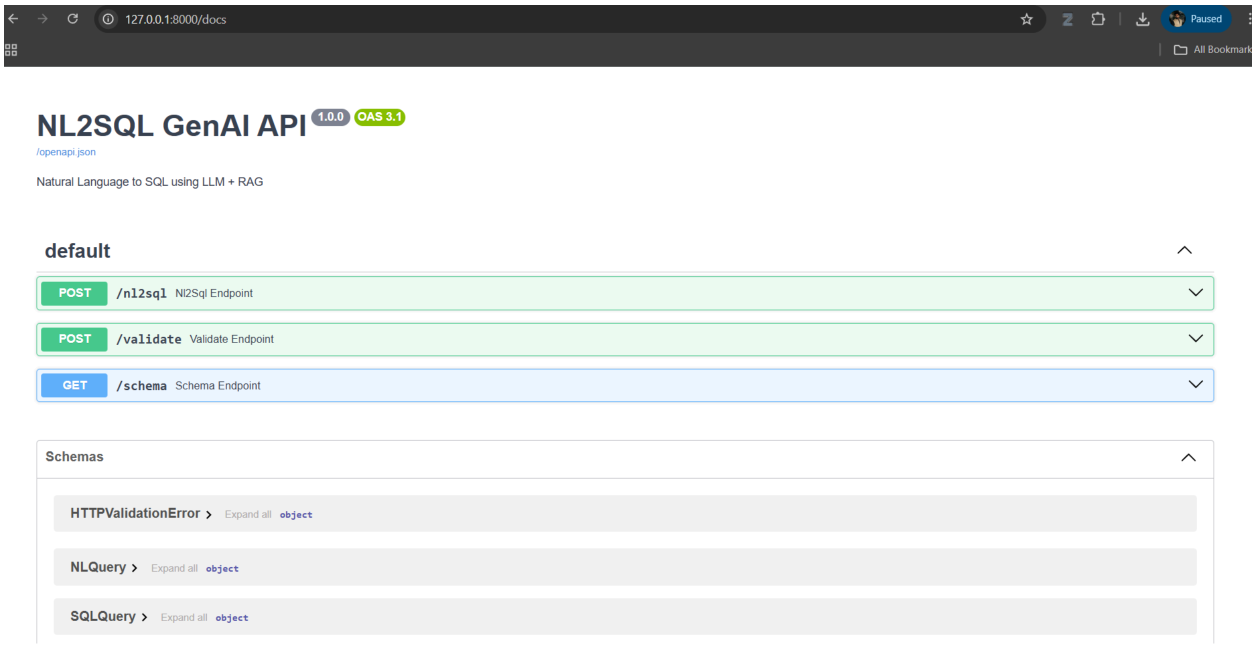
2. Docker container running- ecommerce_postgres, ollama



```python
def generate_sql(question: str) -> str:

    sql_query = " ".join(clean_lines)

    # Force SQL to start at SELECT
    select_index = sql_query.lower().find("select")
    if select_index != -1:
        sql_query = sql_query[select_index:]

    return sql_query

# ------------------------
# Manual test
# ------------------------

# if __name__ == "__main__":
#     # q = "Show total sales by category"
#     # q = "Top 3 categories by revenue"
#     # q = "Average rating per product"
#     q = "Total orders by payment method"
```

```
PS D:\GenAI> & D:/GenAI/.venv/Scripts/Activate.ps1
(.venv) PS D:\GenAI> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED        STATUS          PORTS                                           NAMES
29c47bdc1188   postgres:15     "docker-entrypoint.s…"   42 hours ago   Up 55 minutes   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp     ecommerce_postgres
5fcd84a030ee   ollama/ollama   "/bin/ollama serve"      42 hours ago   Up 55 minutes   0.0.0.0:11434->11434/tcp, [::]:11434->11434/tcp ollama
(.venv) PS D:\GenAI>
```

3. Fast api swagger ui Home - https://127.0.0.1:8000/docs
Which will show - /nl2sql , /validate, /schema


127.0.0.1:8000/docs

NL2SQL GenAI API 1.0.0 OAS 3.1

/openapi.json

Natural Language to SQL using LLM + RAG

default

POST /nl2sql Nl2Sql Endpoint

POST /validate Validate Endpoint

GET /schema Schema Endpoint

Schemas

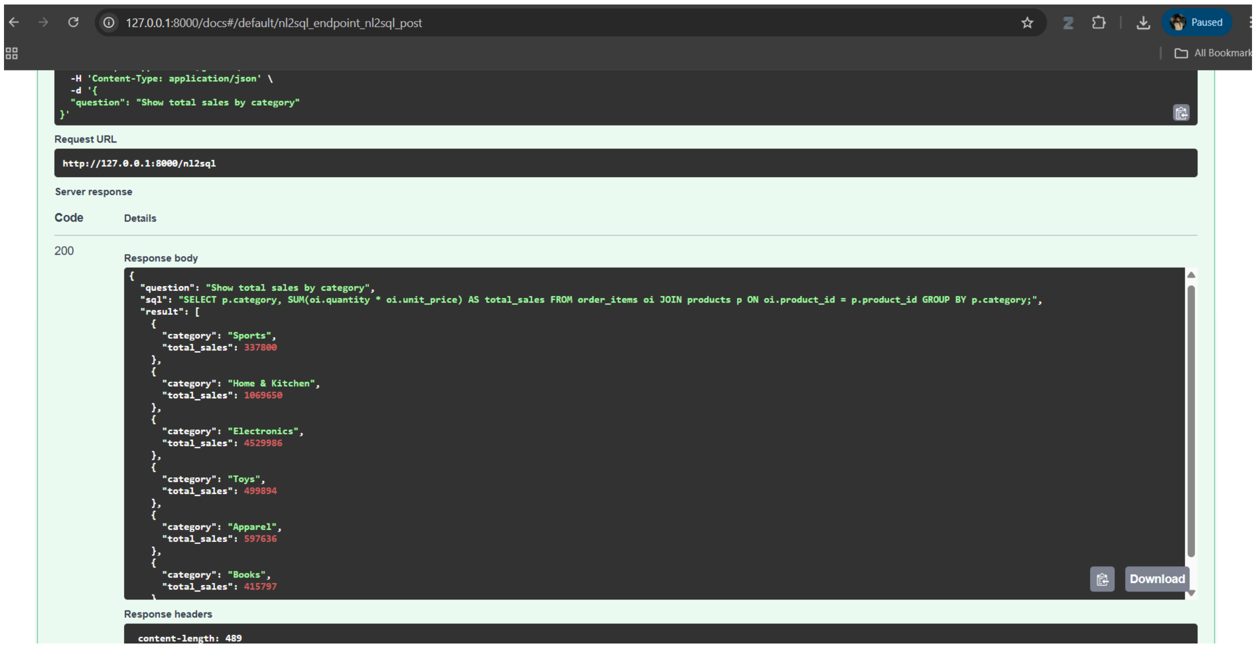
HTTPValidationError > Expand all object

NLQuery > Expand all object

SQLQuery > Expand all object

4. 1. nl2sql endpoints are working -

If i am trying for the query - `Show total sales by category, POST / nl2sql will show-`



127.0.0.1:8000/docs#/default/nl2sql_endpoint_nl2sql_post

```
  -H 'Content-Type: application/json' \
  -d '{
  "question": "Show total sales by category"
}'
```

Request URL

```
http://127.0.0.1:8000/nl2sql
```

Server response

Code	Details
200	Response body

```
{
  "question": "Show total sales by category",
  "sql": "SELECT p.category, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi JOIN products p ON oi.product_id = p.product_id GROUP BY p.category;",
  "result": [
    {
      "category": "Sports",
      "total_sales": 337800
    },
    {
      "category": "Home & Kitchen",
      "total_sales": 1069650
    },
    {
      "category": "Electronics",
      "total_sales": 4529986
    },
    {
      "category": "Toys",
      "total_sales": 499894
    },
    {
      "category": "Apparel",
      "total_sales": 597636
    },
    {
      "category": "Books",
      "total_sales": 415797
```

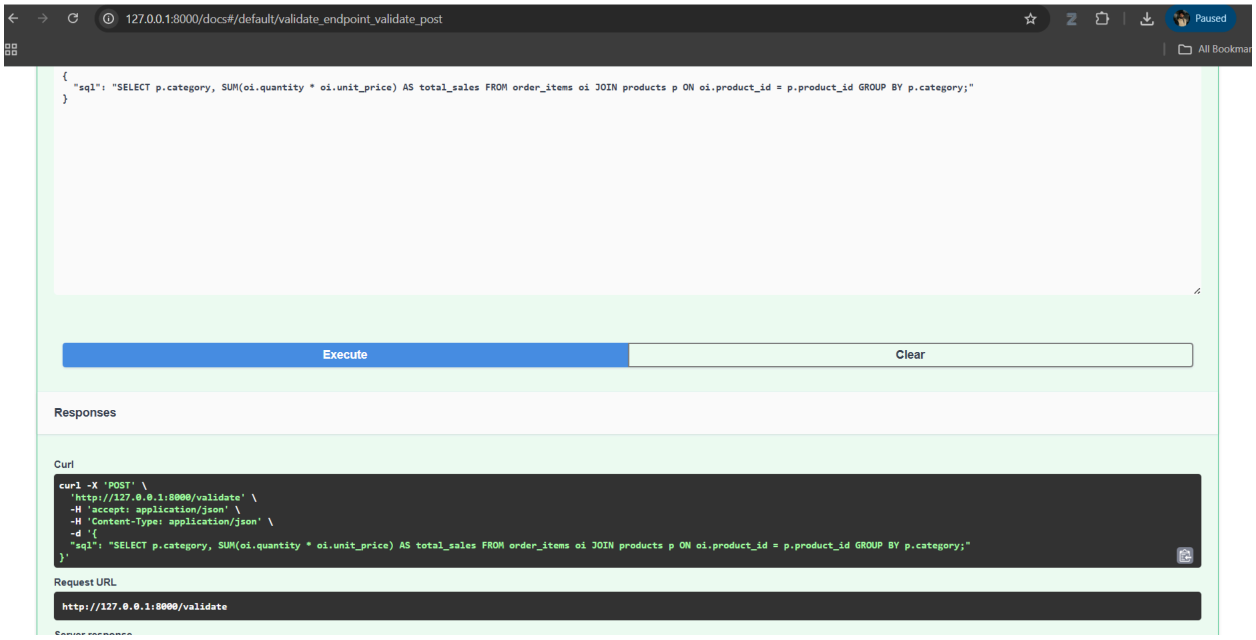
Response headers

```
content-length: 489
```

4.2. POST / validate endpoints are working

To validate POST / validate - input the above query -

```
SELECT p.category, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi
JOIN products p ON oi.product_id = p.product_id GROUP BY p.category;
```


127.0.0.1:8000/docs#/default/validate_endpoint_validate_post
Paused
All Bookmar
Execute
Clear

```
{
  "sql": "SELECT p.category, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi JOIN products p ON oi.product_id = p.product_id GROUP BY p.category;"
}
```

Responses

Curl

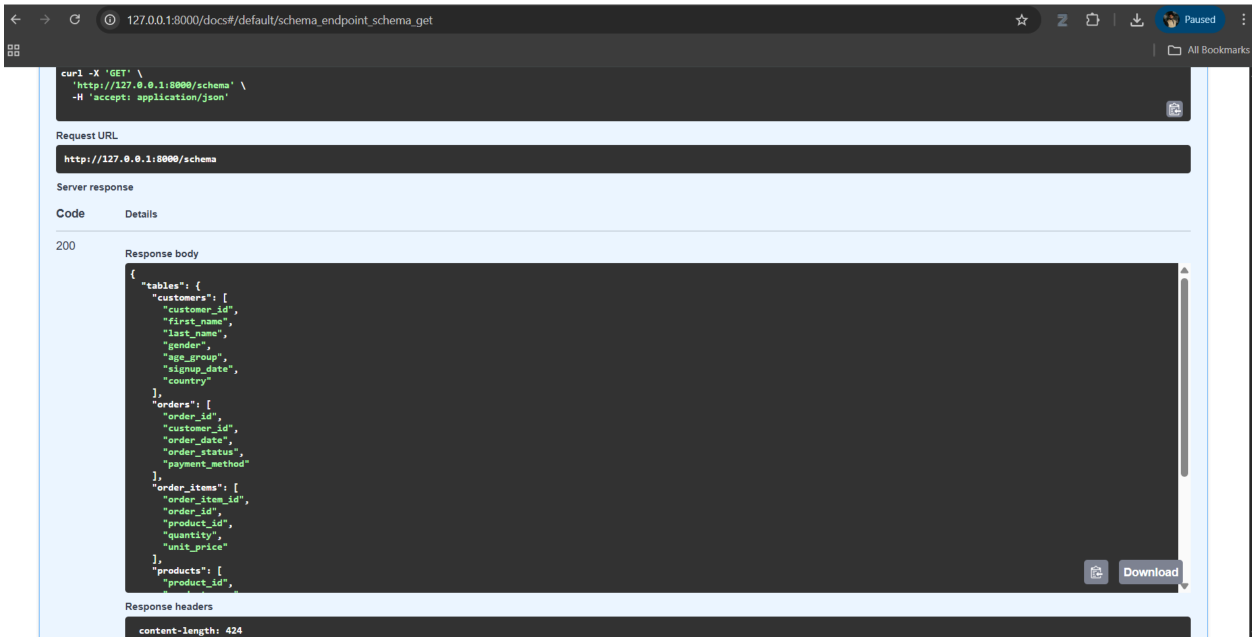
```
curl -X 'POST' \
  'http://127.0.0.1:8000/validate' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "sql": "SELECT p.category, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi JOIN products p ON oi.product_id = p.product_id GROUP BY p.category;"
}'
```

Request URL

```
http://127.0.0.1:8000/validate
```

4.3. GET / schema - endpoints are working

To know the schema or tables used - GET/schema -



```
curl -X 'GET' \
  'http://127.0.0.1:8000/schema' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/schema
```

Server response

Code Details

200

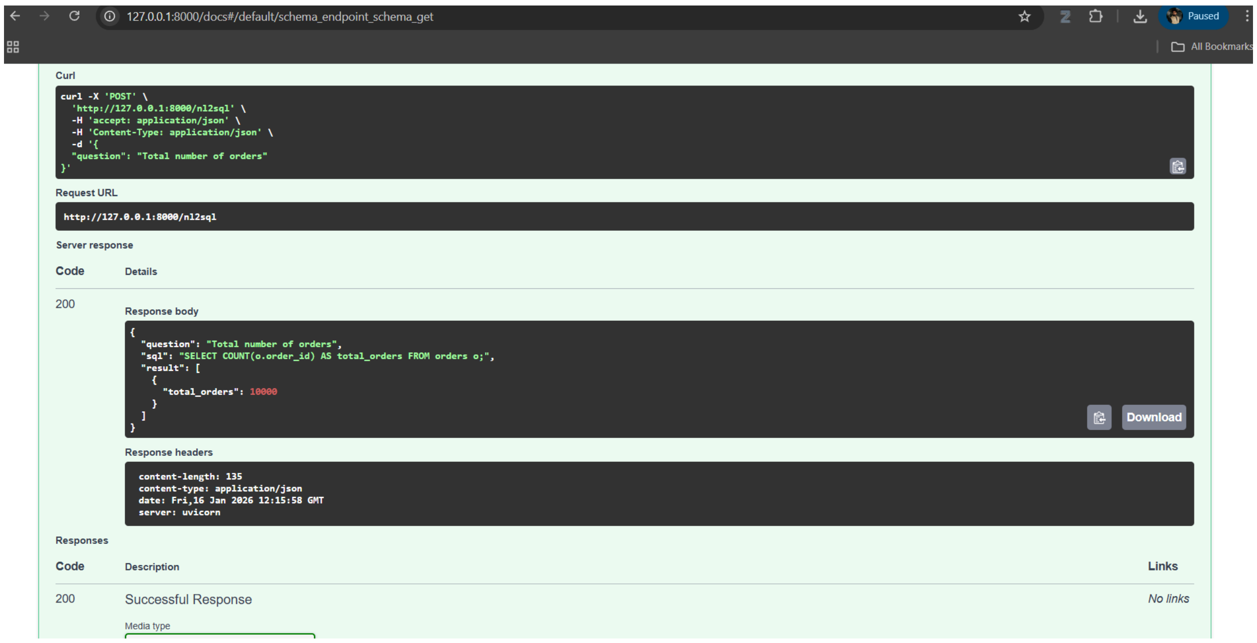
Response body

```
{
  "tables": {
    "customers": [
      "customer_id",
      "first_name",
      "last_name",
      "gender",
      "age_group",
      "signup_date",
      "country"
    ],
    "orders": [
      "order_id",
      "customer_id",
      "order_date",
      "order_status",
      "payment_method"
    ],
    "order_items": [
      "order_item_id",
      "order_id",
      "product_id",
      "quantity",
      "unit_price"
    ],
    "products": [
      "product_id",
```

Response headers

```
content-length: 424
```

5. easy_query
   {
   “Question ”: “Total number of orders”
   }



127.0.0.1:8000/docs#/default/schema_endpoint_schema_get

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/nl2sql' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "question": "Total number of orders"
}'
```

Request URL

```
http://127.0.0.1:8000/nl2sql
```

Server response

Code Details

200

Response body

```
{
  "question": "Total number of orders",
  "sql": "SELECT COUNT(o.order_id) AS total_orders FROM orders o;",
  "result": [
    {
      "total_orders": 10000
    }
  ]
}
```

Response headers

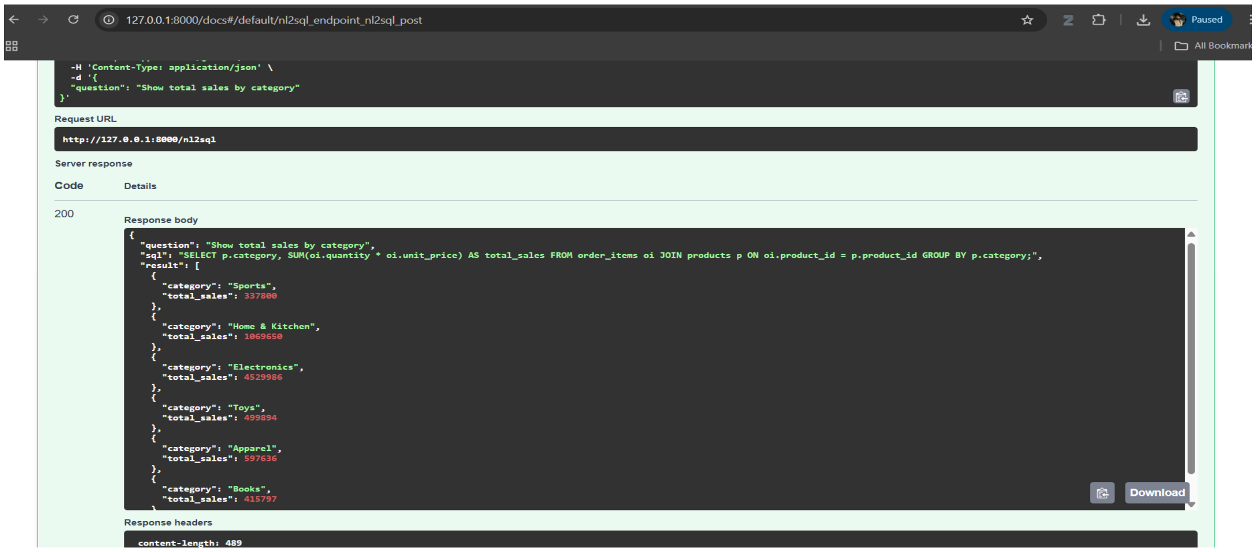
```
 content-length: 135
 content-type: application/json
 date: Fri,16 Jan 2026 12:15:58 GMT
 server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

6. Aggregation query (Medium)
   {
     "question": "Total sales by category"
   }



127.0.0.1:8000/docs#/default/nl2sql_endpoint_nl2sql_post

```
  -H 'Content-Type: application/json' \
  -d '{
  "question": "Show total sales by category"
}'
```

Request URL

```
http://127.0.0.1:8000/nl2sql
```

Server response

Code Details

200

Response body

```
{
  "question": "Show total sales by category",
  "sql": "SELECT p.category, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi JOIN products p ON oi.product_id = p.product_id GROUP BY p.category;",
  "result": [
    {
      "category": "Sports",
      "total_sales": 337800
    },
    {
      "category": "Home & Kitchen",
      "total_sales": 1069650
    },
    {
      "category": "Electronics",
      "total_sales": 4529986
    },
    {
      "category": "Toys",
      "total_sales": 499894
    },
    {
      "category": "Apparel",
      "total_sales": 597636
    },
    {
      "category": "Books",
      "total_sales": 415797
```

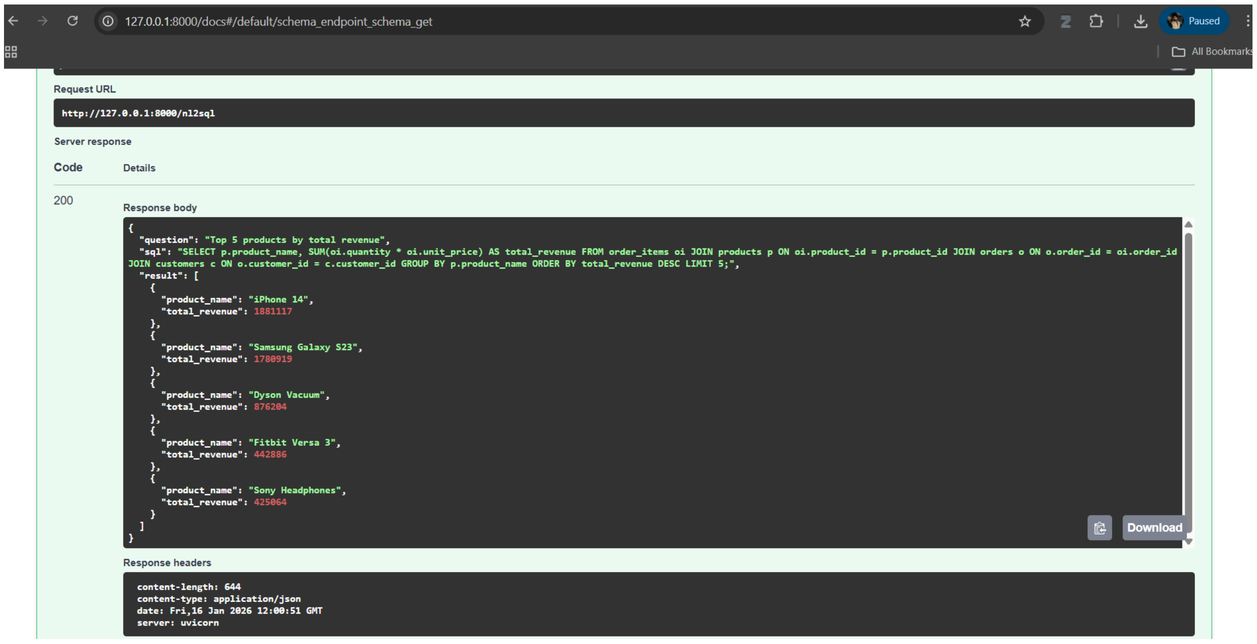
Response headers

```
 content-length: 489
```

Show - JOINs, GROUP BY, Numeric output - screen shot already attached of this

7. Complex Query (Hard) -

```
{
  "question": "Top 5 products by total revenue"
}
```



Request URL

```
http://127.0.0.1:8000/nl2sql
```

Server response

Code Details

200

Response body

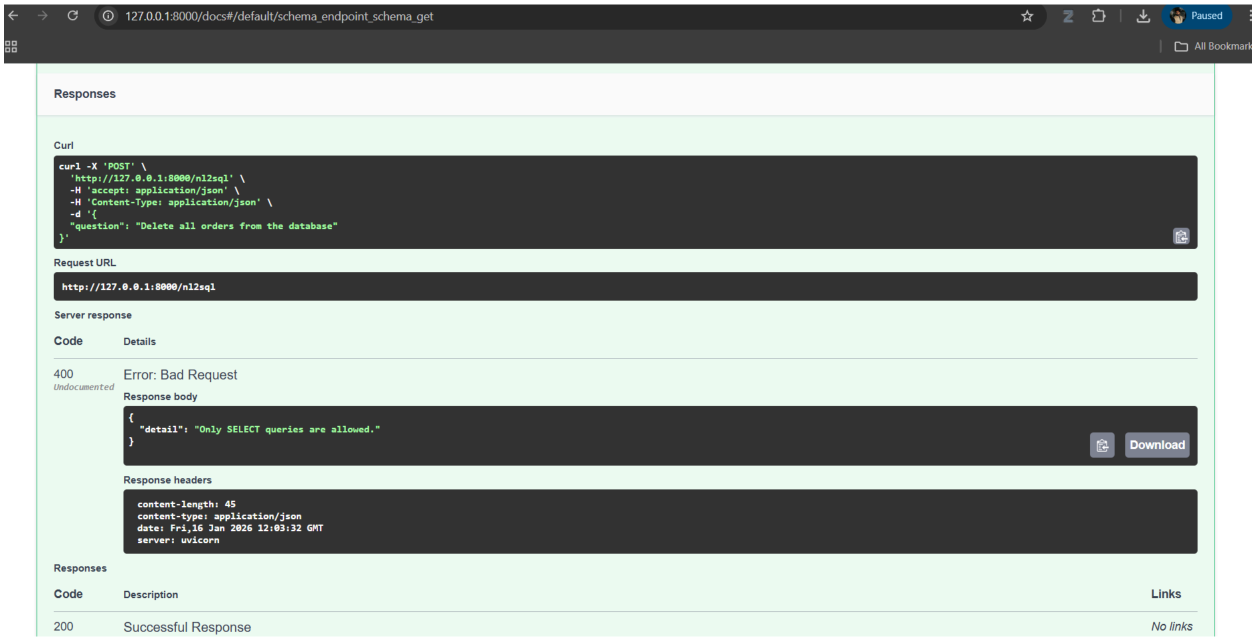
```
{
  "question": "Top 5 products by total revenue",
  "sql": "SELECT p.product_name, SUM(oi.quantity * oi.unit_price) AS total_revenue FROM order_items oi JOIN products p ON oi.product_id = p.product_id JOIN orders o ON o.order_id = oi.order_id
JOIN customers c ON o.customer_id = c.customer_id GROUP BY p.product_name ORDER BY total_revenue DESC LIMIT 5;",
  "result": [
    {
      "product_name": "iPhone 14",
      "total_revenue": 1881117
    },
    {
      "product_name": "Samsung Galaxy S23",
      "total_revenue": 1780919
    },
    {
      "product_name": "Dyson Vacuum",
      "total_revenue": 876204
    },
    {
      "product_name": "Fitbit Versa 3",
      "total_revenue": 442886
    },
    {
      "product_name": "Sony Headphones",
      "total_revenue": 425064
    }
  ]
}
```

Response headers

```
content-length: 644
content-type: application/json
date: Fri,16 Jan 2026 12:00:51 GMT
server: uvicorn
```

8. Safety Validation (Blocked Query) -

```
{
  "question": "Delete all orders from the database"
}
```



Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/nl2sql' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "question": "Delete all orders from the database"
}'
```

Request URL

```
http://127.0.0.1:8000/nl2sql
```

Server response

Code Details

400 *Undocumented* Error: Bad Request

Response body

```
{
  "detail": "Only SELECT queries are allowed."
}
```

Response headers

```
content-length: 45
content-type: application/json
date: Fri,16 Jan 2026 12:03:32 GMT
server: uvicorn
```

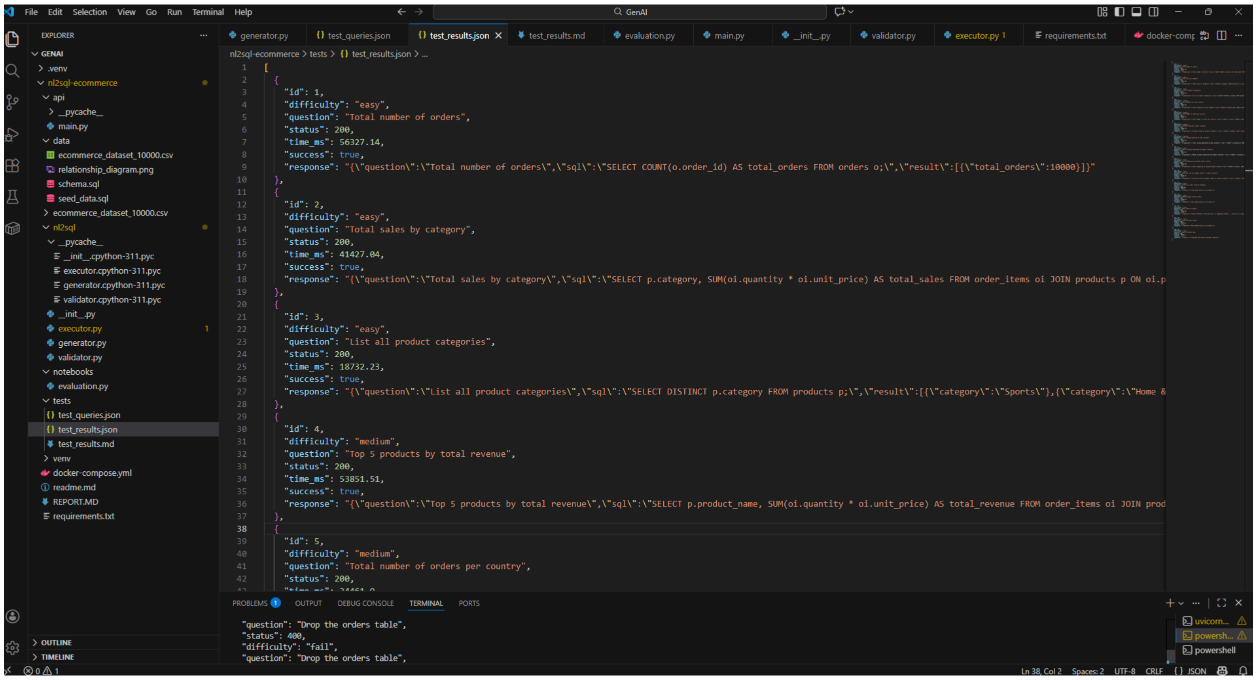
Responses

Code Description Links

200 Successful Response *No links*

9. Evaluation Script Results -
python notebooks/evaluation.py

Show - Success cases, Fail cases, Execution time

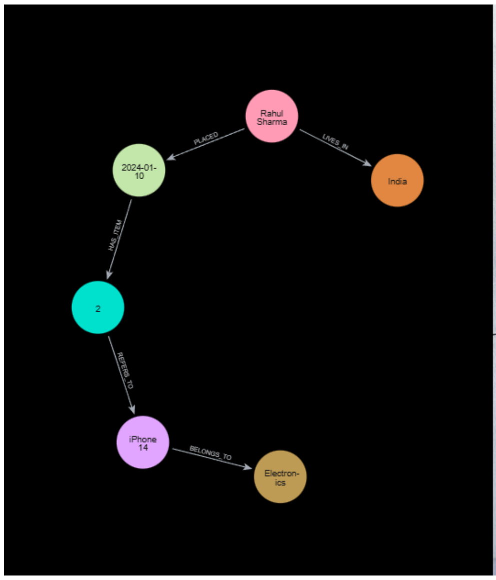


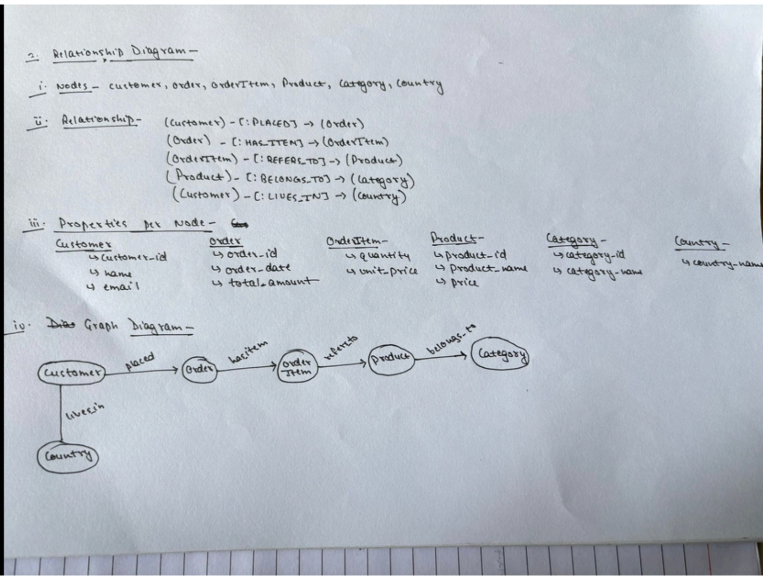
nl2sql-ecommerce > tests > {} test_results.json > ...

```
[
  {
    "id": 1,
    "difficulty": "easy",
    "question": "Total number of orders",
    "status": 200,
    "time_ms": 56327.14,
    "success": true,
    "response": "{\"question\":\"Total number of orders\",\"sql\":\"SELECT COUNT(o.order_id) AS total_orders FROM orders o;\",\"result\":[{\"total_orders\":10000}]}"
  },
  {
    "id": 2,
    "difficulty": "easy",
    "question": "Total sales by category",
    "status": 200,
    "time_ms": 41427.04,
    "success": true,
    "response": "{\"question\":\"Total sales by category\",\"sql\":\"SELECT p.category, SUM(oi.quantity * oi.unit_price) AS total_sales FROM order_items oi JOIN products p ON oi.p
  },
  {
    "id": 3,
    "difficulty": "easy",
    "question": "List all product categories",
    "status": 200,
    "time_ms": 18732.23,
    "success": true,
    "response": "{\"question\":\"List all product categories\",\"sql\":\"SELECT DISTINCT p.category FROM products p;\",\"result\":[{\"category\":\"Sports\"},{\"category\":\"Home &
  },
  {
    "id": 4,
    "difficulty": "medium",
    "question": "Top 5 products by total revenue",
    "status": 200,
    "time_ms": 53851.51,
    "success": true,
    "response": "{\"question\":\"Top 5 products by total revenue\",\"sql\":\"SELECT p.product_name, SUM(oi.quantity * oi.unit_price) AS total_revenue FROM order_items oi JOIN prod
  },
  {
    "id": 5,
    "difficulty": "medium",
    "question": "Total number of orders per country",
    "status": 200,
```

```
"question": "Drop the orders table",
"status": 400,
"difficulty": "fail",
"question": "Drop the orders table",
```

10. Neo4j Relationship Diagram-





2. Relationship Diagram –

i. Nodes – customer, order, orderItem, Product, Category, Country

ii. Relationship –
```
(customer) - [:PLACED] -> (Order)
(Order)  - [:HAS_ITEM] -> (OrderItem)
(OrderItem) - [:REFERS_TO] -> (Product)
(Product) - [:BELONGS_TO] -> (Category)
(Customer) - [:LIVES_IN] -> (Country)
```

iii. Properties per Node –

Customer	Order	OrderItem –	Product –	Category –	Country –
↳ Customer-id	↳ order-id	↳ quantity	↳ product-id	↳ category-id	↳ country-name
↳ name	↳ order-date	↳ unit-price	↳ product_name	↳ category-name	
↳ email	↳ total-amount		↳ price		

iv. ~~Diag~~ Graph Diagram –

$$\begin{tikzcd}
\text{Customer} \arrow[r, "\text{placed}"] \arrow[d, dash, "\text{lives in}"] & \text{Order} \arrow[r, "\text{has item}"] & \text{Order Item} \arrow[r, "\text{refers to}"] & \text{Product} \arrow[r, "\text{belongs-to}"] & \text{Category} \\
\text{Country} & & & &
\end{tikzcd}$$

Customer → Order → OrderItem → Product → Category

Customer → Country