

E-Commerce Admin Dashboard API

This is a FastAPI-powered backend for an e-commerce admin dashboard. The API allows e-commerce managers to view and manage sales data, inventory, and products, as well as track revenue over different periods.

Features

- **Sales Status:** Retrieve, filter, and analyze sales data. Compare revenue across different periods.
- **Inventory Management:** View and update current inventory levels, and track stock changes.
- **Product Registration:** Register new products in the system.
- **Revenue Analysis:** Get insights on revenue over daily, weekly, monthly, or annual periods.

Requirements

- Python 3.8+
- MySQL (or any compatible relational database)

Setup Instructions

1. Clone the Repository

Clone this repository to your local machine:

```
git clone .git
```

2. Create a Virtual Environment

Create and activate a virtual environment to manage dependencies:

```
# For Windows
python -m venv venv
venv\Scripts\activate
```

3. Install Dependencies

Install the required Python dependencies:

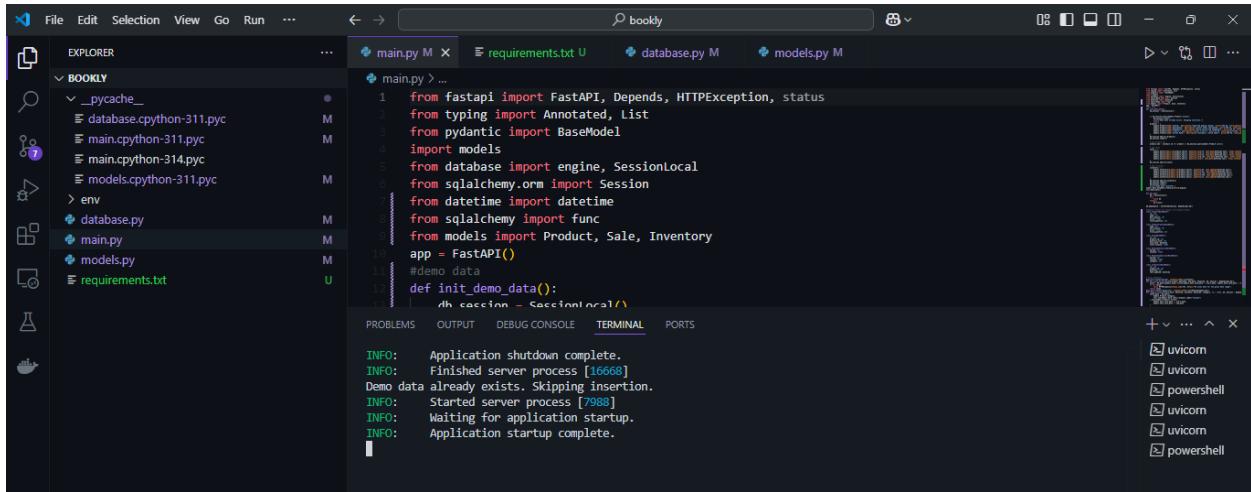
```
pip install -r requirements.txt
```

4. Start the API

Run the FastAPI application:

```
uvicorn main:app --reload
```

The API will be accessible at <http://localhost:8000>.



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the project structure under 'EXPLORER' with files like 'main.py', 'requirements.txt', 'database.py', 'models.py', and 'main.pyc'. The main editor area shows the 'main.py' file content. Below the editor is the 'TERMINAL' tab, which displays the command-line output of the application startup. The right sidebar shows a list of running processes.

```
INFO: Application shutdown complete.  
INFO: Finished server process [10668].  
Demo data already exists. Skipping insertion.  
INFO: Started server process [7988].  
INFO: Waiting for application startup.  
INFO: Application startup complete.
```

5. Access the API Documentation

Once the application is running, you can access the automatically generated documentation for the API:

- **Swagger UI:** <http://localhost:8000/docs>
- **ReDoc UI:** <http://localhost:8000/redoc>

Endpoints

1. Sales Status Endpoints

Get Sales by Product ID

- **Method:** GET
- **Endpoint:** /sales/{product_id}
- **Example:** /sales/1

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:8000/sales/1`
- Method:** `GET`
- Body:** Raw JSON (selected)

```
1 {  
2   "username": "aden"  
3 }  
4
```
- Response Status:** `200 OK`
- Response Time:** `102 ms`
- Response Size:** `228 B`
- Response Content:**

```
1 [  
2   {  
3     "id": 1,  
4     "product_id": 1,  
5     "quantity_sold": 10,  
6     "sale_date": "2025-05-14T19:38:14",  
7     "total_revenue": 12000.0  
8   }  
9 ]
```

Get Sales by Date Range

- **Method:** `GET`
- **Endpoint:** `/sales/period`
- **Query Parameters:**
 - `start_date`: (e.g. `2025-01-01T00:00:00`)
 - `end_date`: (e.g. `2025-12-31T23:59:59`)

The screenshot shows a Postman interface with the following details:

- Request URL:** `http://127.0.0.1:8000/sales/period?start_date=2025-01-01T00:00:00&end_date=2025-12-31T23:59:59`
- Method:** GET
- Body:** Raw JSON payload:

```
1 {
2   "username": "aden"
3 }
4
```
- Response Status:** 200 OK
- Response Headers:** (4 items)
- Response Body (JSON):**

```
1 [
2   {
3     "id": 1,
4     "product_id": 1,
5     "quantity_sold": 10,
6     "sale_date": "2025-05-14T19:38:14",
7     "total_revenue": 12000.0
8   },
9   {
10    "id": 2,
11    "product_id": 2,
12    "quantity_sold": 15,
13    "sale_date": "2025-05-14T19:38:14"
14 }
```

❖ Get Daily Revenue

- **Method:** GET
- **Endpoint:** /sales/revenue/daily

The screenshot shows a REST client interface with the following details:

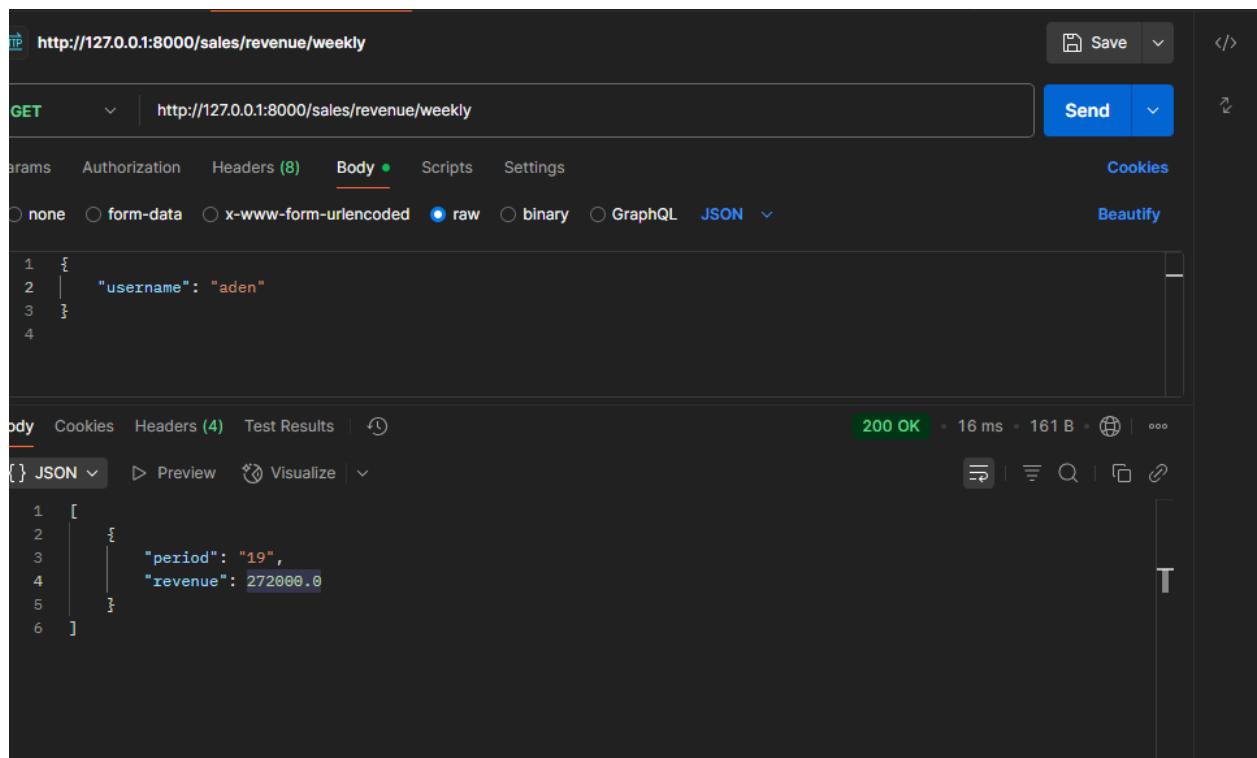
- URL:** `http://127.0.0.1:8000/sales/revenue/daily`
- Method:** GET
- Body:** Raw JSON (selected)

```
1 {  
2 |   "username": "aden"  
3 }  
4
```
- Response Status:** 200 OK
- Response Headers:** (4) - includes Content-Type: application/json, Content-Length: 212, Date: etc.
- Response Body:** JSON

```
1 [  
2 | {  
3 | |   "period": "2025-05-14",  
4 | |   "revenue": 136000.0  
5 | | },  
6 | {  
7 | |   "period": "2025-05-15",  
8 | |   "revenue": 136000.0  
9 | }  
10 ]
```

Get Weekly Revenue

- **Method:** GET
- **Endpoint:** /sales/revenue/weekly



The screenshot shows a Postman request to `http://127.0.0.1:8000/sales/revenue/weekly`. The request method is `GET` and the URL is `http://127.0.0.1:8000/sales/revenue/weekly`. The `Body` tab is selected, showing a raw JSON payload:

```
1 {  
2   "username": "aden"  
3 }  
4
```

The response status is `200 OK` with a response time of `16 ms` and a size of `161 B`. The response body is:

```
[{"period": "19", "revenue": 272000.0}]
```

Get Monthly Revenue

- **Method:** `GET`
- **Endpoint:** `/sales/revenue/monthly`

The screenshot shows a REST client interface with the following details:

- URL:** http://127.0.0.1:8000/sales/revenue/monthly
- Method:** GET
- Body:** Raw JSON (selected)

```
1 {  
2   "username": "aden"  
3 }  
4
```
- Headers:** (8)
- Test Results:** 200 OK, 13 ms, 160 B

The response body is displayed as:

```
[  
  {  
    "period": "5",  
    "revenue": 272000.0  
  }]
```

Get Annual Revenue

- **Method:** GET
- **Endpoint:** /sales/revenue/annual

The screenshot shows the Postman application interface. At the top, there are two tabs: 'POST http://localhost:9000/' and 'GET http://127.0.0.1:8000/sales/revenue/annual'. The second tab is selected. In the center, the URL 'http://127.0.0.1:8000/sales/revenue/annual' is displayed. To the right, there are 'Save' and 'Send' buttons. Below the URL, the method 'GET' is selected. The 'Body' tab is active, showing a raw JSON payload:

```
1 {
2   "username": "aden"
3 }
```

Below the body, the response status is '200 OK' with a response time of '11 ms' and a size of '163 B'. The response body is shown in JSON format:

```
1 [
2   {
3     "period": "2025",
4     "revenue": 272000.0
5   }
6 ]
```

Compare Revenue Method: GET

- **Endpoint:** /sales/comparison
- **Query Parameters:**
 - start_date: (e.g. 2025-01-01T00:00:00)
 - end_date: (e.g. 2025-12-31T23:59:59)
 - category:

The screenshot shows the Postman application interface. At the top, there are three tabs: 'POST http://localhost:9000/t' (red), 'GET http://127.0.0.1:8000/GE' (green), and 'GET http://127.0.0.1:8000/sa' (orange). A new tab button '+' is at the top right. The main area has a title 'HTTP http://127.0.0.1:8000/sales/comparison?start_date=2025-01-01T00:00:00&end_date=2025-12-31T23:59:59&categ...' with a 'Save' and 'Share' button. Below it, a 'GET' request is selected, and the URL is shown again. To the right is a 'Send' button. Under 'Params', four items are listed: 'start_date' (2025-01-01T00:00:00), 'end_date' (2025-12-31T23:59:59), 'category' (Accessories), and 'Key'. There are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Test Results' tab shows a green '200 OK' status with 10 ms response time and 207 B size. Below this, a JSON response is displayed:

```
1 [  
2 {  
3   "period": "2025-01-01 to 2025-12-31",  
4   "revenue": 90000.0,  
5   "category": "Accessories"  
6 }
```

2. Inventory Management Endpoints

❖ Get Inventory for a Product

- **Method:** GET
- **Endpoint:** /inventory/{product_id}
- **Example:** /inventory/1

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:8000/inventory/1`
- Method:** GET
- Headers:** (6)
- Body:** (Empty)
- Cookies:** (Empty)
- Query Params:** (Empty)
- Response Status:** 200 OK
- Response Time:** 9 ms
- Response Size:** 199 B
- Response Body:**

```
1 {  
2     "id": 1,  
3     "product_id": 1,  
4     "quantity": 40,  
5     "last_updated": "2025-05-14T19:38:14"  
6 }
```

View All Inventory Status

- **Method:** GET
- **Endpoint:** /inventory/status

The screenshot shows a Postman interface with the following details:

- URL:** `http://127.0.0.1:8000/inventory/status`
- Method:** GET
- Headers:** (6)
- Body:** (JSON array with 2 items)
- Response Status:** 200 OK
- Latency:** 33 ms
- Size:** 6.15 KB

The JSON response body is:

```
1 [  
2 {  
3     "id": 1,  
4     "product_id": 1,  
5     "quantity": 40,  
6     "last_updated": "2025-05-14T19:38:14"  
7 },  
8 {  
9     "id": 2,  
10    "product_id": 2,  
11    "quantity": 85,  
12    "last_updated": "2025-05-14T19:38:14"  
13 }]
```

Update Inventory Quantity

- **Method:** POST
- **Endpoint:** /inventory/{product_id}/update
- **Query Parameters:**
 - quantity: new quantity value
- **Example**

The screenshot shows a Postman request to `http://127.0.0.1:8000/inventory/1/update?quantity=35`. The method is set to `POST`. In the `Query Params` section, there is a checked checkbox for `Key` and another checked checkbox for `quantity` with a value of `35`. The `Body` tab shows a JSON response with the following structure:

```
1 {  
2   "id": 1,  
3   "product_id": 1,  
4   "quantity": 35,  
5   "last_updated": "2025-05-14T19:38:14"  
6 }
```

The response status is `200 OK` with a duration of `32 ms`, a size of `199 B`, and a timestamp of `2025-05-14T19:38:14`.

Get Low Stock Alerts

- **Method:** `GET`
- **Endpoint:** `/inventory/low-stock`

The screenshot shows a Postman request to `http://127.0.0.1:8000/inventory/low-stock`. The method is set to `GET`. The `Body` tab shows a JSON response with the following structure:

```
1 [ ]
```

The response status is `200 OK` with a duration of `16 ms`, a size of `126 B`, and a timestamp of `2025-05-14T19:38:14`.

3. Product Endpoints

⚡ Get All Products

- **Method:** GET
- **Endpoint:** /products

The screenshot shows the Postman interface with a successful API call to the '/products' endpoint. The response body is a JSON array containing two product objects.

```
[{"id": 1, "name": "Laptop", "description": "High-end gaming laptop", "price": 1200.0, "stock_quantity": 50}, {"id": 2, "name": "Smartphone", "description": "Latest model smartphone", "price": 800.0, "stock_quantity": 100}]
```

Register New Product

- **Method:** POST
- **Endpoint:** /products

HTTP <http://127.0.0.1:8000/products>

POST <http://127.0.0.1:8000/products>

Send

Params Authorization Headers (8) Body Scripts Settings Cookies

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

```
2 "name": "Tablet",
3 "description": "10-inch Android tablet",
4 "price": 300.00,
5 "stock_quantity": 60
6 , "category": "Electronics"
7 }
```

Body Cookies Headers (4) Test Results

200 OK 53 ms 222 B

{ } JSON Preview Visualize

```
1 {
2   "id": 7,
3   "name": "Tablet",
4   "description": "10-inch Android tablet",
5   "price": 300.0,
6   "stock_quantity": 60
7 }
```