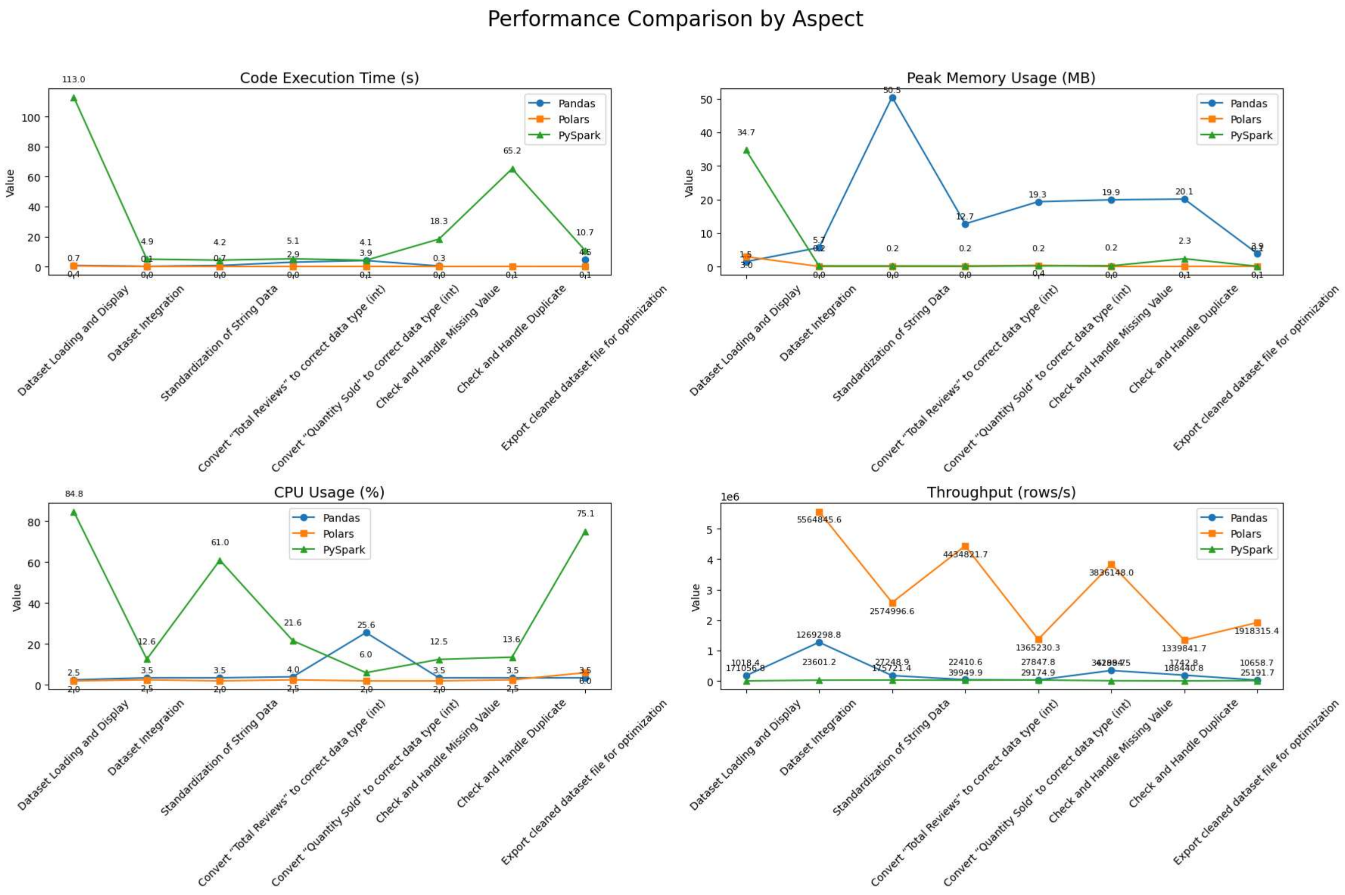
```
import pandas as pd
import matplotlib.pyplot as plt

# Load your data
xls = pd.ExcelFile('Data for graph.xlsx')
pandas_df = pd.read_excel(xls, 'Pandas')
polars_df = pd.read_excel(xls, 'Polars')
pyspark_df = pd.read_excel(xls, 'PySpark')

# List of aspects to compare
aspects = ['Code Execution Time (s)', 'Peak Memory Usage (MB)', 'CPU Usage (%)', 'Throughput (rows/s)']

# Convert all relevant columns to numeric (coerce errors into NaN)
for aspect in aspects:
    pandas_df[aspect] = pd.to_numeric(pandas_df[aspect], errors='coerce')
    polars_df[aspect] = pd.to_numeric(polars_df[aspect], errors='coerce')
    pyspark_df[aspect] = pd.to_numeric(pyspark_df[aspect], errors='coerce')

# Plot setup
fig, axs = plt.subplots(2, 2, figsize=(18, 12))
offsets = {'Pandas': 5, 'Polars': -10, 'PySpark': 15}

# Plot each aspect
for i, aspect in enumerate(aspects):
    ax = axs[i//2, i%2]

    ops = pandas_df['Operation']

    ax.plot(ops, pandas_df[aspect], marker='o', label='Pandas')
    ax.plot(ops, polars_df[aspect], marker='s', label='Polars')
    ax.plot(ops, pyspark_df[aspect], marker='^', label='PySpark')

    ax.set_title(aspect, fontsize=14)
    ax.set_ylabel('Value')
    ax.tick_params(axis='x', rotation=45)
    ax.legend()

    # Value annotations with staggered positions
    for x, y in zip(ops, pandas_df[aspect]):
        if pd.notnull(y):
            ax.annotate(f'{y:.1f}', xy=(x, y), textcoords="offset points", xytext=(0, offsets['Pandas']), ha='center', fontsize=8)
    for x, y in zip(ops, polars_df[aspect]):
        if pd.notnull(y):
            ax.annotate(f'{y:.1f}', xy=(x, y), textcoords="offset points", xytext=(0, offsets['Polars']), ha='center', fontsize=8)
    for x, y in zip(ops, pyspark_df[aspect]):
        if pd.notnull(y):
            ax.annotate(f'{y:.1f}', xy=(x, y), textcoords="offset points", xytext=(0, offsets['PySpark']), ha='center', fontsize=8)

fig.suptitle('Performance Comparison by Aspect', fontsize=20)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```


Performance Comparison by Aspect
Code Execution Time (s)
Value
0
20
40
60
80
100
Pandas
Polars
PySpark
113.0
4.9
4.2
5.1
4.1
18.3
65.2
10.7
0.7
0.1
0.7
2.9
3.9
0.3
4.5
0.4
0.0
0.0
0.0
0.1
0.0
0.1
0.1
Dataset Loading and Display
Dataset Integration
Standardization of String Data
Convert "Total Reviews" to correct data type (int)
Convert "Quantity Sold" to correct data type (int)
Check and Handle Missing Value
Check and Handle Duplicate
Export cleaned dataset file for optimization
Peak Memory Usage (MB)
Value
0
10
20
30
40
50
Pandas
Polars
PySpark
34.7
50.5
5.7
12.7
19.3
19.9
20.1
3.9
1.5
3.0
0.2
0.2
0.2
0.2
0.2
0.2
2.3
0.1
0.0
0.0
0.0
0.4
0.0
0.1
0.1
Dataset Loading and Display
Dataset Integration
Standardization of String Data
Convert "Total Reviews" to correct data type (int)
Convert "Quantity Sold" to correct data type (int)
Check and Handle Missing Value
Check and Handle Duplicate
Export cleaned dataset file for optimization
CPU Usage (%)
Value
0
20
40
60
80
Pandas
Polars
PySpark
84.8
12.6
61.0
21.6
25.6
6.0
12.5
13.6
75.1
2.5
3.5
3.5
4.0
3.5
3.5
3.5
2.0
2.5
2.0
2.5
2.0
2.0
2.5
6.0
Dataset Loading and Display
Dataset Integration
Standardization of String Data
Convert "Total Reviews" to correct data type (int)
Convert "Quantity Sold" to correct data type (int)
Check and Handle Missing Value
Check and Handle Duplicate
Export cleaned dataset file for optimization
Throughput (rows/s)
1e6
Value
0
1
2
3
4
5
Pandas
Polars
PySpark
5564845.6
2574996.6
4434821.7
1365230.3
3836148.0
1339841.7
1918315.4
1269298.8
1018.4
171056.8
23601.2
27248.9
175721.4
22410.6
39949.9
27847.8
29174.9
1742.8
188440.8
10658.7
25191.7
Dataset Loading and Display
Dataset Integration
Standardization of String Data
Convert "Total Reviews" to correct data type (int)
Convert "Quantity Sold" to correct data type (int)
Check and Handle Missing Value
Check and Handle Duplicate
Export cleaned dataset file for optimization