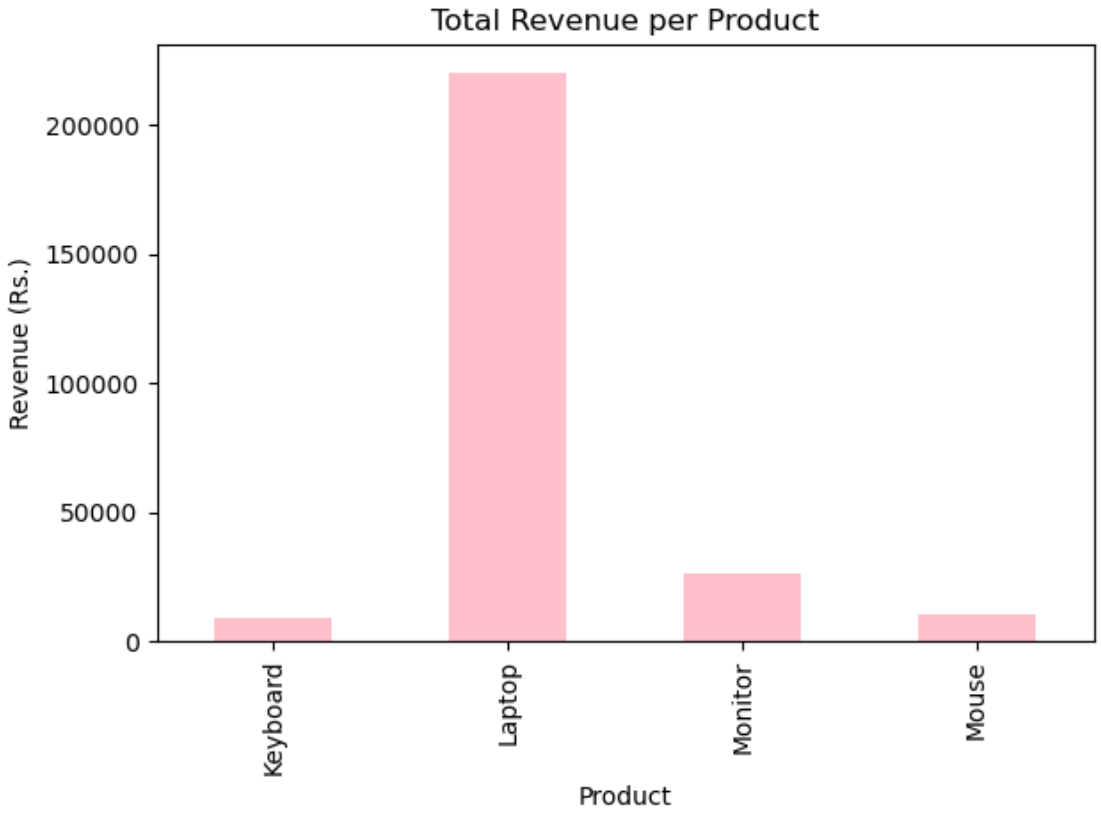
```python
import sqlite3

# Connect to database (will create if doesn't exist)
conn = sqlite3.connect("sales_data.db")
cursor = conn.cursor()

# Drop table if exists for reruns
cursor.execute("DROP TABLE IF EXISTS sales")

# Create table
cursor.execute("""
CREATE TABLE sales (
    id INTEGER PRIMARY KEY,
    date TEXT,
    product TEXT,
    quantity INTEGER,
    price REAL
)
""")

# Insert sample data
sales_data = [
    (1, '2025-06-01', 'Laptop', 2, 55000.00),
    (2, '2025-06-01', 'Mouse', 5, 500.00),
    (3, '2025-06-02', 'Keyboard', 3, 1200.00),
    (4, '2025-06-03', 'Monitor', 1, 9000.00),
    (5, '2025-06-03', 'Laptop', 1, 56000.00),
    (6, '2025-06-04', 'Mouse', 10, 450.00),
    (7, '2025-06-05', 'Monitor', 2, 8700.00),
    (8, '2025-06-05', 'Keyboard', 4, 1300.00),
    (9, '2025-06-06', 'Laptop', 1, 54000.00),
    (10, '2025-06-07', 'Mouse', 8, 475.00),
]

cursor.executemany("INSERT INTO sales VALUES (?, ?, ?, ?, ?)", sales_data)
conn.commit()
conn.close()
```

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

# Step 1: Connect to SQLite database
conn = sqlite3.connect("sales_data.db")

# Step 2: Define and run SQL query
query = """
SELECT
    product,
```

```
    SUM(quantity) AS total_quantity_sold,
    SUM(quantity * price) AS total_revenue
FROM
    sales
GROUP BY
    product
"""
df = pd.read_sql_query(query, conn)

# Step 3: Close the connection
conn.close()

# Step 4: Print DataFrame
print("Sales Summary:")
print(df)

# Step 5: Plot bar chart of revenue per product
plt.figure(figsize=(8, 5))
df.plot(kind='bar', x='product', y='total_revenue', legend=False,
color='pink')
plt.title("Total Revenue per Product")
plt.ylabel("Revenue (Rs.)")
plt.xlabel("Product")
plt.tight_layout()

# Optional: Save the chart
plt.savefig("sales_chart.png")

# Show plot
plt.show()
```

```
Sales Summary:
    product  total_quantity_sold  total_revenue
0  Keyboard                    7         8800.0
1    Laptop                    4       220000.0
2   Monitor                    3        26400.0
3     Mouse                   23        10800.0
```



SQLite Sales Summary and Visualization using Python

- Objective: To build a mini analytics tool using SQLite, Python, and Matplotlib.
- Highlights: End-to-end integration of SQL with Python for data analysis and visualization.

Objective

- The primary goal is to analyze sales data stored in a local SQLite database and summarize key metrics such as:
    - Total quantity sold per product.
    - Total revenue generated per product.
- The results are visualized using bar charts for better interpretation.
- This task is ideal for understanding how databases and data visualization libraries work together.

Tools and Technologies Used

- **SQLite:** A lightweight, serverless database engine used for storing sales data.
- **Python 3:** Programming language for scripting and data processing.
- **Pandas:** Library for data manipulation and analysis.
- **Matplotlib:** Visualization library used for plotting bar charts.
- **Jupyter Notebook:** Execution environment for the project.

Step 1 - Database Setup

- A new SQLite database file sales_data.db is created using Python's sqlite3 module.
- A single table sales is created with columns:
    - id: Primary Key
    - date: Date of sale
    - product: Product name
    - quantity: Number of units sold
    - price: Unit price of the product
- Sample data of 10 transactions covering multiple products (Laptop, Mouse, Keyboard, Monitor) is inserted.

```
CREATE TABLE sales (
    id INTEGER PRIMARY KEY,
    date TEXT,
    product TEXT,
    quantity INTEGER,
    price REAL
)
```

Step 2 - Writing the SQL Query

- The SQL query retrieves:
    - Total quantity sold for each product.
    - Total revenue, calculated as SUM(quantity * price) for each product.
- The GROUP BY clause is used to group the data based on product name.

```
SELECT
    product,
    SUM(quantity) AS total_quantity_sold,
    SUM(quantity * price) AS total_revenue
FROM
    sales
GROUP BY
    product
```

Step 3 - Query Execution and Data Loading

- Using pandas.read_sql_query, the SQL query is executed and the result is loaded into a DataFrame.
- This makes it easy to manipulate or visualize data using pandas.

python

```
df = pd.read_sql_query(query, conn)
```

Step 4 - Displaying Results

- The final summary includes:
    - Each product's name.
    - Total units sold.
    - Total revenue generated.
- Example output (DataFrame):

```
  Sales Summary:
    product  total_quantity_sold  total_revenue
0  Keyboard                    7         8800.0
1    Laptop                    4       220000.0
2   Monitor                    3        26400.0
3     Mouse                   23        10800.0
```

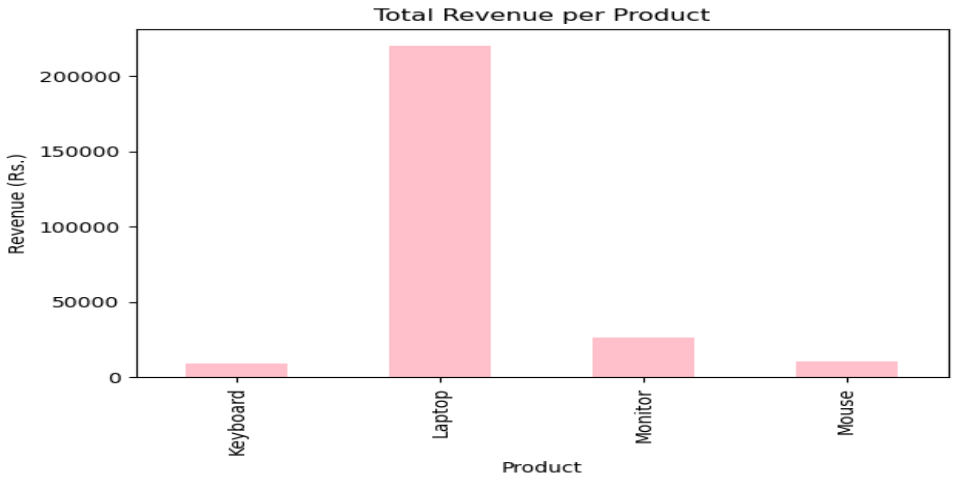
Step 5 - Visualizing Revenue

- A bar chart is plotted using matplotlib.pyplot.

- x-axis: Product names
- y-axis: Total revenue
- Aesthetics:
    - Pink bars for clarity.
    - Titles and axis labels added.
    - Figure saved as sales_chart.png for future use or reporting.

```
df.plot(kind='bar', x='product', y='total_revenue', legend=False, color='pink')
```

Chart Output



- The chart gives a quick visual comparison of which products generate the most revenue.
- For example, Laptops have the highest revenue, while the Mouse—despite high units sold—has relatively lower revenue.

Key Insights

- Laptops generate the highest revenue even though only 4 units were sold.
- Mice have the highest quantity sold but contribute less to overall revenue due to low unit price.
- Visual representation helps identify high-value and high-volume products easily.
- Demonstrates the power of combining SQL with Python for business analytics.