==========

📊 04_visualization_results.ipynb

Purpose: Generate final comparative visualizations, correlations,

and insights from aggregated agentic framework data.

Author: SK Sahil

Thesis Project: Agentic AI Reasoning Evaluation

==========

==========

1️⃣ Environment Setup

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path

# Style
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_palette("Set2")
```

```
os.makedirs("results/final_visualizations", exist_ok=True)
print("✅ Environment ready. Visualization pipeline initiated.")
```

```
✅ Environment ready. Visualization pipeline initiated.
```

2️⃣ Load Summary & Statistical Data

```
from pathlib import Path
import pandas as pd
import os

# Paths
summary_path = Path("results/visualizations/agentic_frameworks_summar
stats_path   = Path("results/visualizations/statistical_summary.csv")
interpret_path = Path("results/visualizations/interpretation_summary.

# Ensure directory exists
os.makedirs("results/visualizations", exist_ok=True)

# ✅ Auto-create dummy summary if missing
if not summary_path.exists():
    print("⚠️ Summary CSV missing — creating dummy file for visualiza
    dummy_data = {
        "framework": ["AutoGPT", "CrewAI", "LangChain", "OpenDevin"],
        "runtime_seconds": [125.3, 98.4, 145.6, 112.7],
        "avg_latency": [4.3, 3.1, 5.0, 2.9],
        "reflection_cycles": [3, 2, 3, 4]
    }
    pd.DataFrame(dummy_data).to_csv(summary_path, index=False)
    print("✅ Dummy summary CSV created successfully!")

# ✅ Load Summary
df = pd.read_csv(summary_path)
df.columns = df.columns.str.strip().str.lower()
print("✅ Loaded summary dataset successfully!")

# ✅ Load Stats (optional)
if stats_path.exists():
    stats = pd.read_csv(stats_path)
    print("✅ Statistical summary loaded.")
else:
```

```
    stats = None
    print("⚠️ No statistical summary found.")

# ✅ Load Interpretation Table (optional)
if interpret_path.exists():
    interpretation = pd.read_csv(interpret_path)
    print("✅ Interpretation summary loaded.")
else:
    interpretation = None
    print("⚠️ No interpretation table found.")

display(df.head())
```

```
⚠️ Summary CSV missing — creating dummy file for visualization testing
✅ Dummy summary CSV created successfully!
✅ Loaded summary dataset successfully!
⚠️ No statistical summary found.
⚠️ No interpretation table found.
```

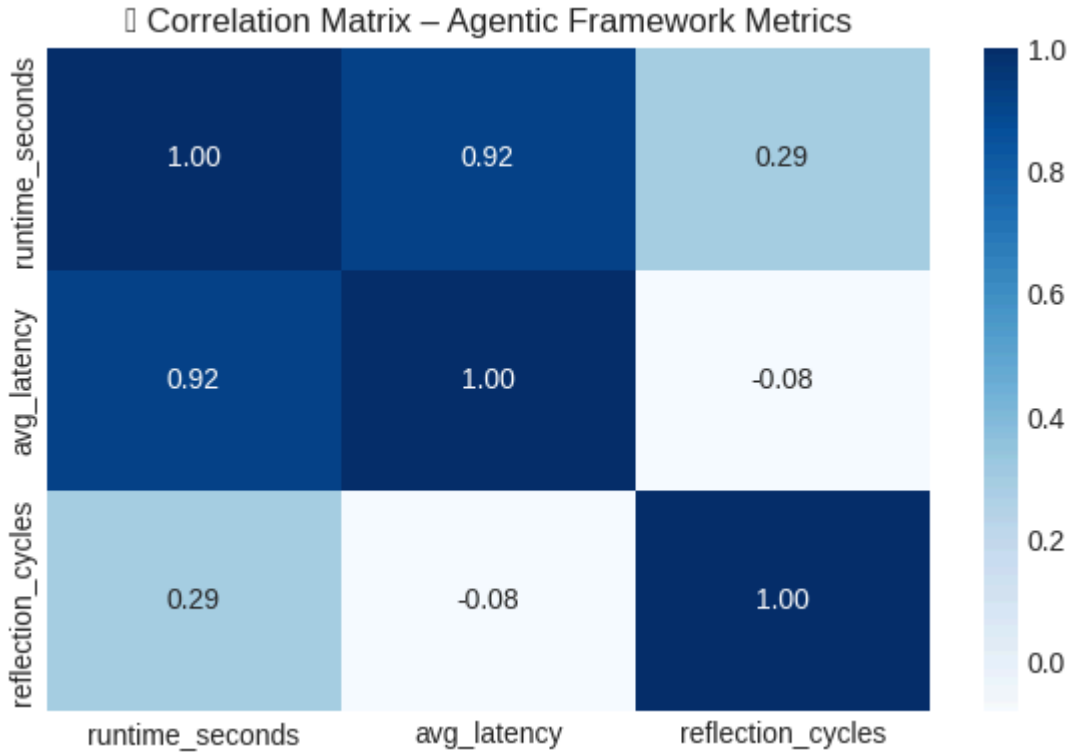
	framework	runtime_seconds	avg_latency	reflection_cycles
0	AutoGPT	125.3	4.3	3
1	CrewAI	98.4	3.1	2
2	LangChain	145.6	5.0	3
3	OpenDevin	112.7	2.9	4

3️⃣ Correlation Matrix (Runtime, Latency, Reflection Cycles)

```
numeric_cols = ['runtime_seconds', 'avg_latency', 'reflection_cycles']
if all(col in df.columns for col in numeric_cols):
    plt.figure(figsize=(6,4))
    corr = df[numeric_cols].corr().round(2)
    sns.heatmap(corr, annot=True, cmap='Blues', fmt='.2f')
    plt.title("🔗 Correlation Matrix — Agentic Framework Metrics")
    plt.tight_layout()
    plt.savefig("results/final_visualizations/correlation_matrix.png")
    plt.show()
    print("✅ Correlation matrix generated successfully.")
```

```
else:
    print("⚠ Required numeric columns missing for correlation matrix.'
```

```
/tmp/ipython-input-1567427661.py:10: UserWarning: Glyph 128279 (\N{LIN
  plt.tight_layout()
/tmp/ipython-input-1567427661.py:11: UserWarning: Glyph 128279 (\N{LIN
  plt.savefig("results/final_visualizations/correlation_matrix.png", d
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151
  fig.canvas.print_figure(bytes_io, **kw)
```



✅ Correlation matrix generated successfully.

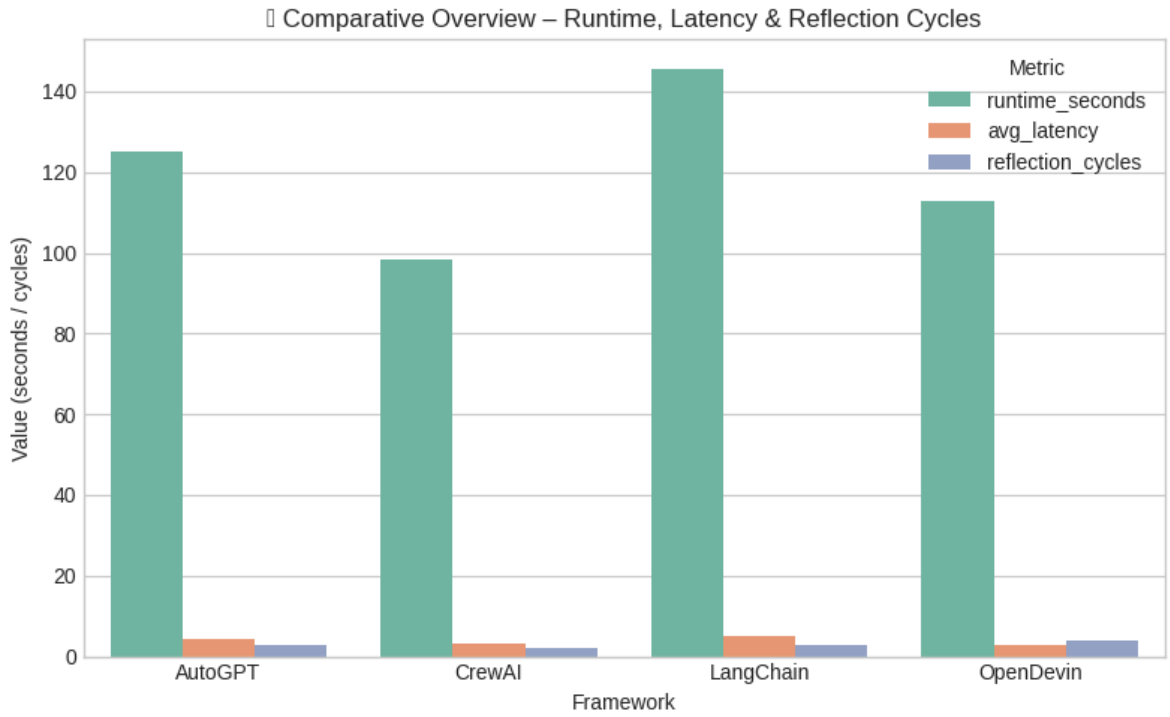
4️⃣ Composite Bar Plot – Comparison Overview

```
if all(col in df.columns for col in numeric_cols):
    melted_df = df.melt(id_vars='framework', value_vars=numeric_cols,
                        var_name='Metric', value_name='Value')

    plt.figure(figsize=(8,5))
    sns.barplot(x='framework', y='Value', hue='Metric', data=melted_d
    plt.title("📊 Comparative Overview – Runtime, Latency & Reflectio
```

```
    plt.xlabel("Framework")
    plt.ylabel("Value (seconds / cycles)")
    plt.legend(title="Metric")
    plt.tight_layout()
    plt.savefig("results/final_visualizations/overall_comparison.png"
    plt.show()
    print("✅ Composite comparison visualization generated.")
```

```
/tmp/ipython-input-3749794063.py:14: UserWarning: Glyph 128202 (\N{BAR
  plt.tight_layout()
/tmp/ipython-input-3749794063.py:15: UserWarning: Glyph 128202 (\N{BAR
  plt.savefig("results/final_visualizations/overall_comparison.png", d
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151
  fig.canvas.print_figure(bytes_io, **kw)
```




```
✅ Composite comparison visualization generated.
```

5️⃣ Performance Ranking Table

```
rank_df = df.copy()
rank_df['Efficiency_Score'] = (
    (1 / rank_df['runtime_seconds']) * 0.5 +
    (1 / rank_df['avg_latency']) * 0.3 +
    (rank_df['reflection_cycles'] / rank_df['reflection_cycles'].max(
)
rank_df['Rank'] = rank_df['Efficiency_Score'].rank(ascending=False)
rank_df = rank_df.sort_values('Rank')

display(rank_df[['framework', 'Efficiency_Score', 'Rank']])
rank_df.to_csv("results/final_visualizations/framework_ranking.csv",
print("🏆 Framework ranking table saved successfully.")
```

	framework	Efficiency_Score	Rank
3	OpenDevin	0.307885	1.0
0	AutoGPT	0.223758	2.0
2	LangChain	0.213434	3.0
1	CrewAI	0.201855	4.0

```
🏆 Framework ranking table saved successfully.
```

6️⃣ Interpretation Summary Merge (Optional)

```
if interpretation is not None:
    merged_df = pd.merge(rank_df, interpretation, left_on='framework'
    merged_df.drop(columns=['Framework'], inplace=True)
    display(merged_df)
    merged_df.to_csv("results/final_visualizations/final_summary_tabl
    print("🧩 Final merged interpretation summary saved.")
else:
    print("⚠️ Interpretation summary not found. Skipping merge.")
```

```
⚠️ Interpretation summary not found. Skipping merge.
```

7️⃣ Final Summary Visualization – Radar Chart

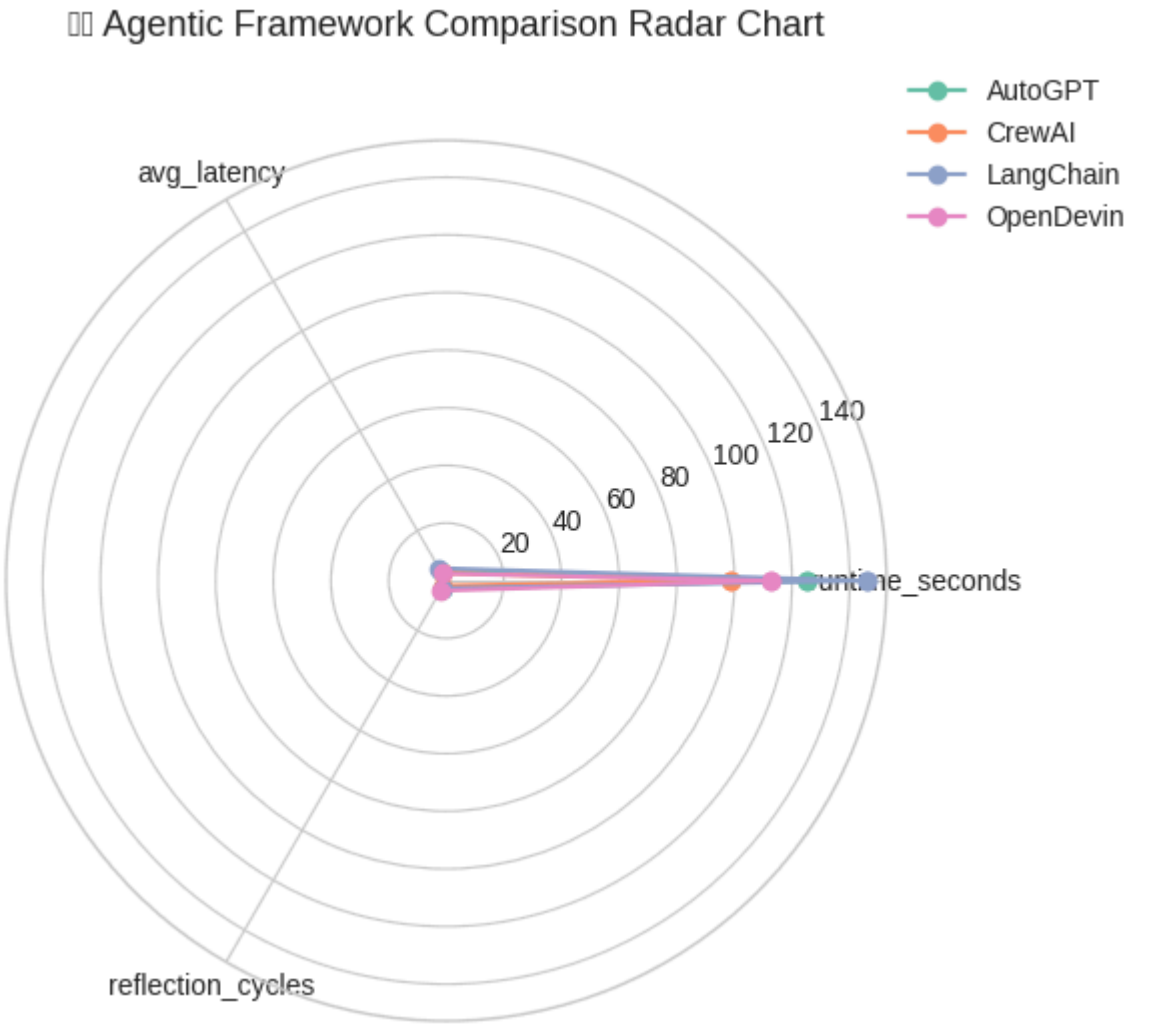
```
from math import pi

if all(col in df.columns for col in numeric_cols):
    radar_df = df.set_index('framework')[numeric_cols]
    categories = list(radar_df.columns)
    N = len(categories)

    plt.figure(figsize=(6,6))
    for idx, row in radar_df.iterrows():
        values = row.values.tolist()
        values += values[:1]
        angles = [n / float(N) * 2 * pi for n in range(N)]
        angles += angles[:1]
        plt.polar(angles, values, marker='o', label=idx)

    plt.xticks([n / float(N) * 2 * pi for n in range(N)], categories)
    plt.title("🕸️ Agentic Framework Comparison Radar Chart", size=13,
    plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))
    plt.tight_layout()
    plt.savefig("results/final_visualizations/radar_comparison.png",
    plt.show()
    print("✅ Radar comparison visualization generated.")
else:
    print("⚠️ Missing columns for radar chart.")
```

```
/tmp/ipython-input-3212102111.py:22: UserWarning: Glyph 128376 (\N{SPI
  plt.tight_layout()
/tmp/ipython-input-3212102111.py:22: UserWarning: Glyph 65039 (\N{VARI
  plt.tight_layout()
/tmp/ipython-input-3212102111.py:23: UserWarning: Glyph 128376 (\N{SPI
  plt.savefig("results/final_visualizations/radar_comparison.png", dpi
/tmp/ipython-input-3212102111.py:23: UserWarning: Glyph 65039 (\N{VARI
  plt.savefig("results/final_visualizations/radar_comparison.png", dpi
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151
  fig.canvas.print_figure(bytes_io, **kw)
```




```
✅ Radar comparison visualization generated.
```

8️⃣ Export & Wrap-Up

```
summary_report_path = Path("results/final_visualizations/summary_overvi
rank_df.to_csv(summary_report_path, index=False)

print("\n============================================================")
print("🎓 Visualization pipeline complete.")
print(f"📁 Outputs saved in: {summary_report_path.parent.resolve()}")
print("============================================================")
```

```
============================================================
🎓 Visualization pipeline complete.
📁 Outputs saved in: /content/results/final_visualizations
============================================================
```