EXERCISE 2 FOR UZAIF

(DJANGO REST FRAMEWORK)

API DEVELOPMENT

*The brand below is for test purpose only
*You have to use DRF for making the below API:

<Back-end Development Task: E-commerce Admin API>

The objective of this task is to design and implement a back-end API that can power a web admin dashboard for e-commerce managers. This API should provide detailed insights into sales, revenue, and inventory status, as well as allow new product registration. The implementation should be done using Python and DRF.

Core Features:

1. **Sales Status:**
    - Endpoints to retrieve, filter, and analyze sales data.
    - Endpoints to analyze revenue on a daily, weekly, monthly, and annual basis.
    - Ability to compare revenue across different periods and categories.
    - Provide sales data by date range, product, and category.

2. **Inventory Management:**
    - Endpoints to view current inventory status, including low stock alerts.
    - Functionality to update inventory levels, and track changes over time.

Endpoints documentation:

1. Create Sale Record

- **Endpoint:** /sales/create
- **Method:** POST
- **Request Body:**
  - product_id (int)
  - quantity_sold (int)
  - total_amount (float)
  - sale_date (date) (Format: "YYYY-MM-DD")
- **Response Body:**
  - product_id (int)
  - quantity_sold (int)
  - total_amount (float)
  - sale_date (date) (Format: "YYYY-MM-DD")

---

2. Get Revenue within Date Range

- **Endpoint:** /sales/revenue
- **Method:** GET
- **Query Parameters:**
  - start_date (str) (Format: "YYYY-MM-DD")
  - end_date (str) (Format: "YYYY-MM-DD")
- **Response Body:**
  - revenue (float)

---

3. Get Current Inventory

- **Endpoint:** /inventory
- **Method:** GET
- **Query Parameter:**
  - low_stock_threshold (int, default=10, description="Low stock threshold")
- **Response Body:**
  - inventory (list)
  - low_stock_alerts (list) - if any products are below the low stock threshold

---

4. Update Inventory

- **Endpoint:** /inventory/update
- **Method:** PUT
- **Request Body:**
  - product_id (int)
  - quantity (int)
- **Response Body:**
  - message (str)

---

5. Register Product

- **Endpoint:** /products/register
- **Method:** POST
- **Request Body:**
  - product_name (str)
  - price (float)
- **Response Body:**
  - product_id (int)
  - product_name (str)
  - price (float)

---

6. Get All Sales

- **Endpoint:** /sales
- **Method:** GET
- **Response Body:** List of sale records

---

7. Get Daily Sales

- **Endpoint:** /sales/daily
- **Method:** GET
- **Response Body:** List of sale records for the current day

---

8. Get Weekly Sales

- **Endpoint:** /sales/weekly
- **Method:** GET
- **Response Body:** List of sale records for the current week

---

9. Get Monthly Sales

- **Endpoint:** /sales/monthly
- **Method:** GET
- **Response Body:** List of sale records for the current month

---

10. Get Annual Sales

- **Endpoint:** /sales/annual
- **Method:** GET
- **Response Body:** List of sale records for the current year

---

11. Filter Sales

- **Endpoint:** /sales/filter
- **Method:** GET
- **Query Parameters:**
  - start_date (optional) (Format: "YYYY-MM-DD")
  - end_date (optional) (Format: "YYYY-MM-DD")
  - product_id (optional)
  - category (optional)
  - quantity_sold_min (optional)
  - quantity_sold_max (optional)
  - total_amount_min (optional)
  - total_amount_max (optional)
- **Response Body:** List of filtered sale records

---

12. Analyze Sales

- **Endpoint:** /sales/analysis
- **Method:** GET
- **Query Parameters:**
  - (Same as "Filter Sales")
- **Response Body:**
  - total_quantity_sold (int)
  - total_revenue (float)

---

13. Compare Sales

- **Endpoint:** /sales/compare
- **Method:** GET
- **Query Parameters:**
  - start_date_1 (str) (Format: "YYYY-MM-DD")
  - end_date_1 (str) (Format: "YYYY-MM-DD")
  - start_date_2 (str) (Format: "YYYY-MM-DD")
  - end_date_2 (str) (Format: "YYYY-MM-DD")
- **Response Body:**
  - revenue_comparison (dict) with details for both date ranges

---

14. Sales by Date Range

- **Endpoint:** /sales/bydate
- **Method:** GET
- **Query Parameters:**
  - start_date (str) (Format: "YYYY-MM-DD")
  - end_date (str) (Format: "YYYY-MM-DD")
- **Response Body:** List of sale records within the specified date range

---

15. Sales by Product

- **Endpoint:** /sales/byproduct
- **Method:** GET
- **Query Parameters:**
  - product_id (int)
- **Response Body:** List of sale records for a specific product

---

16. Sales by Category

- **Endpoint:** /sales/bycategory
- **Method:** GET
- **Query Parameters:**
  - category (str)
- **Response Body:** List of sale records for a specific product category

Technical Requirements:

1. **API Development:**
   - Design and implement API endpoints using Python and DRF to handle operations like retrieving sales data, analyzing revenue, managing inventory, and registering new products.

2. **Database Modeling and Design:**
    - Design a database schema to support the required functionalities.
    - Implement the database using a relational database management system like PSQL.

Database Specifics:

- The database should have tables for products, sales, inventory, and other relevant entities.
- Ensure proper indexing for optimized query performance.
- Ensure the database design supports the requirements of the API and is normalized to prevent redundancy and maintain consistency.

Demo Data:

- Provide a script to populate the database with demo data to help evaluate the functionality of the API.
- You may use sample data (of your own) regarding sales and inventory for the products sold on Amazon & Walmart.

Submission Instructions:

1. **Postman Collection**
    - Provide Postman Collection for Testing

2. **Database Documentation:**
    - Document the database schema, explaining the purpose of each table and its relationships.