


```
import sqlite3, os, pandas as pd
import matplotlib.pyplot as plt
print("Libraries imported.")
```

 Libraries imported.

```
from google.colab import files
uploaded = files.upload()
csv_path = list(uploaded.keys())[0] if uploaded else None
```

  Online Sales Data.csv

- **Online Sales Data.csv**(text/csv) - 21746 bytes, last modified: 8/12/2025 - 100% done  
Saving Online Sales Data.csv to Online Sales Data.csv

```
import pandas as pd
import numpy as np
import os
```

```
def find_col(cols, *candidates):
    cols_lower = {c.lower(): c for c in cols}
    for cand in candidates:
        for c in cols:
            if cand in c.lower():
                return c
    return None
```

```
import pandas as pd
import numpy as np
import os
```

```
if csv_path and os.path.exists(csv_path):
    df_raw = pd.read_csv(csv_path)
else:
    rng = np.random.default_rng(42)
    products = ["Phone Case", "USB Cable", "Headphones", "Charger", "Power Bank"]
    rows = []
    for i in range(200):
        prod = rng.choice(products)
        qty = int(rng.integers(1, 6))
        price = float(rng.choice([199.0, 299.0, 399.0, 499.0, 999.0]))
        rows.append({"Product": prod, "Quantity": qty, "Price Each": price})
    df_raw = pd.DataFrame(rows)
```

```

cols = list(df_raw.columns)

product_col = find_col(cols, "product", "item", "product name", "sku")
quantity_col = find_col(cols, "quantity", "qty", "units")
price_col = find_col(cols, "price each", "unit price", "price", "selling price", "mrp")
sales_col = find_col(cols, "sales", "revenue", "amount")

df = pd.DataFrame()

if product_col:
    df["product"] = df_raw[product_col].astype(str)
else:
    df["product"] = "Unknown"

if quantity_col:
    df["quantity"] = pd.to_numeric(df_raw[quantity_col], errors="coerce").fillna(0).astype(int)
elif sales_col and price_col:
    approx_qty = pd.to_numeric(df_raw[sales_col], errors="coerce") / pd.to_numeric(df_raw[price_col], errors="coerce")
    df["quantity"] = approx_qty.fillna(1).round().clip(lower=1).astype(int)
else:
    df["quantity"] = 1

if price_col:
    df["price"] = pd.to_numeric(df_raw[price_col], errors="coerce").fillna(0.0).astype(float)
elif sales_col and quantity_col:
    approx_price = pd.to_numeric(df_raw[sales_col], errors="coerce") / pd.to_numeric(df_raw[quantity_col], errors="coerce")
    df["price"] = approx_price.fillna(approx_price.median() if not np.isnan(approx_price) else 0.0)
elif sales_col:
    df["price"] = pd.to_numeric(df_raw[sales_col], errors="coerce").fillna(0.0).astype(float)
else:
    df["price"] = 1.0

df["quantity"] = df["quantity"].clip(lower=0)
df["price"] = df["price"].clip(lower=0.0)

df = df[df["product"].astype(str).str.strip().ne("")]

df.head()

```



	product	quantity	price
0	Electronics	2	999.99
1	Home Appliances	1	499.99
2	Clothing	3	69.99
3	Books	4	15.99
4	Beauty Products	1	89.99



Next steps:

[Generate code with df](#)[View recommended plots](#)[New interactive sheet](#)

```

db_path = "sales_data.db"
if os.path.exists(db_path):
    os.remove(db_path)

conn = sqlite3.connect(db_path)
cur = conn.cursor()
cur.execute("""
CREATE TABLE IF NOT EXISTS sales (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    product TEXT,
    quantity INTEGER,
    price REAL
)
""")
conn.commit()

df[["product", "quantity", "price"]].to_sql("sales", conn, if_exists="append", index=False)
print(f"Inserted {len(df)} rows into 'sales'.")

```



Inserted 240 rows into 'sales'.

```

query = """
SELECT
    product,
    SUM(quantity) AS total_qty,
    SUM(quantity * price) AS revenue
FROM sales
GROUP BY product
ORDER BY revenue DESC
"""

df_summary = pd.read_sql_query(query, conn)
df_summary.to_csv("sales_summary.csv", index=False)
df_summary.head(10)

```



	product	total_qty	revenue
0	Electronics	66	34982.41
1	Home Appliances	59	18646.16
2	Sports	88	14326.52
3	Clothing	145	8128.93
4	Beauty Products	46	2621.90
5	Books	114	1861.93



Next steps:

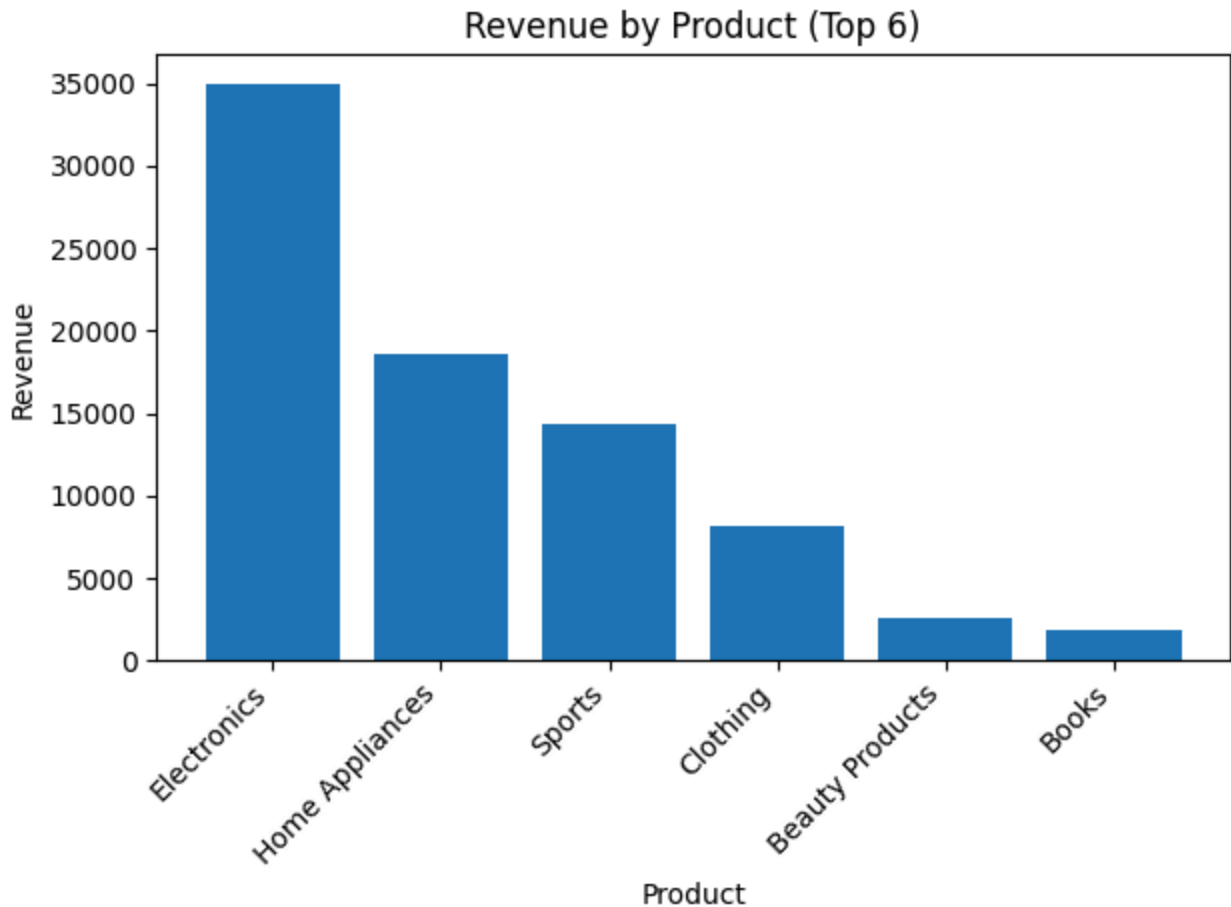
[Generate code with df\\_summary](#)[View recommended plots](#)[New interactive sheet](#)

```

topN = min(10, len(df_summary))
to_plot = df_summary.head(topN)

plt.figure()
plt.bar(to_plot["product"].astype(str), to_plot["revenue"])
plt.xticks(rotation=45, ha="right")
plt.title(f"Revenue by Product (Top {topN})")
plt.xlabel("Product")
plt.ylabel("Revenue")
plt.tight_layout()
plt.savefig("sales_revenue_bar.png")
plt.show()

```



```
conn.close()
print("Saved: sales_data.db, sales_summary.csv, sales_revenue_bar.png")

from google.colab import files
for fname in ["sales_data.db", "sales_summary.csv", "sales_revenue_bar.png"]:
    files.download(fname)
```



Saved: sales\_data.db, sales\_summary.csv, sales\_revenue\_bar.png

```
import seaborn as sns
sns.set(style="whitegrid")

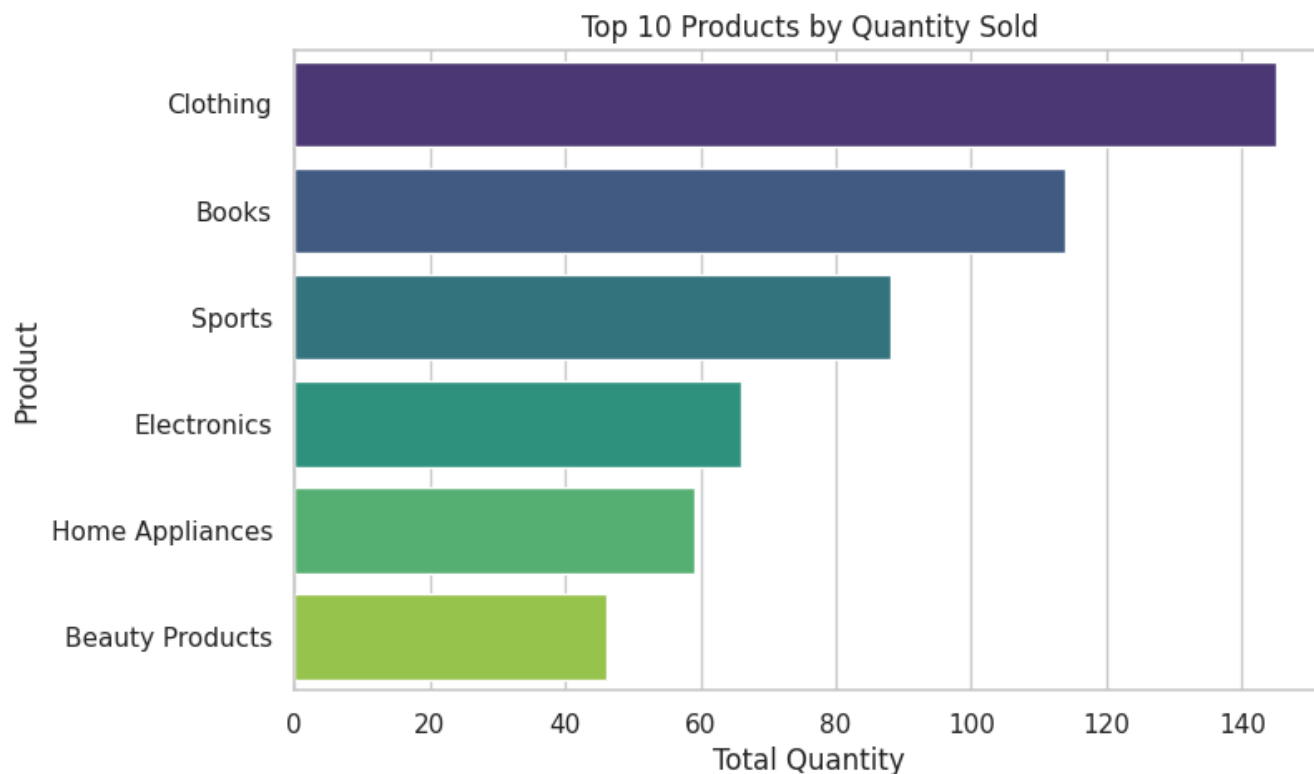
top_qty = df_summary.sort_values("total_qty", ascending=False).head(10)

plt.figure(figsize=(8,5))
sns.barplot(x="total_qty", y="product", data=top_qty, palette="viridis")
plt.title("Top 10 Products by Quantity Sold")
plt.xlabel("Total Quantity")
plt.ylabel("Product")
plt.show()
```

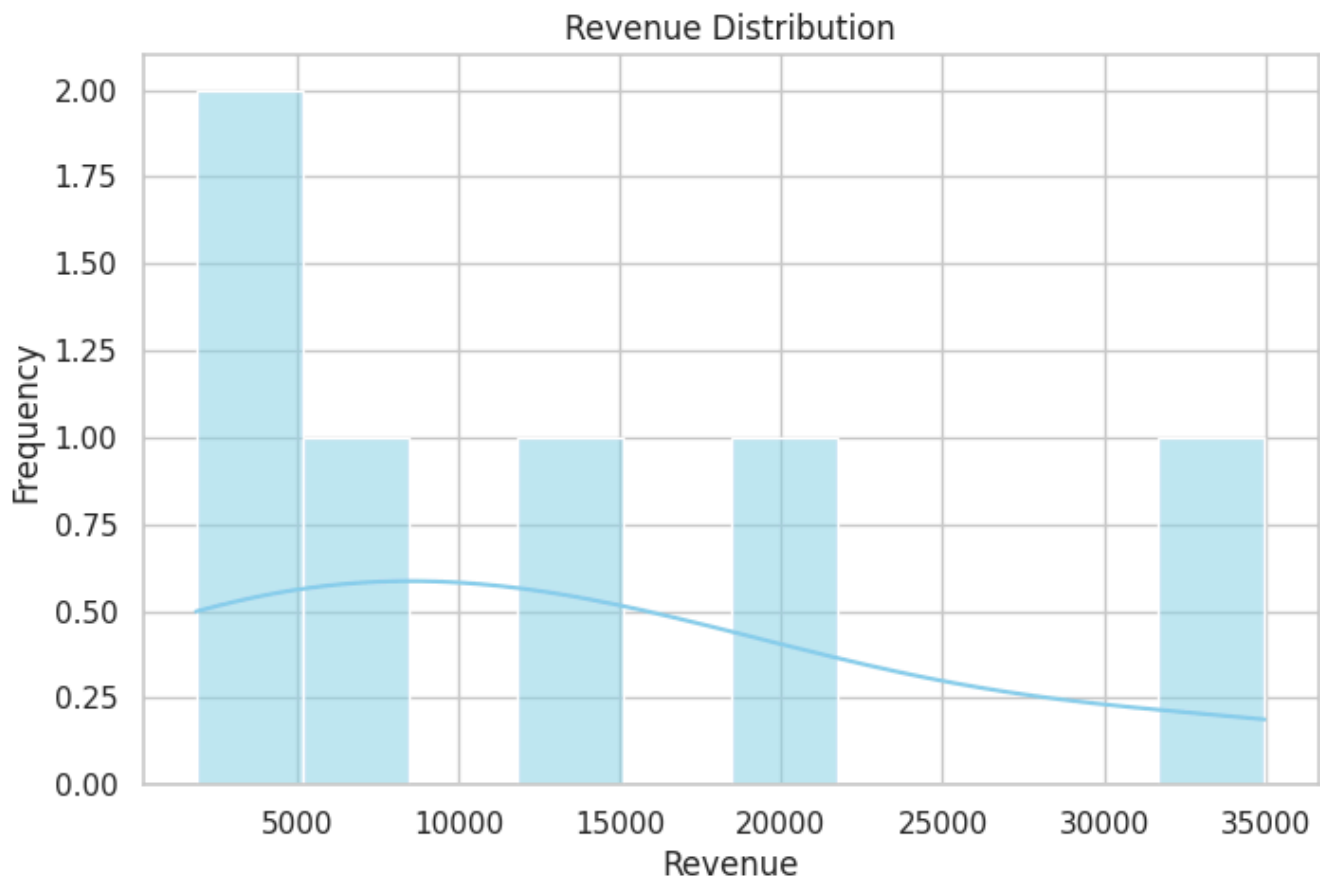
↗ /tmp/ipython-input-1648916026.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

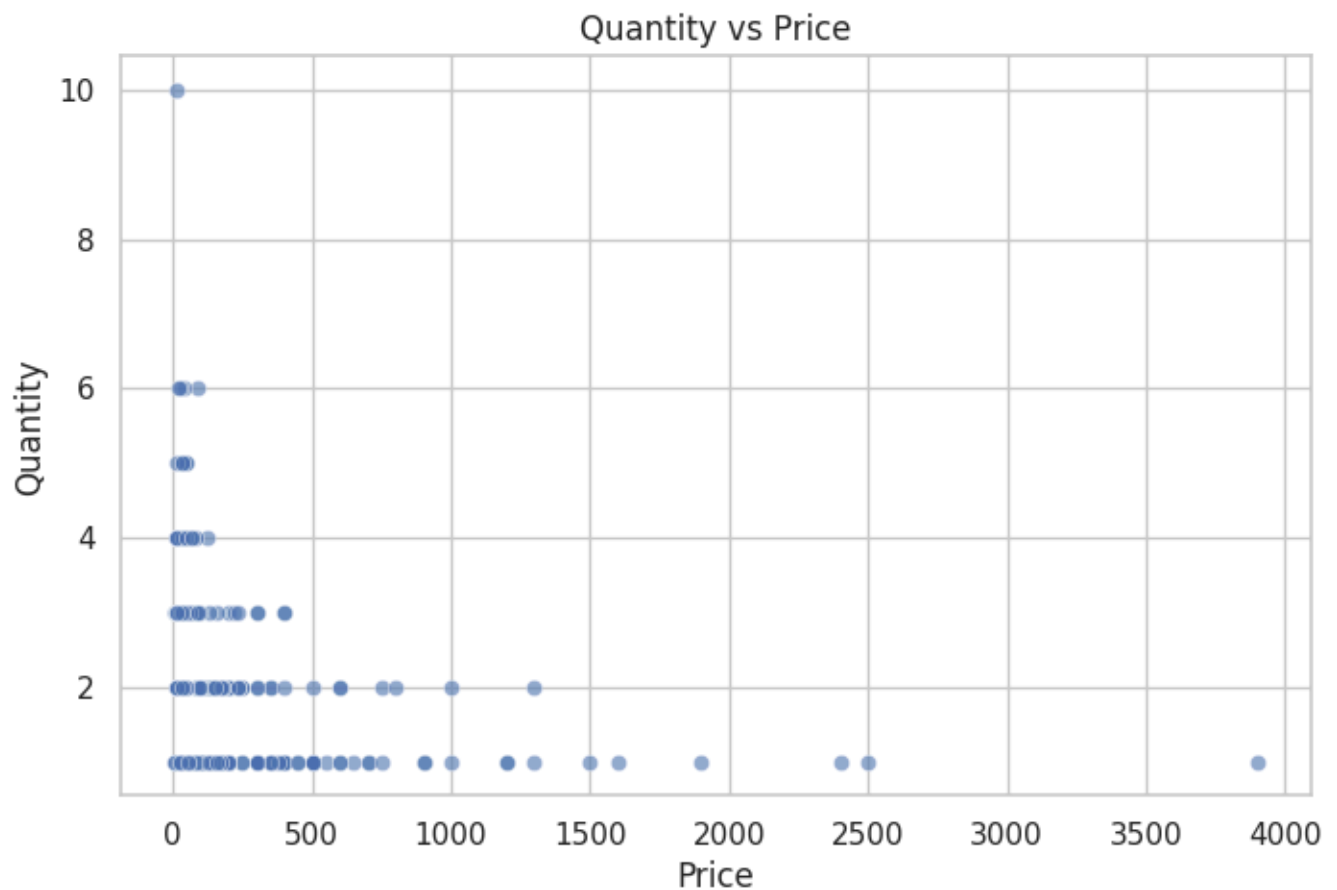
```
sns.barplot(x="total_qty", y="product", data=top_qty, palette="viridis")
```



```
plt.figure(figsize=(8,5))
sns.histplot(df_summary["revenue"], bins=10, kde=True, color="skyblue")
plt.title("Revenue Distribution")
plt.xlabel("Revenue")
plt.ylabel("Frequency")
plt.show()
```



```
plt.figure(figsize=(8,5))
sns.scatterplot(x="price", y="quantity", data=df, alpha=0.6)
plt.title("Quantity vs Price")
plt.xlabel("Price")
plt.ylabel("Quantity")
plt.show()
```



```
plt.figure(figsize=(8,5))
sns.scatterplot(
    x="total_qty",
    y="revenue",
    size="revenue",
    data=df_summary,
    sizes=(50, 500),
    alpha=0.7
)
plt.title("Revenue vs Quantity (Bubble Size = Revenue)")
plt.xlabel("Total Quantity")
plt.ylabel("Revenue")
plt.show()
```



