TASK: GET BASIC SALES SUMMARY FROM A TINY SQLite DATABASE USING PYTHON

1. Objective

The goal of this project is to:

- Store sales data in a small SQLite database (sales_data.db).
- Run basic SQL queries inside Python.
- Generate a sales summary (total quantity & revenue by product).
- Visualize results using a simple bar chart.

Tools used:

- **Python 3** (with sqlite3, pandas, matplotlib)
- **SQLite** (lightweight DB, built into Python)

2. Import libraries

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
```

3. Create Database & Table

```
# Connect to SQLite DB (creates sales_data.db if not exists)
conn = sqlite3.connect("sales_data.db")
cursor = conn.cursor()

# Create sales table
cursor.execute('''
CREATE TABLE IF NOT EXISTS sales (
   id INTEGER PRIMARY KEY,
   product_name TEXT,
   quantity INTEGER,
   price REAL
)
''')
conn.commit()
```

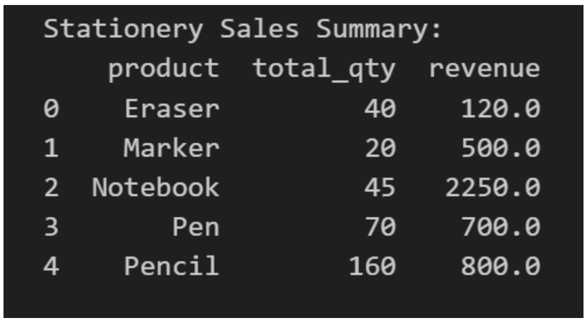
```
# Insert stationery data if table is empty
cursor.execute("SELECT COUNT(*) FROM sales")
if cursor.fetchone()[0] == 0:
    sample_data = [
        ("Pen", 50, 10.0),
        ("Pencil", 100, 5.0),
        ("Notebook", 30, 50.0),
        ("Eraser", 40, 3.0),
        ("Marker", 20, 25.0),
        ("Pen", 20, 10.0),
        ("Notebook", 15, 50.0),
        ("Pencil", 60, 5.0)
    ]
    cursor.executemany("INSERT INTO sales (product_name, quantity, price)
VALUES (?, ?, ?)", sample_data)
    conn.commit()
```

At this point, sales_data.db contains a table sales with some demo records.

4. Run SQL Query

```
query = """
SELECT product_name AS product,
       SUM(quantity) AS total_qty,
       SUM(quantity * price) AS revenue
FROM sales
GROUP BY product_name
"""
df = pd.read_sql_query(query, conn)
print("Stationery Sales Summary:")
print(df)
```

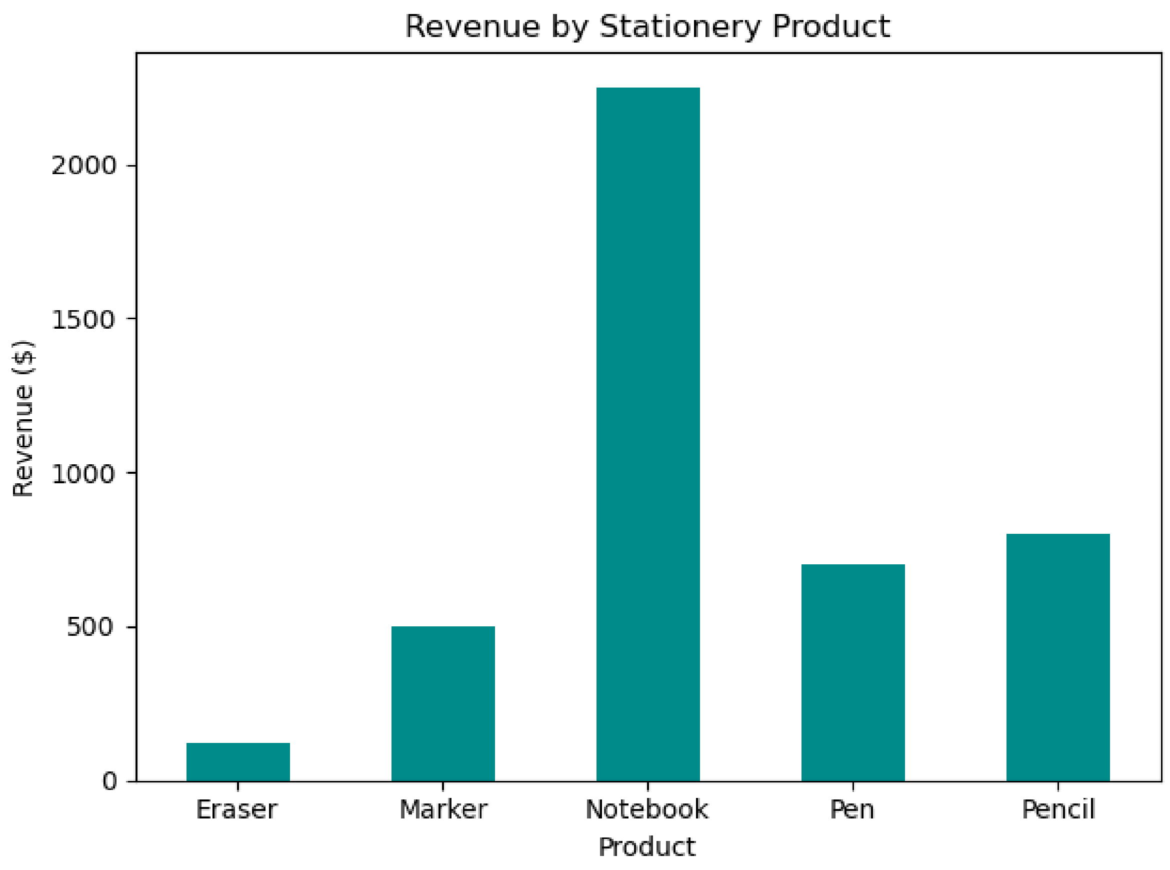
Output:



```
Stationery Sales Summary:
    product  total_qty  revenue
0    Eraser         40    120.0
1    Marker         20    500.0
2  Notebook         45   2250.0
3       Pen         70    700.0
4    Pencil        160    800.0
```

5. Visualization – Revenue by Product

```
df.plot(kind='bar', x='product', y='revenue', legend=False, color="darkcyan")
plt.title("Revenue by Stationery Product")
plt.xlabel("Product")
plt.ylabel("Revenue ($)")
plt.xticks(rotation=0)
plt.tight_layout()
plt.savefig("stationery_sales_chart.png")
plt.show()
```



6.Findings:

1. Notebook generated the highest revenue (₹2250) even though only 45 units were sold.
2. Pencil sold the largest quantity (160 units), but its low price per unit (₹5) resulted in only ₹800 revenue.
3. Pen had a decent balance: 70 units sold for a total of ₹700.
4. Marker sold fewer units (20) but still produced a significant ₹500 revenue because of its higher price (₹25 each).
5. Eraser had the lowest revenue contribution (₹120), reflecting both low sales and low price.

7.Insights:

- High-value products like Notebooks dominate revenue despite fewer sales.
- Low-cost consumables like Pencils drive volume but do not contribute much to revenue unless sold in massive quantities.
- Mid-range items like Pens and Markers provide a stable balance between sales volume and revenue.
- The product mix shows a classic sales pattern: a few premium products driving most revenue, while cheaper items maintain customer engagement and repeat sales.

8.Conclusion:

- For revenue growth, focusing on premium products (e.g., Notebooks, Markers) is crucial.
- For market penetration and steady demand, low-cost items (e.g., Pencils, Erasers) remain important.
- An optimized sales strategy should ensure availability of both premium and low-cost stationery items.
- Promotions or bundles (e.g., Notebooks + Pens) could maximize overall revenue while sustaining customer loyalty.