CSV:

```
transaction_id,date,category,product,quantity,price
1,2024-07-01,Widget,Widget-A,10,9.99
2,2024-07-01,Gadget,Gadget-X,5,19.99
3,2024-07-02,Widget,Widget-B,7,9.99
4,2024-07-02,Doodad,Doodad-1,,4.99
5,2024-07-03,Widget,Widget-C,3,9.99
6,2024-07-03,Gadget,Gadget-Y,8,19.99
7,2024-07-04,Widget,Widget-A,2,9.99
8,2024-07-04,Doodad,Doodad-2,4,not_a_number
9,2024-07-05,Widget,Widget-B,6,9.99
10,2024-07-05,Gadget,Gadget-X,3,19.99
11,2024-07-06,Gadget,,5,19.99
12,2024-07-06,Doodad,Doodad-3,1,4.99
13,2024-07-07,Widget,Widget-C,8,9.99
14,2024-07-07,Gadget,Gadget-Y,4,19.99
15,2024-07-08,Widget,Widget-A,3,9.99
16,2024-07-08,Doodad,Doodad-1,2,4.99
17,2024-07-09,Gadget,Gadget-X,,19.99
18,2024-07-09,Widget,Widget-B,5,9.99
19,2024-07-10,Doodad,Doodad-2,4,4.99
20,2024-07-10,Gadget,Gadget-Y,7,19.99
21,2024-07-11,Widget,Widget-C,6,9.99
22,2024-07-11,Doodad,Doodad-3,3,4.99
23,2024-07-12,Gadget,Gadget-X,9,19.99
24,2024-07-12,Widget,Widget-A,1,9.99
25,2024-07-13,Doodad,Doodad-1,2,4.99
26,2024-07-13,Gadget,Gadget-Y,5,19.99
27,2024-07-14,Widget,Widget-B,,9.99
28,2024-07-14,Doodad,Doodad-2,4,4.99
29,2024-07-15,Gadget,Gadget-X,7,19.99
30,2024-07-15,Widget,Widget-C,3,9.99
31,2024-07-16,Doodad,Doodad-3,1,4.99
32,2024-07-16,Gadget,Gadget-Y,5,not_a_number
33,2024-07-17,Widget,Widget-A,8,9.99
34,2024-07-17,Doodad,Doodad-1,2,4.99
35,2024-07-18,Gadget,Gadget-X,6,19.99
36,2024-07-18,Widget,Widget-B,4,9.99
37,2024-07-19,Doodad,Doodad-2,3,4.99
38,2024-07-19,Gadget,Gadget-Y,2,19.99
39,2024-07-20,Widget,Widget-C,5,9.99
40,2024-07-20,Doodad,Doodad-3,,4.99
41,2024-07-21,Gadget,Gadget-X,7,19.99
42,2024-07-21,Widget,Widget-A,3,9.99
```

```
43,2024-07-22,Doodad,Doodad-1,2,4.99
44,2024-07-22,Gadget,Gadget-Y,6,19.99
45,2024-07-23,Widget,Widget-B,7,9.99
46,2024-07-23,Doodad,Doodad-2,3,4.99
47,2024-07-24,Gadget,Gadget-X,,19.99
48,2024-07-24,Widget,Widget-C,5,9.99
49,2024-07-25,Doodad,Doodad-3,4,4.99
50,2024-07-25,Gadget,Gadget-Y,8,19.99
```

Homework Assignment: Sales Dashboard Application

Objective:

The goal is to create a simple sales dashboard application. The candidate will need to ingest and process sales data, build an API to serve the processed data

Problem Statement:

You are tasked with building a small web application that displays a sales summary for an e-commerce platform.

Detailed Instructions:

Task:

Process the provided CSV file, which contains raw sales data with **data irregularities**. Your goal is to clean, transform, and prepare the data for efficient querying.

Challenges:

1. **Data Cleaning**:
   - Handle missing values:
     - Replace missing quantity with 0.
     - Replace missing or invalid price values (not_a_number) with the **median price** for the same product category.
   - Drop rows where both quantity and price are invalid or missing.
2. **Derived Columns**:
   - Calculate total_sales (quantity * price) for each transaction.

- Create a day_of_week column based on the date.
- Add a high_volume flag: True if quantity > 10, otherwise False.

3. **Complex Transformations**:

- Group data by category and calculate:
  - **Average price** per product in the category.
  - **Total revenue** for each category.
  - **Day with highest sales** for the category.
- Identify **outliers** in the data:
  - Transactions where quantity is more than 2 standard deviations from the category mean.
  - Mark these rows with an outlier flag.

4. **Storage**:

- Store the cleaned and processed data in a SQLite database with separate tables for:
  - Transactions
  - Aggregated metrics by category
  - Outliers

Tools:

Use Python with libraries like Pandas, SQLAlchemy, or similar.

Output:

1. A SQLite database containing:

- Cleaned and processed data.
- Aggregated metrics for efficient querying.
- Outliers flagged and stored in a separate table.

2. A Python script demonstrating:

- Data cleaning steps.
- SQL queries for fetching aggregated metrics.

Part 2: Backend

Create a RESTful API to expose the processed data.

Endpoints:

1. **GET /sales/product**:
   - Returns total sales for each product.
   - Supports optional filtering by product name or category.
2. **GET /sales/day**:
   - Returns total sales for each day.
   - Supports filtering by date range.
3. **GET /sales/category**:
   - Returns aggregated metrics for each category (e.g., total revenue, average price, day with highest sales).
4. **GET /sales/outliers**:
   - Returns flagged outlier transactions.

Deliverables

- A Python script for data processing. Feel free to use any tool you are used to.
- A Flask or FastAPI backend.

Estimated time: 24 hours.
Expected output: Git repository