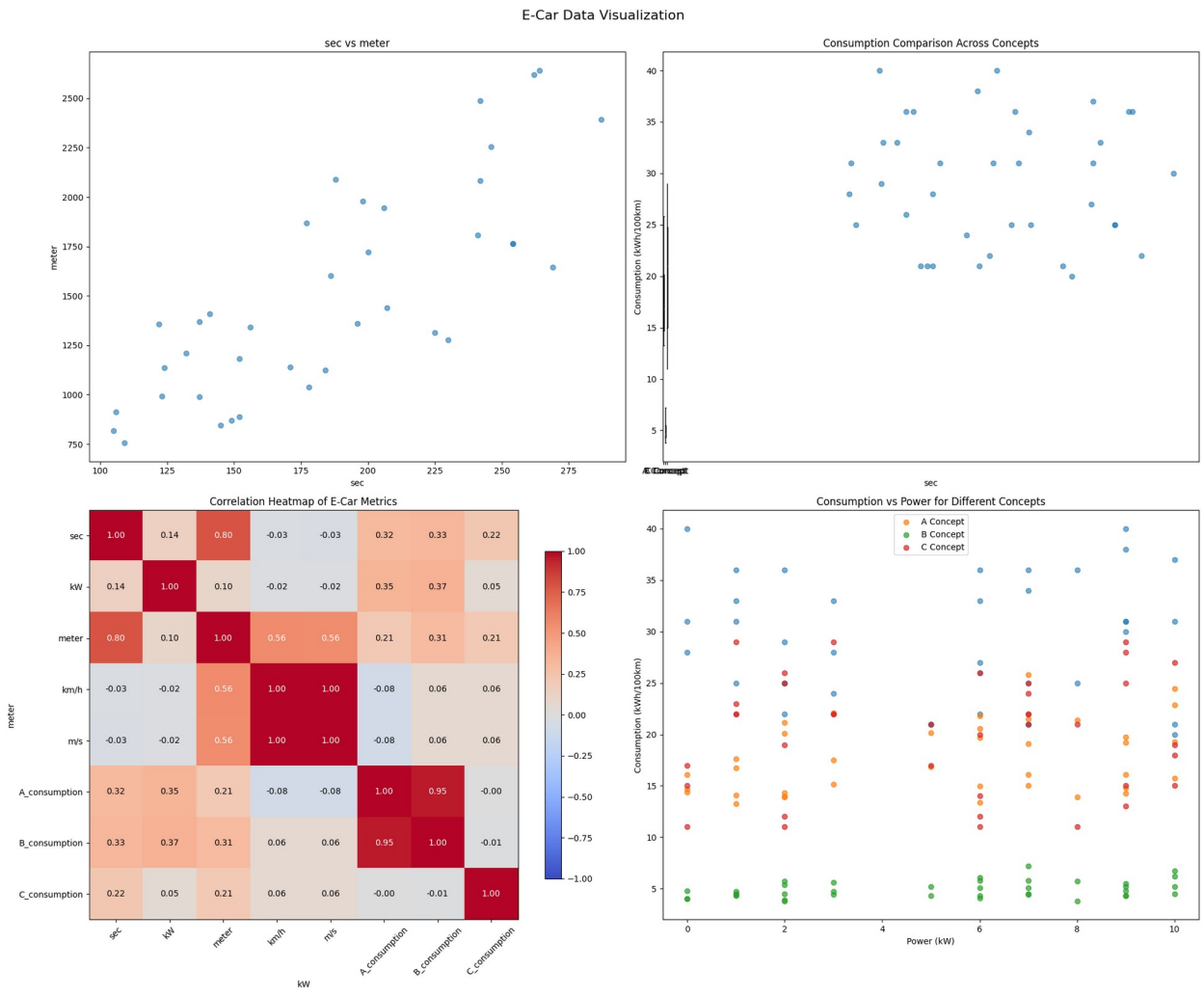
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the data
data = pd.DataFrame({
    'sec': [242, 178, 152, 287, 188, 109, 145, 225, 242, 196, 171,
132, 246, 149, 200, 269, 207, 105, 137, 254, 137, 106, 141, 186, 177,
124, 262, 123, 152, 230, 122, 184, 241, 198, 254, 264, 156, 206],
    'kW': [1, 10, 3, 9, 0, 7, 7, 5, 10, 2, 3, 1, 3, 5, 9, 6, 1, 0, 8,
2, 6, 0, 2, 9, 9, 6, 1, 2, 7, 10, 9, 2, 6, 6, 8, 7, 10, 7],
    'meter': [2084, 1038, 1182, 2392, 2089, 757, 846, 1313, 2487,
1361, 1140, 1210, 2255, 869, 1722, 1644, 1438, 817, 1370, 1764, 989,
913, 1410, 1602, 1868, 1137, 2620, 991, 887, 1278, 1356, 1124, 1808,
1980, 1764, 2640, 1343, 1946],
    'km/h': [31, 21, 28, 30, 40, 25, 21, 21, 37, 25, 24, 33, 33, 21,
31, 22, 25, 28, 36, 25, 26, 31, 36, 31, 38, 33, 36, 29, 21, 20, 40,
22, 27, 36, 25, 36, 31, 34],
    'm/s': [8.61, 5.83, 7.78, 8.33, 11.11, 6.94, 5.83, 5.83, 10.28,
6.94, 6.67, 9.17, 9.17, 5.83, 8.61, 6.11, 6.94, 7.78, 10.00, 6.94,
7.22, 8.61, 10.00, 8.61, 10.56, 9.17, 10.00, 8.06, 5.83, 5.56, 11.11,
6.11, 7.50, 10.00, 6.94, 10.00, 8.61, 9.44],
    'A_consumption': [14.089, 15.747, 15.152, 16.069, 16.069, 16.091,
19.080, 20.170, 22.873, 21.173, 22.100, 17.641, 17.479, 16.860,
19.763, 19.691, 16.757, 14.376, 13.896, 13.896, 14.953, 14.650,
14.297, 14.819, 14.284, 13.375, 13.253, 13.943, 15.054, 19.254,
19.196, 20.105, 21.772, 20.535, 21.367, 21.493, 24.441, 25.810],
    'B_consumption': [4.400, 4.500, 4.400, 4.800, 4.800, 4.500, 5.100,
5.200, 6.200, 5.400, 5.600, 4.700, 4.700, 4.300, 5.200, 5.100, 4.500,
4.000, 3.800, 3.800, 4.100, 4.000, 3.900, 4.300, 4.300, 4.300, 4.300,
4.500, 4.400, 5.200, 5.500, 5.700, 6.100, 5.800, 5.700, 5.800, 6.700,
7.200],
    'C_consumption': [29.000, 18.000, 22.000, 28.000, 17.000, 22.000,
22.000, 21.000, 15.000, 25.000, 29.000, 22.000, 22.000, 17.000,
13.000, 26.000, 22.000, 15.000, 21.000, 26.000, 12.000, 11.000,
19.000, 15.000, 29.000, 20.000, 23.000, 11.000, 24.000, 19.000,
25.000, 12.000, 11.000, 14.000, 11.000, 25.000, 27.000, 21.000]
})

# Visualization Functions
def create_visualizations():
    # Set up the plotting style
    plt.style.use('default')

    # Create a figure with subplots
    fig, axs = plt.subplots(2, 2, figsize=(20, 16))
```

```
    # 1. Scatter Matrix (partial view)
    metrics = ['sec', 'kW', 'meter', 'km/h']
    for i, metric1 in enumerate(metrics[:2]):
        for j, metric2 in enumerate(metrics[2:]):
            axs[i, j].scatter(data[metric1], data[metric2], alpha=0.6)
            axs[i, j].set_xlabel(metric1)
            axs[i, j].set_ylabel(metric2)
            axs[i, j].set_title(f'{metric1} vs {metric2}')

    # 2. Consumption Boxplot
    consumption_data = data[['A_consumption', 'B_consumption',
'C_consumption']]
    axs[0, 1].boxplot([consumption_data['A_consumption'],
                       consumption_data['B_consumption'],
                       consumption_data['C_consumption']],
                      labels=['A Concept', 'B Concept', 'C Concept'])
    axs[0, 1].set_title('Consumption Comparison Across Concepts')
    axs[0, 1].set_ylabel('Consumption (kWh/100km)')

    # 3. Correlation Heatmap
    correlation_matrix = data.corr()
    im = axs[1, 0].imshow(correlation_matrix, cmap='coolwarm',
aspect='auto', vmin=-1, vmax=1)
    plt.colorbar(im, ax=axs[1, 0], shrink=0.8)

    # Add text annotations
    for i in range(len(correlation_matrix.columns)):
        for j in range(len(correlation_matrix.columns)):
            axs[1, 0].text(j, i, f'{correlation_matrix.iloc[i,
j]:.2f}',
                           ha='center', va='center',
                           color='white' if
abs(correlation_matrix.iloc[i, j]) > 0.5 else 'black')

    axs[1, 0].set_xticks(range(len(correlation_matrix.columns)))
    axs[1, 0].set_yticks(range(len(correlation_matrix.columns)))
    axs[1, 0].set_xticklabels(correlation_matrix.columns, rotation=45)
    axs[1, 0].set_yticklabels(correlation_matrix.columns)
    axs[1, 0].set_title('Correlation Heatmap of E-Car Metrics')

    # 4. Consumption vs Power Scatter Plot
    axs[1, 1].scatter(data['kW'], data['A_consumption'], label='A
Concept', alpha=0.7)
    axs[1, 1].scatter(data['kW'], data['B_consumption'], label='B
Concept', alpha=0.7)
    axs[1, 1].scatter(data['kW'], data['C_consumption'], label='C
Concept', alpha=0.7)

    axs[1, 1].set_xlabel('Power (kW)')
```

```
    axs[1, 1].set_ylabel('Consumption (kWh/100km)')
    axs[1, 1].set_title('Consumption vs Power for Different Concepts')
    axs[1, 1].legend()

    # Adjust layout and show plot
    plt.tight_layout()
    plt.suptitle('E-Car Data Visualization', fontsize=16, y=1.02)
    plt.show()

# Run visualization
create_visualizations()

# Additional Numerical Analysis
def print_additional_insights():
    print("\nConcept Consumption Statistics:")
    consumption_data = data[['A_consumption', 'B_consumption',
'C_consumption']]
    print(consumption_data.describe())

    print("\nCorrelation with Consumption:")
    metrics = ['sec', 'kW', 'meter', 'km/h', 'm/s']
    for concept in ['A_consumption', 'B_consumption',
'C_consumption']:
        print(f"\nCorrelations with {concept}:")
        correlations = data[metrics + [concept]].corr()[concept]
[metrics]
        print(correlations)

print_additional_insights()
```




```
Concept Consumption Statistics:
       A_consumption  B_consumption  C_consumption
count      38.000000      38.000000      38.000000
mean       17.672974       4.915789      20.026316
std         3.406550       0.821510       5.673489
min        13.253000       3.800000      11.000000
25%        14.692250       4.300000      15.000000
50%        16.808500       4.700000      21.000000
75%        20.153750       5.475000      24.750000
max        25.810000       7.200000      29.000000

Correlation with Consumption:

Correlations with A_consumption:
sec      0.315415
kW       0.349997
meter    0.209615
km/h    -0.076395
```

```
m/s     -0.076608
Name: A_consumption, dtype: float64

Correlations with B_consumption:
sec      0.330766
kW       0.374000
meter    0.306702
km/h     0.056173
m/s      0.055985
Name: B_consumption, dtype: float64

Correlations with C_consumption:
sec      0.219248
kW       0.053594
meter    0.207347
km/h     0.062303
m/s      0.062211
Name: C_consumption, dtype: float64
```