```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import files
uploaded = files.upload()

file_path = "sales_data.csv"   # the uploaded file name
df = pd.read_csv(file_path)

print("Initial Data:")
print(df.head())
```

Choose Files sales_data.csv

sales_data.csv(text/csv) - 571 bytes, last modified: 11/19/2025 - 100% done

```
Saving sales_data.csv to sales_data.csv
Initial Data:
         Date    Product  Sales  Quantity Region
0  01-01-2023  Product A    200         4  North
1  02-01-2023  Product B    150         3  South
2  03-01-2023  Product A    220         5  North
3  04-01-2023  Product C    300         6   East
4  05-01-2023  Product B    180         4   West
```

```
# CELL 3 — Check & Handle Missing Values
import os
OUTPUT_DIR = "/content/workbook_outputs"
os.makedirs(OUTPUT_DIR, exist_ok=True)

print("Missing Values (before):")
print(df.isnull().sum())

# fill Sales with mean, drop rows missing Product/Quantity/Region
if 'Sales' in df.columns:
    df['Sales'].fillna(df['Sales'].mean(), inplace=True)
df.dropna(subset=[c for c in ['Product', 'Quantity', 'Region'] if c in df

print("\nMissing Values (after):")
print(df.isnull().sum())
```

```
Missing Values (before):
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64

Missing Values (after):
Date        0
Product     0
```

```
Sales       0
Quantity    0
Region      0
dtype: int64
/tmp/ipython-input-3438758548.py:11: FutureWarning: A value is trying to be s
The behavior will change in pandas 3.0. This inplace method will never work b

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.


  df['Sales'].fillna(df['Sales'].mean(), inplace=True)
```

```
# CELL 4 — Summary Statistics
print("Summary Statistics (numeric columns):")
display(df.describe(include=[np.number]))   # nicer display in Colab

# If you want non-numeric summaries too:
print("\nSummary (object columns):")
display(df.describe(include=[object]))
```

Summary Statistics (numeric columns):

	Sales	Quantity
count	16.000000	16.000000
mean	237.500000	5.375000
std	64.031242	1.746425
min	150.000000	3.000000
25%	187.500000	4.000000
50%	225.000000	5.500000
75%	302.500000	7.000000
max	340.000000	8.000000

Summary (object columns):

	Date	Product	Region
count	16	16	16
unique	16	3	4
top	01-01-2023	Product A	North
freq	1	6	6

```
# CELL 5 — Group by Product Summary
product_summary = df.groupby('Product').agg({
    'Sales': 'sum' if 'Sales' in df.columns else lambda x: 0,
    'Quantity': 'sum' if 'Quantity' in df.columns else lambda x: 0
}).reset_index()
```

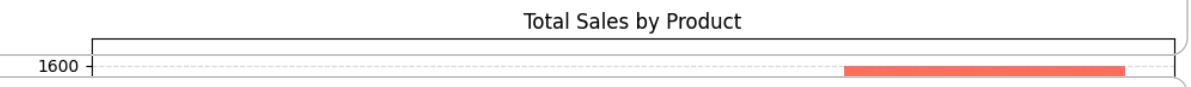
```
print("Product Summary:")
display(product_summary)
```

```
Product Summary:
```

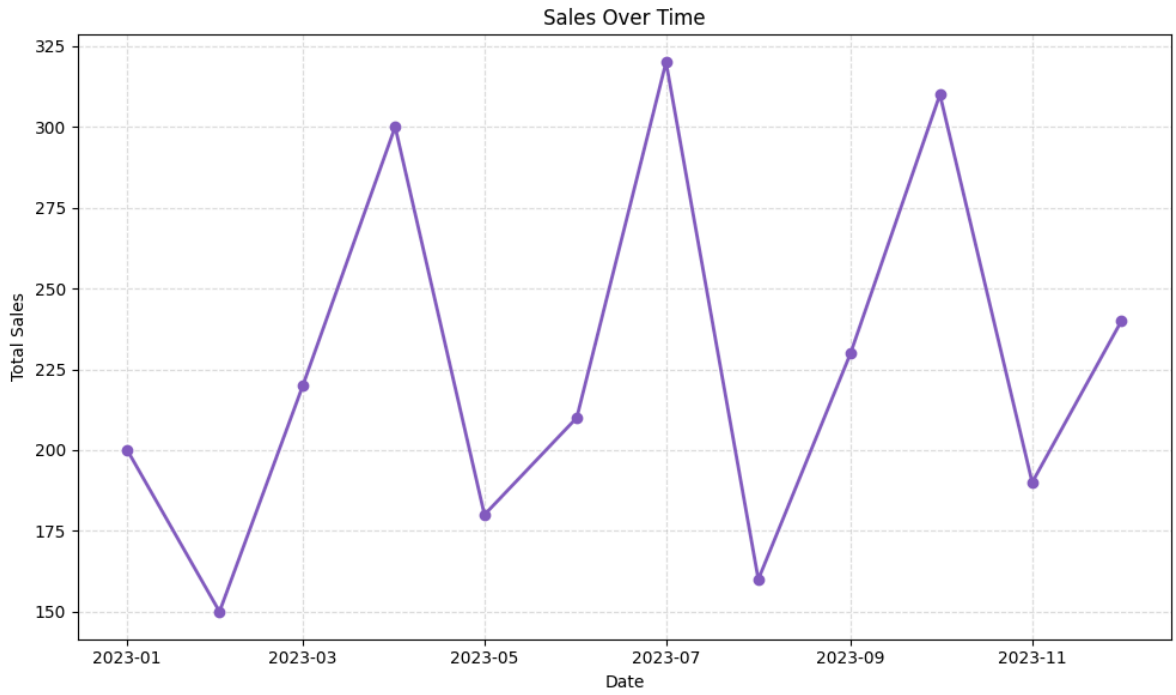
	Product	Sales	Quantity
0	Product A	1350	33
1	Product B	850	17
2	Product C	1600	36

Next steps: Generate code with `product_summary` New interactive sheet

```
# CELL 6 — Bar Chart (coloured) & save image
plt.figure(figsize=(10,6))
colors = ['#5A8DEE', '#39C0ED', '#FF6F61', '#00C9A7']  # extra colours fo
prod = product_summary['Product'].astype(str).tolist()
vals = product_summary['Sales'].tolist()

# cycle colours if fewer/more products
bar_colors = [colors[i % len(colors)] for i in range(len(prod))]
plt.bar(prod, vals, color=bar_colors)
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('Total Sales by Product')
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.xticks(rotation=30, ha='right')
plt.tight_layout()

png_path = os.path.join(OUTPUT_DIR, "bar_total_sales_by_product.png")
plt.savefig(png_path, bbox_inches='tight')
plt.show()
print("Saved:", png_path)
```




```
# CELL 7 — Line Chart (Sales over time) & save image
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
    sales_over_time = df.groupby('Date').agg({'Sales': 'sum'}).reset_index()
    sales_over_time = sales_over_time.dropna(subset=['Date'])

    plt.figure(figsize=(10,6))
    plt.plot(sales_over_time['Date'], sales_over_time['Sales'],
             marker='o', linewidth=2, color='#845EC2')
    plt.xlabel('Date')
    plt.ylabel('Total Sales')
    plt.title('Sales Over Time')
    plt.grid(True, linestyle='--', alpha=0.4)
    plt.tight_layout()

    png_path = os.path.join(OUTPUT_DIR, "line_sales_over_time.png")
    plt.savefig(png_path, bbox_inches='tight')
    plt.show()
    print("Saved:", png_path)
else:
    print("No 'Date' column found — skip this cell.")
```




```
Saved: /content/workbook_outputs/line_sales_over_time.png
```

```
# CELL 8 — Pivot Table (sales by Region x Product) and save CSV
if set(['Region','Product','Sales']).issubset(df.columns):
    pivot_table = df.pivot_table(
        values='Sales',
        index='Region',
        columns='Product',
        aggfunc=np.sum,
        fill_value=0
    )
```

```
    print("Pivot Table:")
    display(pivot_table)
    pivot_csv = os.path.join(OUTPUT_DIR, "pivot_region_product_sales.csv"
    pivot_table.to_csv(pivot_csv)
    print("Pivot saved to:", pivot_csv)
else:
    print("Required columns for pivot (Region, Product, Sales) not presen
```

```
Pivot Table:
/tmp/ipython-input-2961764987.py:3: FutureWarning: The provided callable <fun
  pivot_table = df.pivot_table(
```

Product	Product A	Product B	Product C
Region			
East	0	0	1600
North	1350	0	0
South	0	480	0
West	0	370	0

```
Pivot saved to: /content/workbook_outputs/pivot_region_product_sales.csv
```

Next steps: Generate code with pivot_table | New interactive sheet

```
# CELL 9 — Correlation Matrix (numeric only) and save
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
if len(numeric_cols) >= 2:
    correlation_matrix = df[numeric_cols].corr()
    print("Correlation Matrix:")
    display(correlation_matrix)
    corr_csv = os.path.join(OUTPUT_DIR, "correlation_matrix.csv")
    correlation_matrix.to_csv(corr_csv)
    print("Saved:", corr_csv)
else:
    print("Not enough numeric columns for correlation. Numeric columns found
```

```
Correlation Matrix:
```

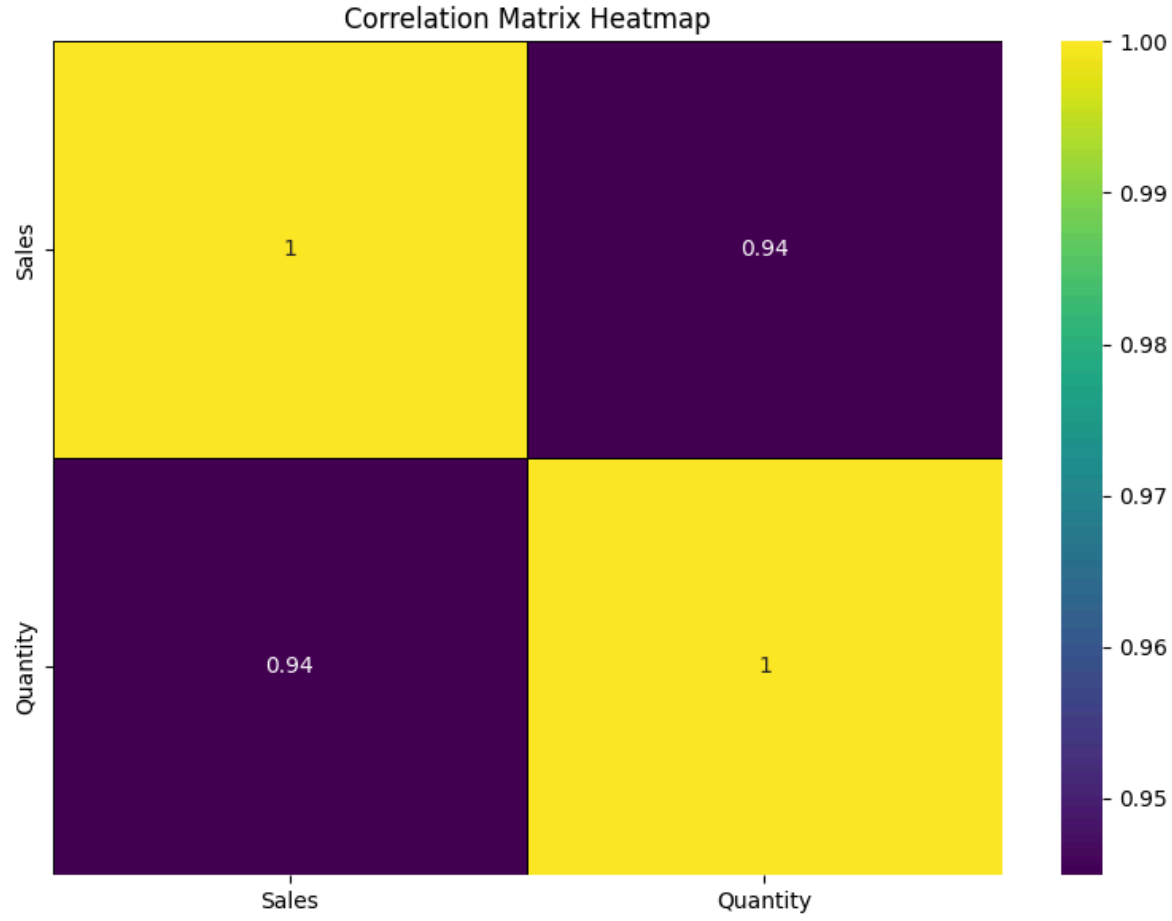
	Sales	Quantity
Sales	1.000000	0.944922
Quantity	0.944922	1.000000

```
Saved: /content/workbook_outputs/correlation_matrix.csv
```

Next steps: Generate code with correlation_matrix | New interactive sheet

```
# CELL 10 — Heatmap of correlation & save image
if 'correlation_matrix' in globals():
```

```
    plt.figure(figsize=(8,6))
    sns.heatmap(
        correlation_matrix,
        annot=True,
        cmap='viridis',
        linewidths=0.5,
        linecolor='black'
    )
    plt.title("Correlation Matrix Heatmap")
    plt.tight_layout()
    png_path = os.path.join(OUTPUT_DIR, "heatmap_correlation.png")
    plt.savefig(png_path, bbox_inches='tight')
    plt.show()
    print("Saved:", png_path)
else:
    print("Run the correlation cell first (CELL 9).")
```




```
Saved: /content/workbook_outputs/heatmap_correlation.png
```