Interpretation of the Plots

$7^{th} Feb, 2025$

Name: Soumyadeep Ganguly

Reg no: 24MDT0082

Multivariate Analysis on Titanic Dataset

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
titanic=pd.read_csv("titanic.csv")
titanic.head()
```

Out[1]:

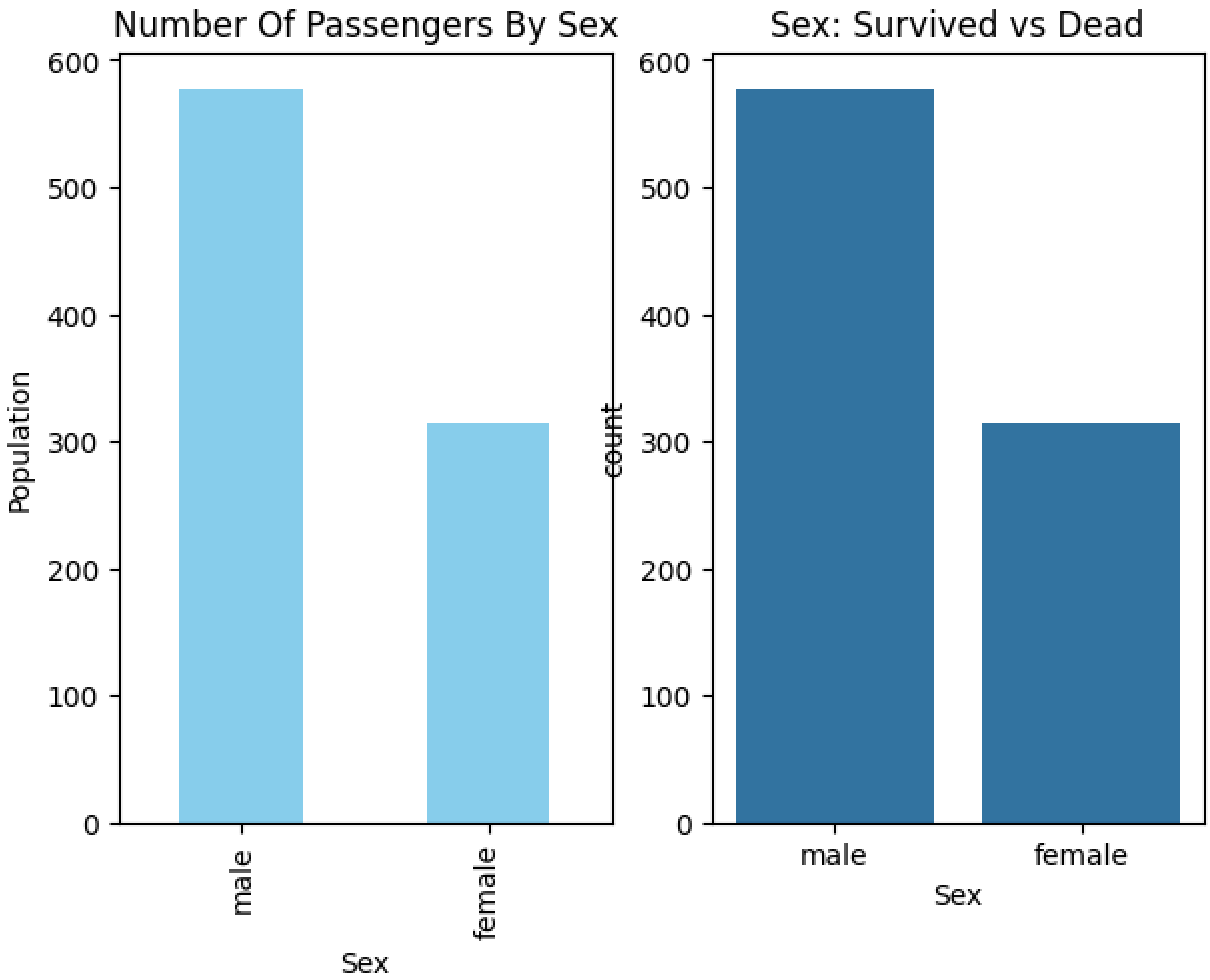
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [2]:
```
#percentage of women survived
women = titanic.loc[titanic.Sex == 'female']["Survived"]
rate_women = sum(women)/len(women)
#percentage of men survived
men = titanic.loc[titanic.Sex == 'male']["Survived"]

rate_men = sum(men)/len(men)
print(str(rate_women) +" % of women who survived." )
print(str(rate_men) + " % of men who survived." )
```

```
0.7420382165605095 % of women who survived.
0.18890814558058924 % of men who survived.
```

In [3]:
```
titanic['Survived'] = titanic['Survived'].map({0:"not_survived", 1:"survived"})
fig, ax = plt.subplots(1, 2, figsize = (7, 5))
titanic["Sex"].value_counts().plot.bar(color = "skyblue", ax = ax[0])
ax[0].set_title("Number Of Passengers By Sex")
ax[0].set_ylabel("Population")
sns.countplot(x="Sex", data=titanic, ax = ax[1])
ax[1].set_title("Sex: Survived vs Dead")
plt.show()
```

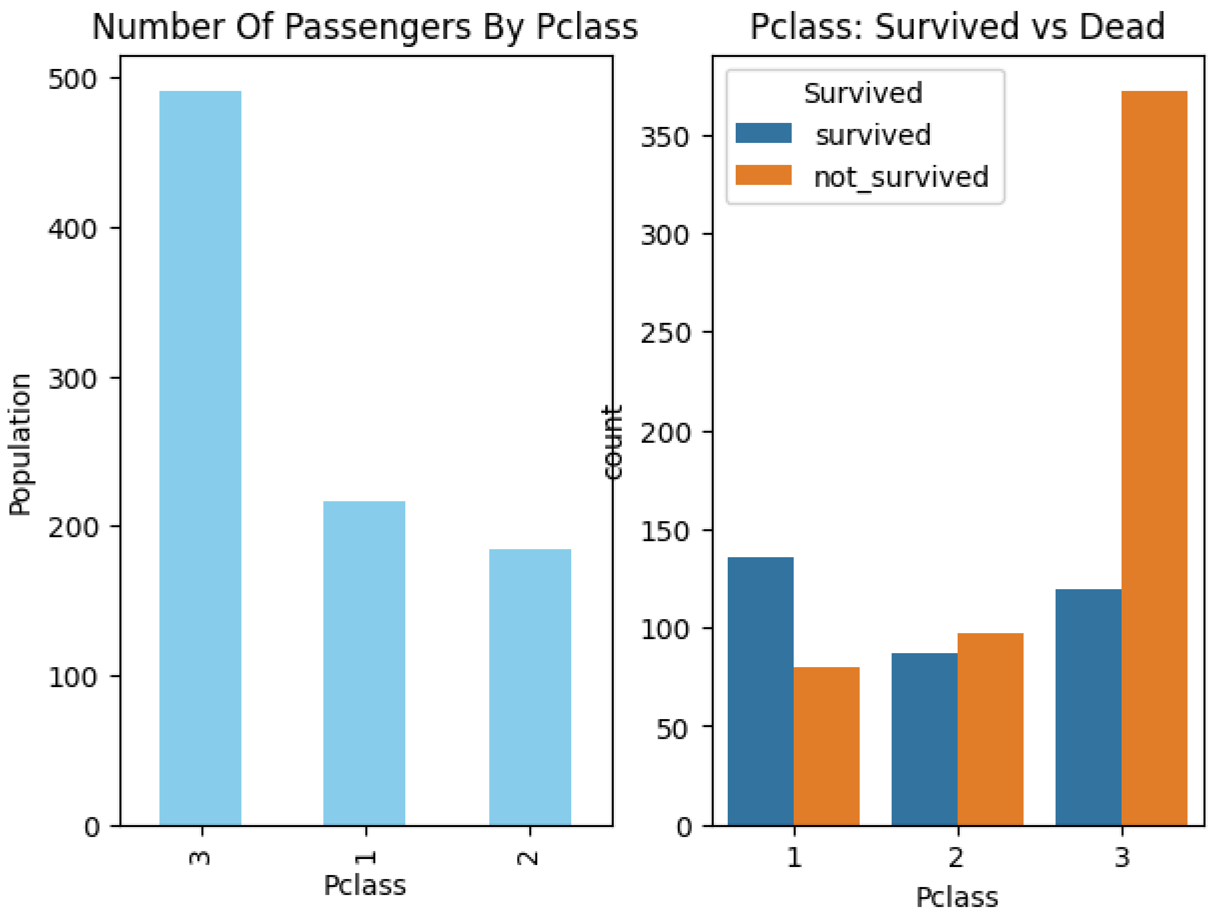


Survival by Gender:

The bar plots clearly indicate that women had a significantly higher survival rate than men. Around 74.2% of female passengers survived compared to only 18.9% of males. This supports the "women and children first" policy during evacuation.

In [4]:
```
fig, ax = plt.subplots(1, 2, figsize = (7, 5))
titanic["Pclass"].value_counts().plot.bar(color = "skyblue", ax = ax[0])
ax[0].set_title("Number Of Passengers By Pclass")
```

```
ax[0].set_ylabel("Population")
sns.countplot(x="Pclass", hue = "Survived", data = titanic, ax = ax[1])
ax[1].set_title("Pclass: Survived vs Dead")
plt.show()
```



Passenger Class Distribution and Survival Rate:

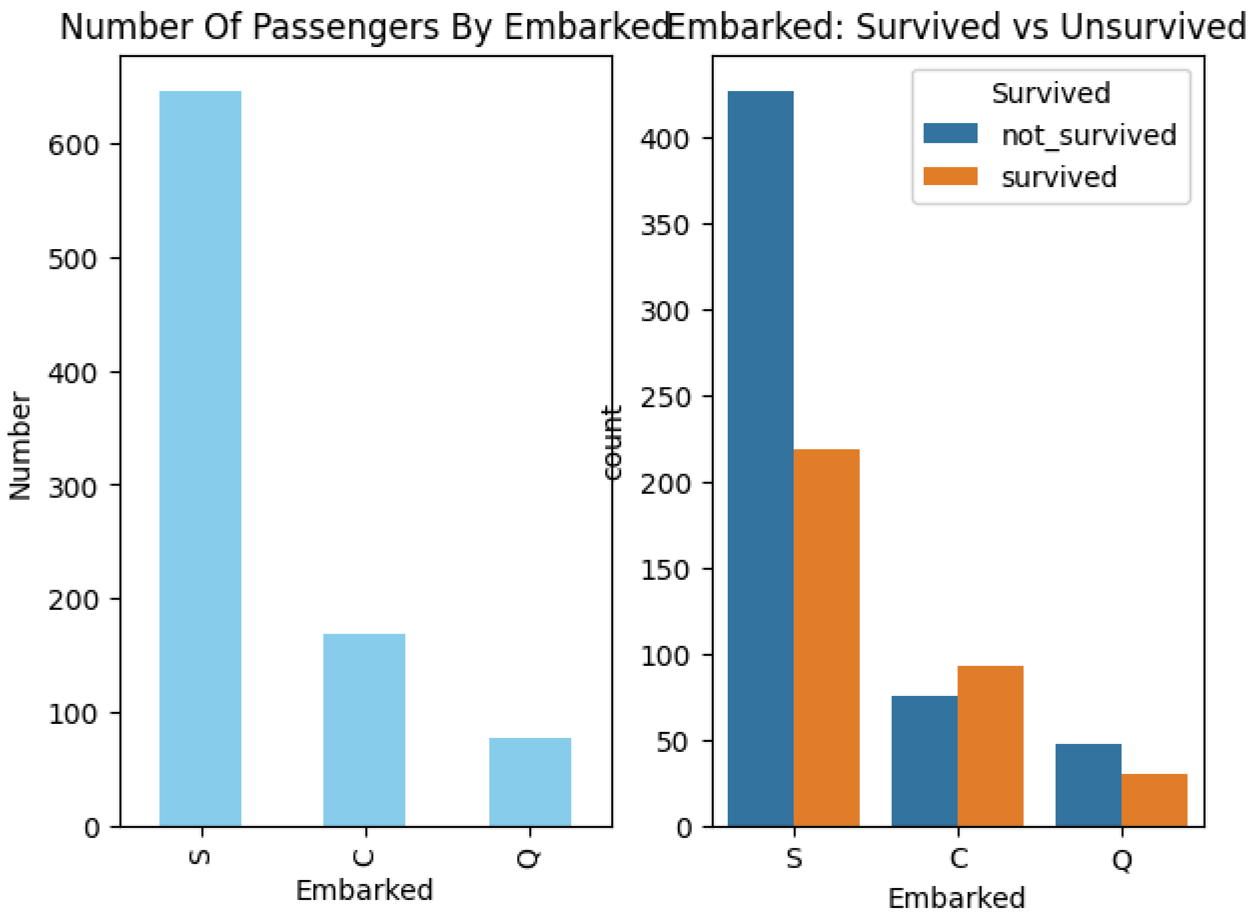
The distribution of passengers by class reveals that the majority were in third class, while the least number of passengers were in first class. The survival count plot shows that first-class passengers had the highest survival rates, whereas third-class passengers had the lowest. This suggests that socio-economic status played a crucial role in survival.

In [5]:
```
titanic["Embarked"] = titanic["Embarked"].fillna("S")
titanic.head(3)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	not_survived	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	survived	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	survived	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

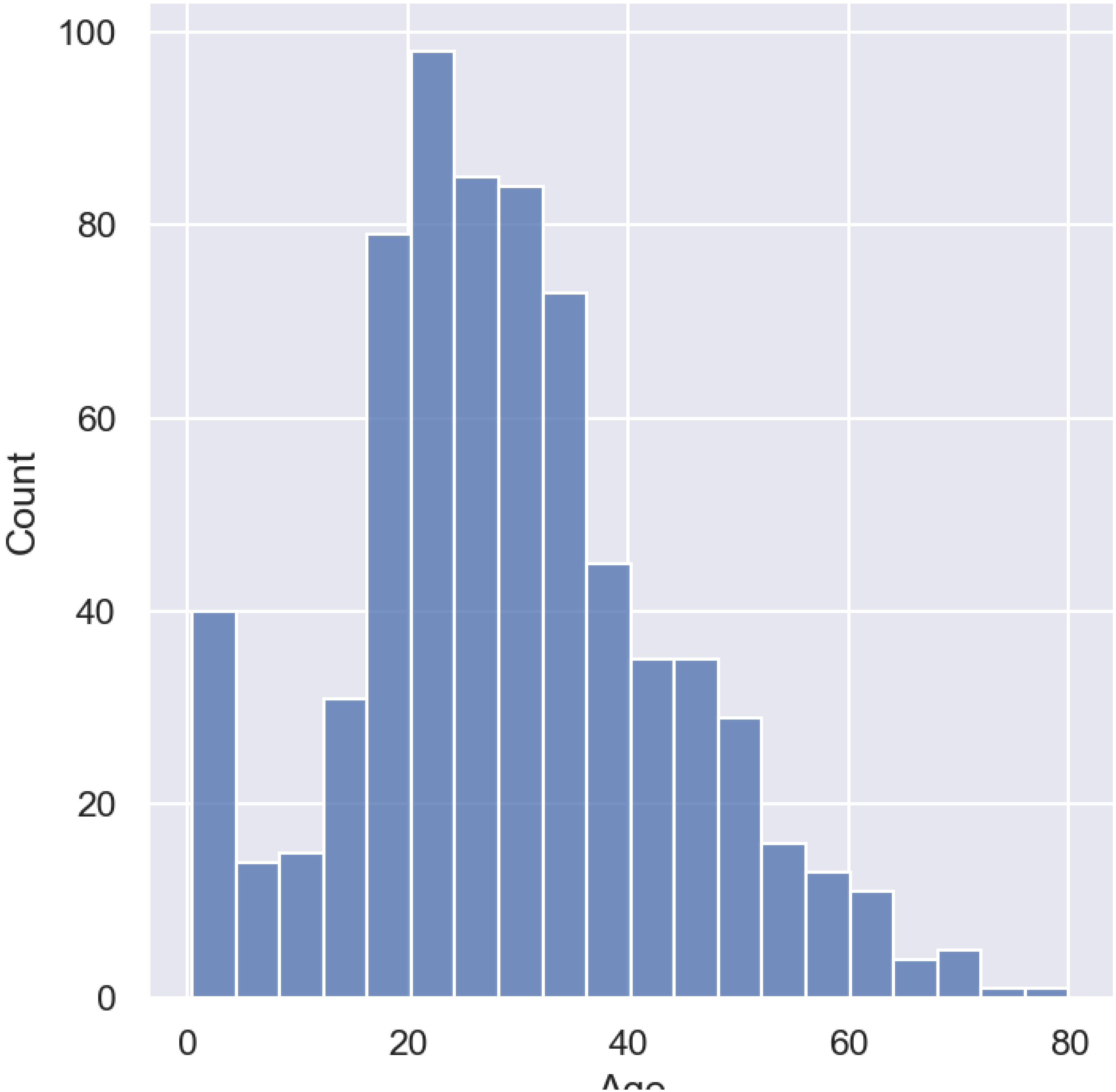
In [6]:
```
fig, ax = plt.subplots(1, 2, figsize = (7, 5))
titanic["Embarked"].value_counts().plot.bar(color = "skyblue", ax = ax[0])
ax[0].set_title("Number Of Passengers By Embarked")
ax[0].set_ylabel("Number")
sns.countplot(x = "Embarked", hue = "Survived", data = titanic, ax = ax[1])
ax[1].set_title("Embarked: Survived vs Unsurvived")
plt.show()
```



Embarkation Analysis:

The embarkation bar chart shows that most passengers boarded at 'S' (Southampton). The survival comparison plot indicates that embarkation location had some effect on survival, with passengers from Cherbourg ('C') having a higher survival rate.

In [50]:
```
sns.displot(titanic['Age'].dropna())
plt.show()
```


100
80
60
40
20
0
Count
0
20
40
60
80

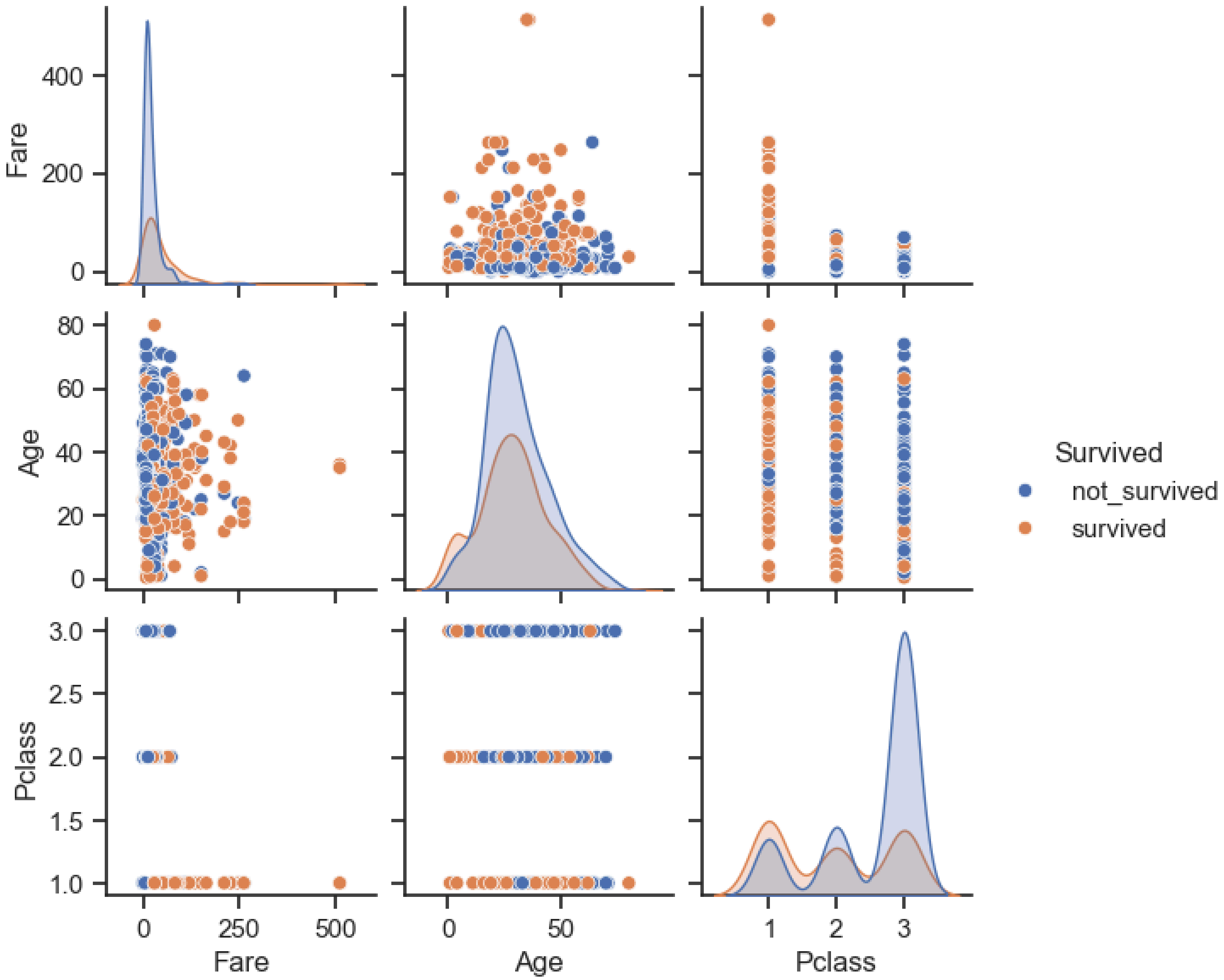
Age

Age Distribution:

The age distribution histogram highlights that most passengers were young adults, with fewer elderly passengers. This distribution is slightly right-skewed, meaning there were more younger individuals onboard. However, age itself does not seem to be a strong determinant of survival, though younger passengers, particularly children, had slightly better chances.

In [8]:

```
sns.set(style="ticks", color_codes=True)
sns.pairplot(titanic,height=2,vars = [ 'Fare','Age','Pclass'], hue="Survived")
plt.show()
```



Pairplot Analysis (Fare, Age, Pclass, Survival):

The pairplot analysis reveals that higher fares are associated with a greater likelihood of survival, indicating that wealthier passengers (often in first class) had better access to lifeboats. This relationship suggests a strong socio-economic influence on survival.

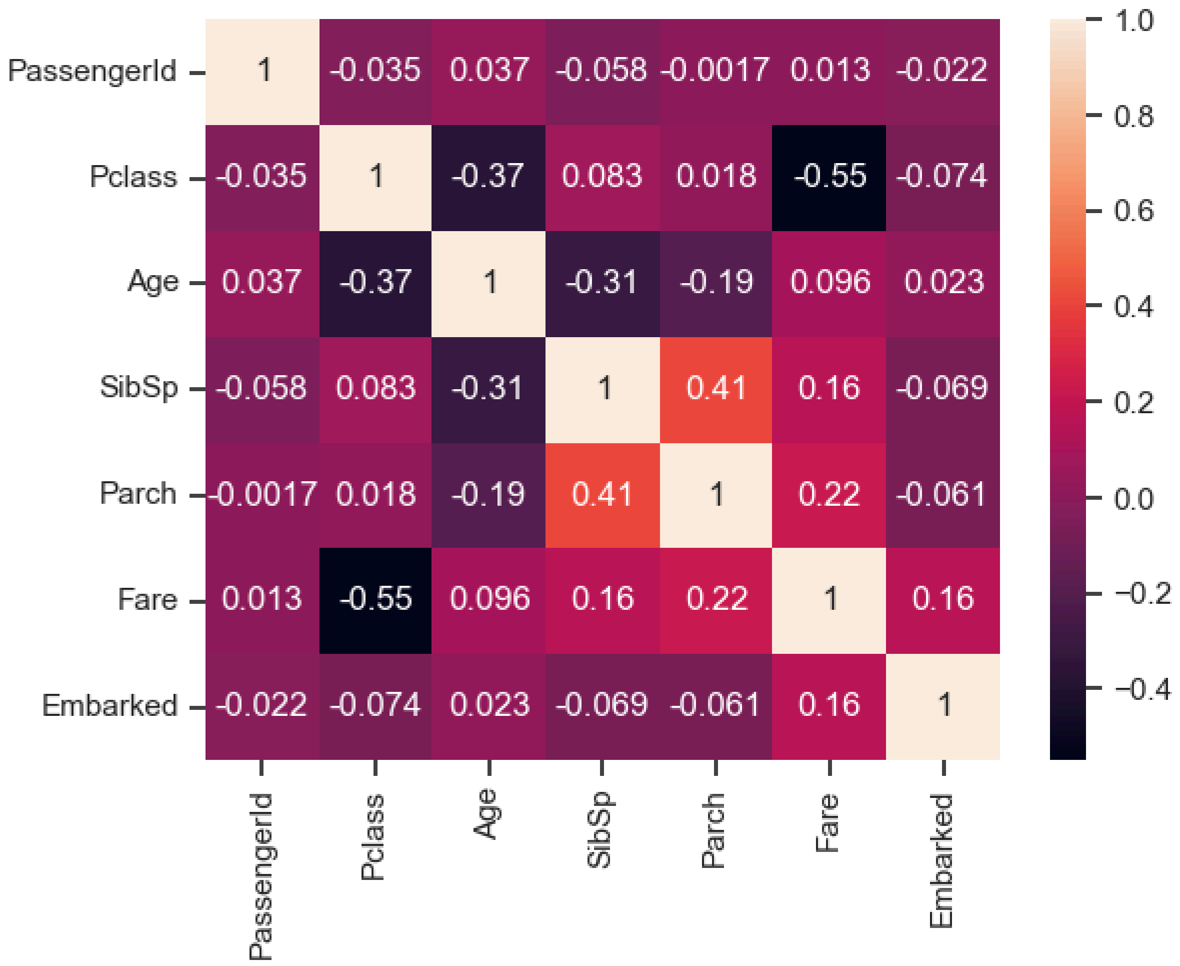
In [9]:
```
titanic['Embarked'] = titanic['Embarked'].map({"S":1, "C":2,"Q":2,"NaN":0})
Tcorrelation = titanic.corr(method='pearson', numeric_only=True)
Tcorrelation
```

Out[9]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	Embarked
PassengerId	1.000000	-0.035144	0.036847	-0.057527	-0.001652	0.012658	-0.022204
Pclass	-0.035144	1.000000	-0.369226	0.083081	0.018443	-0.549500	-0.074053
Age	0.036847	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.023233
SibSp	-0.057527	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.068734
Parch	-0.001652	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.060814
Fare	0.012658	-0.549500	0.096067	0.159651	0.216225	1.000000	0.162184
Embarked	-0.022204	-0.074053	0.023233	-0.068734	-0.060814	0.162184	1.000000

In [10]:
```
sns.heatmap(Tcorrelation,xticklabels=Tcorrelation.columns,
 yticklabels=Tcorrelation.columns, annot=True)
```

Out[10]: <Axes: >



Correlation Heatmap:

The correlation matrix visualized using a heatmap shows that fare and passenger class are strongly negatively correlated, meaning that first-class tickets were significantly more expensive. There is also a weak positive correlation between survival and fare price.

```
In [ ]:
```

Time Series Analysis: OPSD_germany_daily Dataset

In [11]:
```
import pandas as pd
import numpy as np

# load time series dataset
df_power = pd.read_csv("opsd_germany_daily.csv")
print(df_power.columns)
df_power.tail(10)
```

```
Index(['Date', 'Consumption', 'Wind', 'Solar', 'Wind+Solar'], dtype='object')
```

Out[11]:

	Date	Consumption	Wind	Solar	Wind+Solar
4373	2017-12-22	1423.23782	228.773	10.065	238.838
4374	2017-12-23	1272.17085	748.074	8.450	756.524
4375	2017-12-24	1141.75730	812.422	9.949	822.371
4376	2017-12-25	1111.28338	587.810	15.765	603.575
4377	2017-12-26	1130.11683	717.453	30.923	748.376
4378	2017-12-27	1263.94091	394.507	16.530	411.037
4379	2017-12-28	1299.86398	506.424	14.162	520.586
4380	2017-12-29	1295.08753	584.277	29.854	614.131
4381	2017-12-30	1215.44897	721.247	7.467	728.714
4382	2017-12-31	1107.11488	721.176	19.980	741.156

In [12]:
```
print(df_power.shape)
print(df_power.dtypes)
```

```
(4383, 5)
Date             object
Consumption     float64
Wind            float64
Solar           float64
Wind+Solar      float64
dtype: object
```

In [13]:
```
#convert object to datetime format
df_power['Date'] = pd.to_datetime(df_power['Date'])
```

In [14]:
```
df_power = df_power.set_index('Date')
df_power.tail(3)
```

Out[14]:

Date	Consumption	Wind	Solar	Wind+Solar
2017-12-29	1295.08753	584.277	29.854	614.131
2017-12-30	1215.44897	721.247	7.467	728.714
2017-12-31	1107.11488	721.176	19.980	741.156

In [15]:
```
df_power.index
```

Out[15]:
```
DatetimeIndex(['2006-01-01', '2006-01-02', '2006-01-03', '2006-01-04',
               '2006-01-05', '2006-01-06', '2006-01-07', '2006-01-08',
               '2006-01-09', '2006-01-10',
               ...
               '2017-12-22', '2017-12-23', '2017-12-24', '2017-12-25',
               '2017-12-26', '2017-12-27', '2017-12-28', '2017-12-29',
               '2017-12-30', '2017-12-31'],
              dtype='datetime64[ns]', name='Date', length=4383, freq=None)
```

In [16]:
```
# Add columns with year, month, and weekday name
df_power['Year'] = df_power.index.year
df_power['Month'] = df_power.index.month
df_power['Weekday Name'] = df_power.index.day_name
```

In [17]:
```
df_power.sample(5, random_state=0)
```

Out[17]:

Date	Consumption	Wind	Solar	Wind+Solar	Year	Month	Weekday Name
2008-08-23	1152.011	NaN	NaN	NaN	2008	8	<bound method _inherit_from_data.<locals>.meth...
2013-08-08	1291.984	79.666	93.371	173.037	2013	8	<bound method _inherit_from_data.<locals>.meth...
2009-08-27	1281.057	NaN	NaN	NaN	2009	8	<bound method _inherit_from_data.<locals>.meth...
2015-10-02	1391.050	81.229	160.641	241.870	2015	10	<bound method _inherit_from_data.<locals>.meth...
2009-06-02	1201.522	NaN	NaN	NaN	2009	6	<bound method _inherit_from_data.<locals>.meth...

In [18]:
```
df_power.loc['2015-10-02']
```

Out[18]:
```
Consumption                                                  1391.05
Wind                                                          81.229
Solar                                                        160.641
Wind+Solar                                                    241.87
Year                                                            2015
Month                                                             10
Weekday Name    <bound method _inherit_from_data.<locals>.meth...
Name: 2015-10-02 00:00:00, dtype: object
```

In [19]:
```
df_power.loc['2017-01-01':'2017-12-30']
```

Out[19]:

Date	Consumption	Wind	Solar	Wind+Solar	Year	Month	Weekday Name
2017-01-01	1130.41300	307.125	35.291	342.416	2017	1	<bound method _inherit_from_data.<locals>.meth...
2017-01-02	1441.05200	295.099	12.479	307.578	2017	1	<bound method _inherit_from_data.<locals>.meth...
2017-01-03	1529.99000	666.173	9.351	675.524	2017	1	<bound method _inherit_from_data.<locals>.meth...
2017-01-04	1553.08300	686.578	12.814	699.392	2017	1	<bound method _inherit_from_data.<locals>.meth...
2017-01-05	1547.23800	261.758	20.797	282.555	2017	1	<bound method _inherit_from_data.<locals>.meth...
...	...	...	...	...	...	...	...
2017-12-26	1130.11683	717.453	30.923	748.376	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-27	1263.94091	394.507	16.530	411.037	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-28	1299.86398	506.424	14.162	520.586	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-29	1295.08753	584.277	29.854	614.131	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-30	1215.44897	721.247	7.467	728.714	2017	12	<bound method _inherit_from_data.<locals>.meth...

364 rows × 7 columns

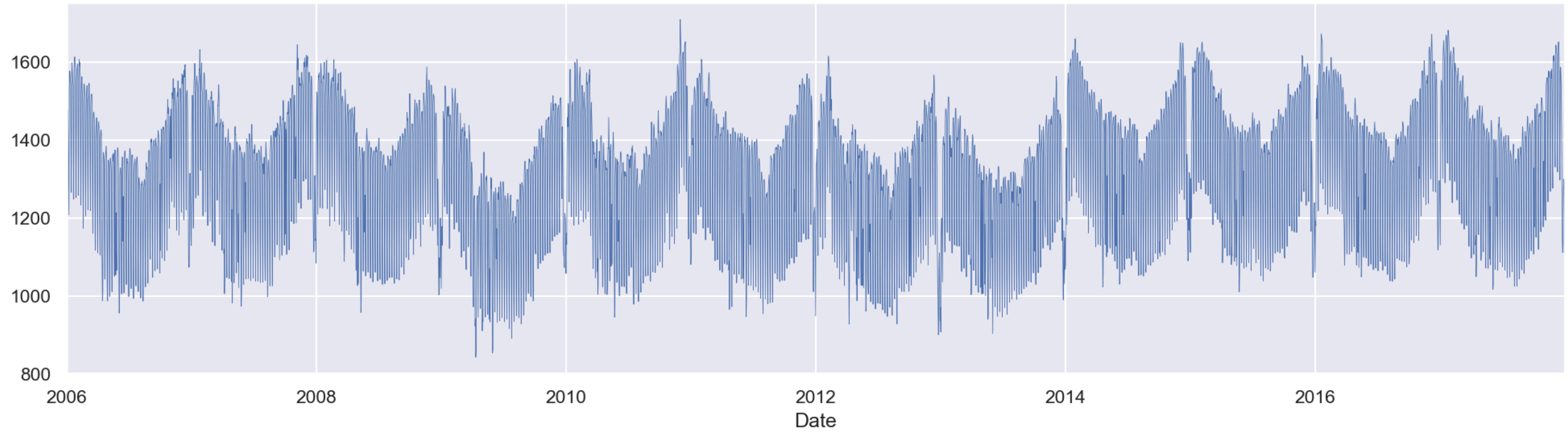
In [20]:

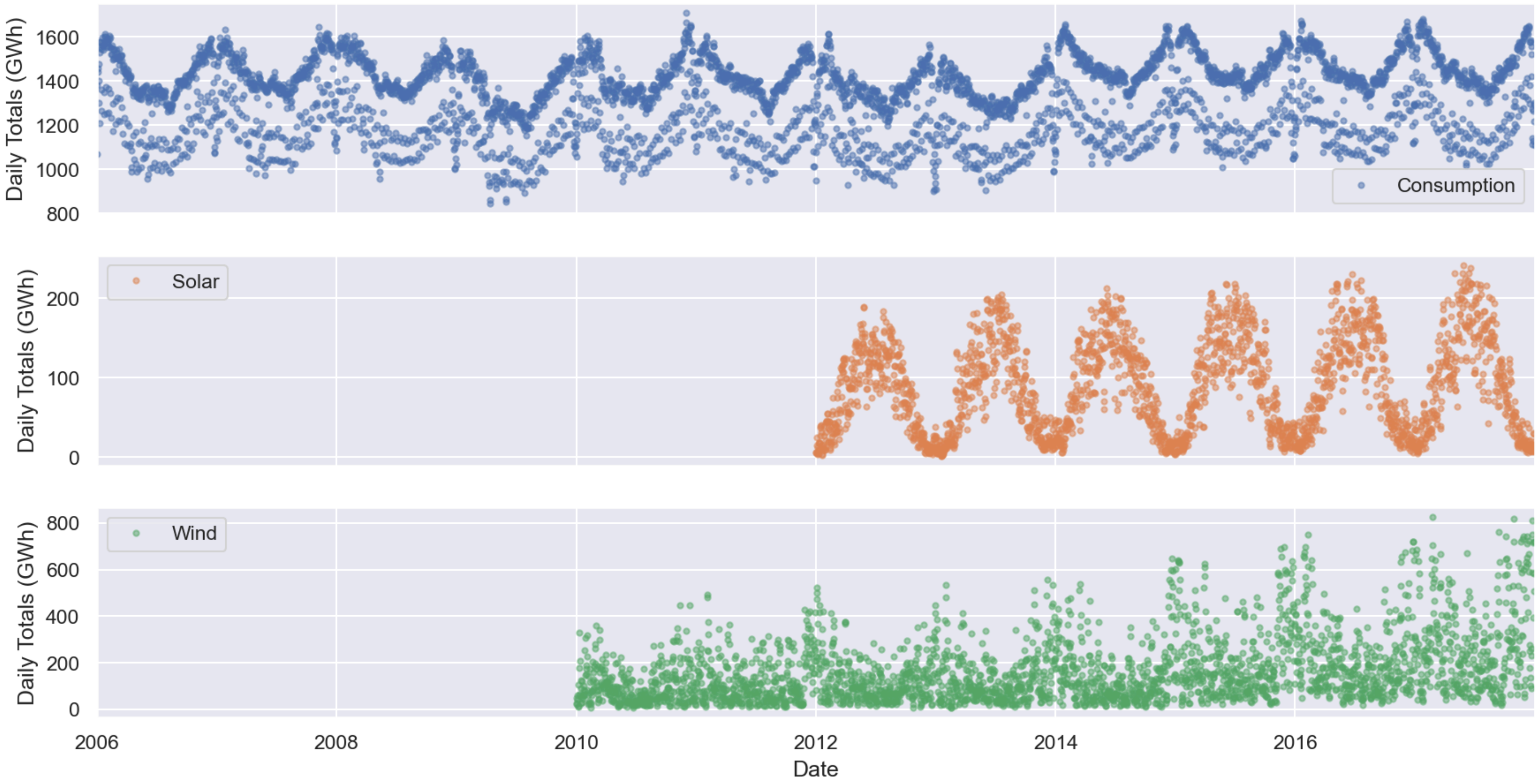
```
sns.set(rc={'figure.figsize':(16, 4)})
plt.rcParams['figure.dpi'] = 150
```

In [21]:

```
df_power['Consumption'].plot(linewidth=0.4)

cols_to_plot = ['Consumption', 'Solar', 'Wind']
axes = df_power[cols_to_plot].plot(marker='.', alpha=0.5, linestyle='None',figsize=(14, 7), subplots=True)
for ax in axes:
    ax.set_ylabel('Daily Totals (GWh)')
```


1600
1400
1200
1000
800
2006
2008
2010
2012
2014
2016
Date

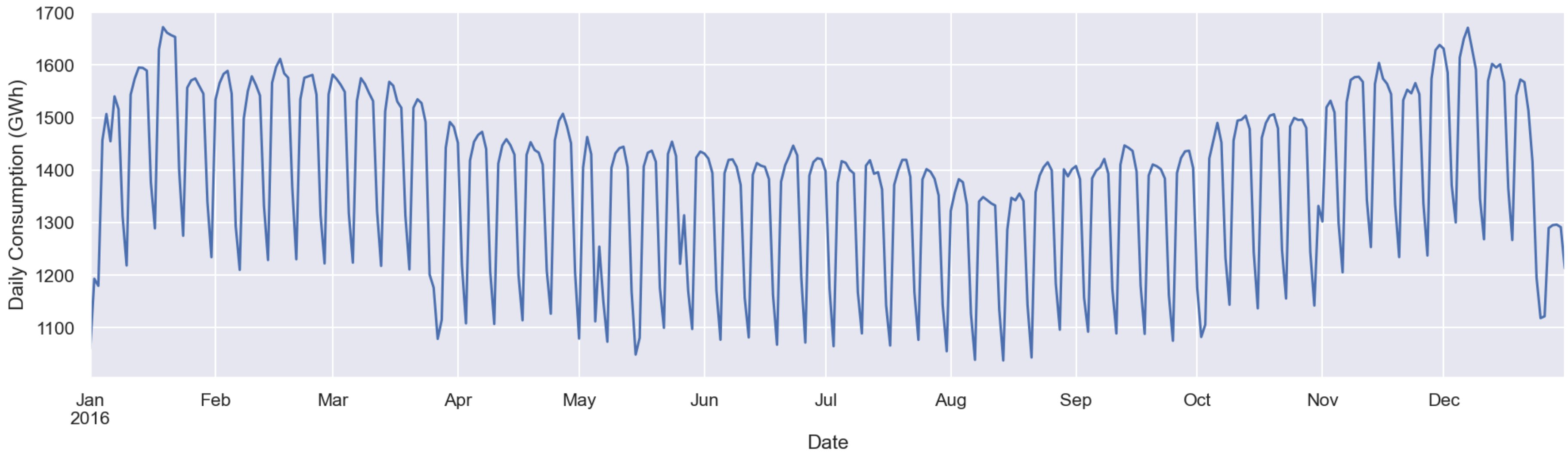


Daily Electricity Consumption Trends:

The time series plot of electricity consumption shows daily fluctuations, with visible patterns indicating periodic peaks and troughs. This suggests that energy consumption follows a predictable pattern influenced by seasons, weekdays, and special events.

In [22]:
```
ax = df_power.loc['2016', 'Consumption'].plot()
ax.set_ylabel('Daily Consumption (GWh)')
```

Out[22]: `Text(0, 0.5, 'Daily Consumption (GWh)')`

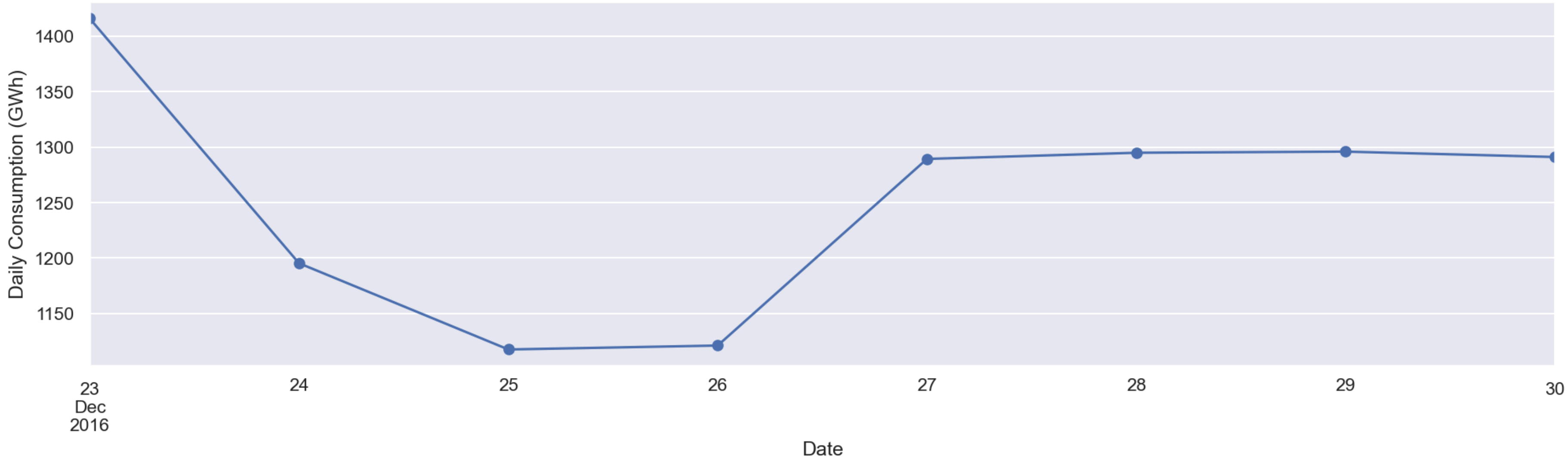


Monthly Variation in Electricity Consumption:

The boxplot analysis shows that electricity consumption is highest in winter months (December to February) and lowest in summer months (June to August). This is expected due to increased heating demand during cold months and reduced energy usage in summer.

In [23]:
```
ax = df_power.loc['2016-12-23':'2016-12-30', 'Consumption'].plot(marker='o', linestyle='-')
ax.set_ylabel('Daily Consumption (GWh)')
```

Out[23]:
```
Text(0, 0.5, 'Daily Consumption (GWh)')
```

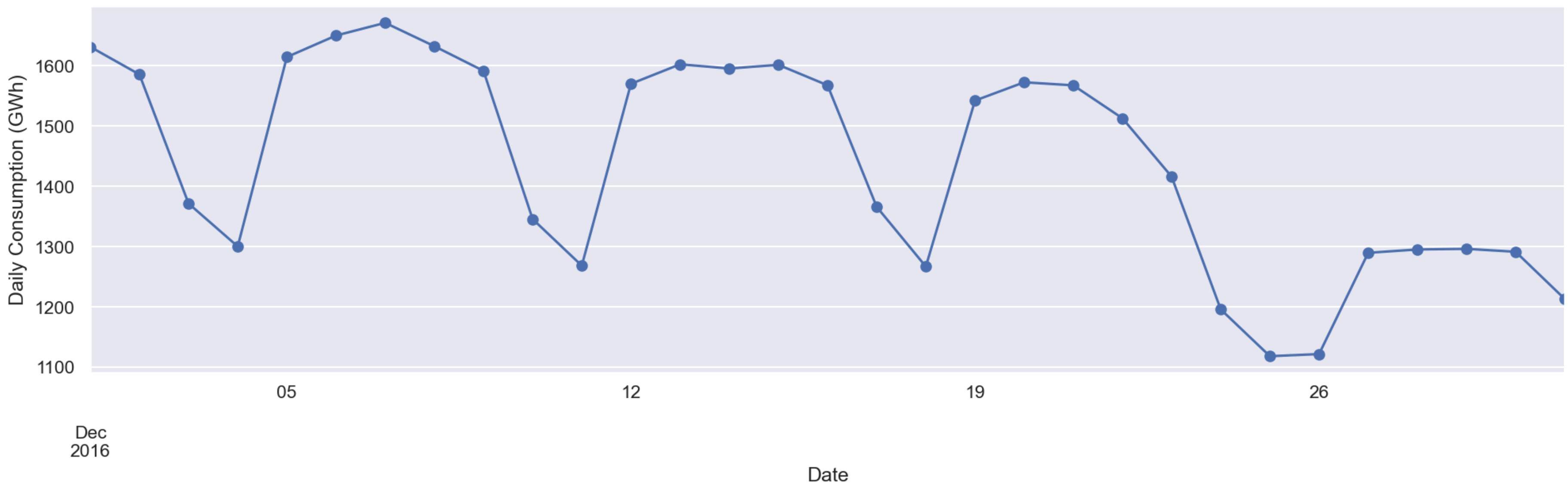


Electricity Consumption by Weekday:

The weekday boxplot reveals that energy consumption varies by day, with weekdays generally having higher consumption than weekends. This is likely due to industrial and commercial electricity usage being higher on working days.

In [24]:
```
ax = df_power.loc['2016-12', 'Consumption'].plot(marker='o', linestyle='-')
ax.set_ylabel('Daily Consumption (GWh)')
```

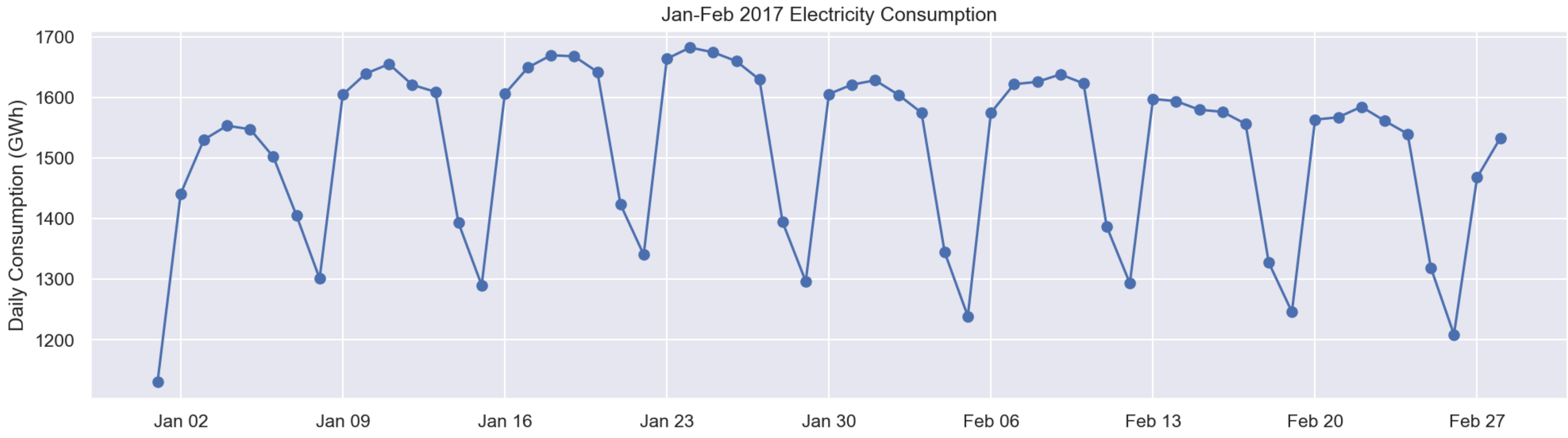
Out[24]:
```
Text(0, 0.5, 'Daily Consumption (GWh)')
```



Weekly Resampling for Trend Analysis:

Resampling the dataset to weekly averages helps smooth out short-term fluctuations and provides a clearer view of long-term trends. This technique is useful in understanding seasonal patterns and detecting anomalies.

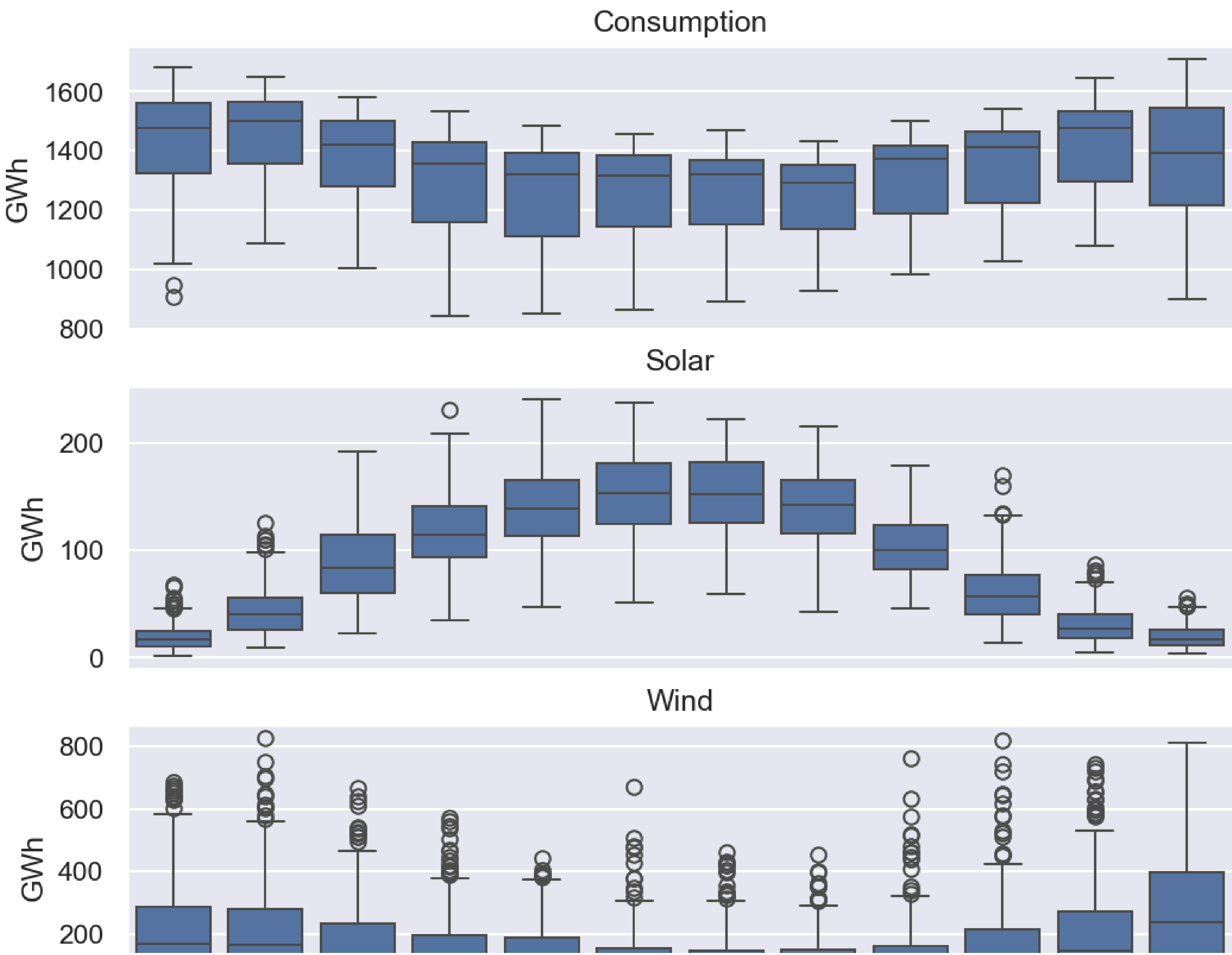
In [25]:
```
import matplotlib.dates as mdates
# plot graph
fig, ax = plt.subplots()
ax.plot(df_power.loc['2017-01':'2017-02', 'Consumption'], marker='o', linestyle='-')
ax.set_ylabel('Daily Consumption (GWh)')
ax.set_title('Jan-Feb 2017 Electricity Consumption')
# to set x-axis |major ticks to weekly interval, on Mondays
ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MONDAY))
# to set format for x-tick labels as 3-letter month name and day number
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %d'))
```

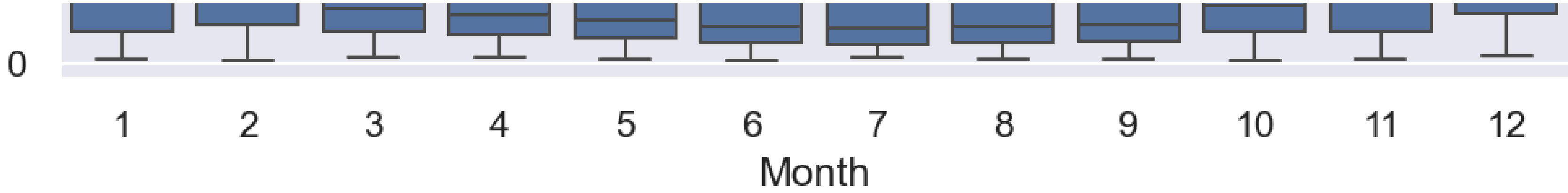


Short-Term Consumption Trends:

Focusing on specific periods (e.g., January–February 2017), we can see smaller trends within the broader time series. Daily fluctuations in energy usage can be observed more clearly when zooming in on specific months.

In [26]:
```
fig, axes = plt.subplots(3, 1, figsize=(8, 7), sharex=True)
for name, ax in zip(['Consumption', 'Solar', 'Wind'], axes):
    sns.boxplot(data=df_power, x='Month', y=name, ax=ax)
    ax.set_ylabel('GWh')
    ax.set_title(name)
    if ax != axes[-1]:
        ax.set_xlabel('')
```


Consumption
GWh
1600
1400
1200
1000
800
Solar
GWh
200
100
0
Wind
GWh
800
600
400
200

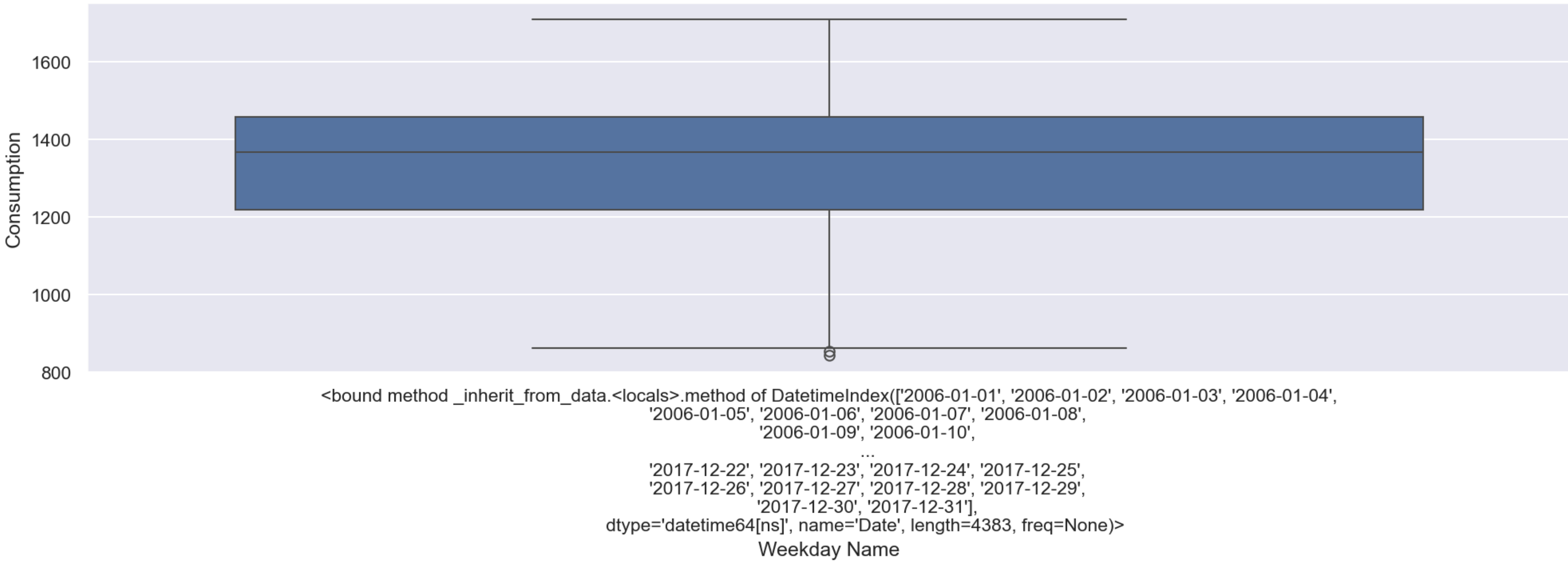


Comparison of Energy Sources (Wind, Solar, and Total Consumption):

The separate plots for wind and solar energy production indicate that renewable energy sources contribute significantly but are highly variable. Solar production follows a strong seasonal pattern, peaking in summer, while wind energy shows more irregular fluctuations.

In [27]:
```
sns.boxplot(data=df_power, x='Weekday Name', y='Consumption')
```

Out[27]:
```
<Axes: xlabel='Weekday Name', ylabel='Consumption'>
```



In [28]:
```
columns = ['Consumption', 'Wind', 'Solar', 'Wind+Solar']
power_weekly_mean = df_power[columns].resample('W').mean()
power_weekly_mean.head(10)
```

Out[28]:

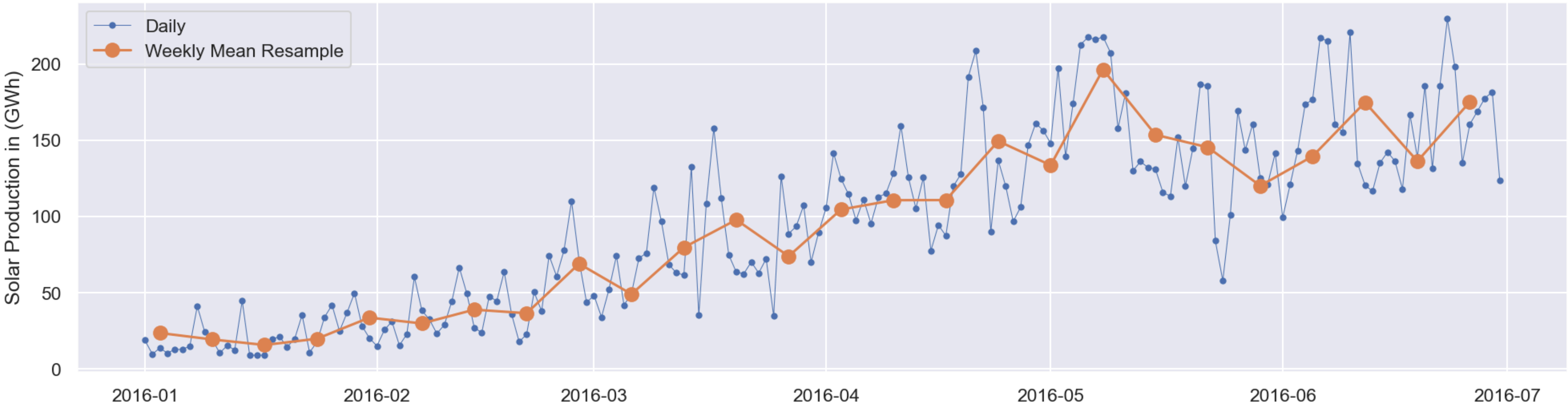
Date	Consumption	Wind	Solar	Wind+Solar
2006-01-01	1069.184000	NaN	NaN	NaN
2006-01-08	1381.300143	NaN	NaN	NaN
2006-01-15	1486.730286	NaN	NaN	NaN
2006-01-22	1490.031143	NaN	NaN	NaN
2006-01-29	1514.176857	NaN	NaN	NaN
2006-02-05	1501.403286	NaN	NaN	NaN
2006-02-12	1498.217143	NaN	NaN	NaN
2006-02-19	1446.507429	NaN	NaN	NaN
2006-02-26	1447.651429	NaN	NaN	NaN
2006-03-05	1439.727857	NaN	NaN	NaN

In [29]:
```
start, end = '2016-01', '2016-06'
```

In [30]:
```
fig, ax = plt.subplots()

ax.plot(df_power.loc[start:end, 'Solar'],
marker='.', linestyle='-', linewidth=0.5, label='Daily')
ax.plot(power_weekly_mean.loc[start:end, 'Solar'],
marker='o', markersize=8, linestyle='-', label='Weekly Mean Resample')
ax.set_ylabel('Solar Production in (GWh)')
ax.legend()
```

Out[30]: <matplotlib.legend.Legend at 0x20002f3a330>



Time Series Analysis: Bitcoin Dataset

In [31]:
```
df_bit = pd.read_csv('btc-eth-prices.csv')
df_bit.head()
```

Out[31]:

	Timestamp	Bitcoin	Ether
0	2017-04-02	1099.169125	48.55
1	2017-04-03	1141.813000	44.13
2	2017-04-04	1141.600363	44.43
3	2017-04-05	1133.079314	44.90
4	2017-04-06	1196.307937	43.23

In [32]:
```
df_bit.dtypes
```

Out[32]:
```
Timestamp     object
Bitcoin      float64
Ether        float64
dtype: object
```

In [33]:
```
df_bit.isna().sum()
```

Out[33]:
```
Timestamp    0
Bitcoin      0
Ether        3
dtype: int64
```

In [34]:
```
df_bit['Ether'] = df_bit['Ether'].fillna(df_bit['Ether'].mean())
```

In [35]:
```
df_bit.isna().sum()
```

Out[35]:
```
Timestamp    0
Bitcoin      0
Ether        0
dtype: int64
```

In [36]:
```
df_bit['Timestamp'] = pd.to_datetime(df_bit['Timestamp'])
df_bit.head()
```

Out[36]:

	Timestamp	Bitcoin	Ether
0	2017-04-02	1099.169125	48.55
1	2017-04-03	1141.813000	44.13
2	2017-04-04	1141.600363	44.43
3	2017-04-05	1133.079314	44.90
4	2017-04-06	1196.307937	43.23

In [37]:
```
df_bit.dtypes
```

Out[37]:
```
Timestamp    datetime64[ns]
Bitcoin             float64
Ether               float64
dtype: object
```

In [38]:
```python
df_bit = df_bit.set_index('Timestamp')
df_bit.tail(3)
```

Out[38]:

	Bitcoin	Ether
Timestamp		
2018-03-30	6882.531667	393.82
2018-03-31	6935.480000	394.07
2018-04-01	6794.105000	378.85

In [39]:
```python
df_bit.index
```

Out[39]:
```
DatetimeIndex(['2017-04-02', '2017-04-03', '2017-04-04', '2017-04-05',
               '2017-04-06', '2017-04-07', '2017-04-08', '2017-04-09',
               '2017-04-10', '2017-04-11',
               ...
               '2018-03-23', '2018-03-24', '2018-03-25', '2018-03-26',
               '2018-03-27', '2018-03-28', '2018-03-29', '2018-03-30',
               '2018-03-31', '2018-04-01'],
              dtype='datetime64[ns]', name='Timestamp', length=365, freq=None)
```

In [40]:
```python
df_bit.dtypes
```

Out[40]:
```
Bitcoin    float64
Ether      float64
dtype: object
```

In [41]:
```python
df_bit['Year'] = df_bit.index.year
df_bit['Month'] = df_bit.index.month
df_bit['Weekday Name'] = df_bit.index.day_name
```

In [42]:
```python
df_bit.sample(5, random_state=0)
```

Out[42]:

Timestamp	Bitcoin	Ether	Year	Month	Weekday Name
2017-07-17	2176.623488	189.97	2017	7	<bound method _inherit_from_data.<locals>.meth...
2017-12-17	19289.785000	717.71	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-05-17	1807.485062	86.98	2017	5	<bound method _inherit_from_data.<locals>.meth...
2017-04-28	1331.294429	72.42	2017	4	<bound method _inherit_from_data.<locals>.meth...
2017-06-19	2617.210263	358.20	2017	6	<bound method _inherit_from_data.<locals>.meth...

In [43]:

```
df_bit.loc['2017-04-28']
```

Out[43]:

```
Bitcoin                                                   1331.294429
Ether                                                           72.42
Year                                                             2017
Month                                                               4
Weekday Name    <bound method _inherit_from_data.<locals>.meth...
Name: 2017-04-28 00:00:00, dtype: object
```

In [44]:

```
df_bit.loc['2017-01-01':'2017-12-30']
```

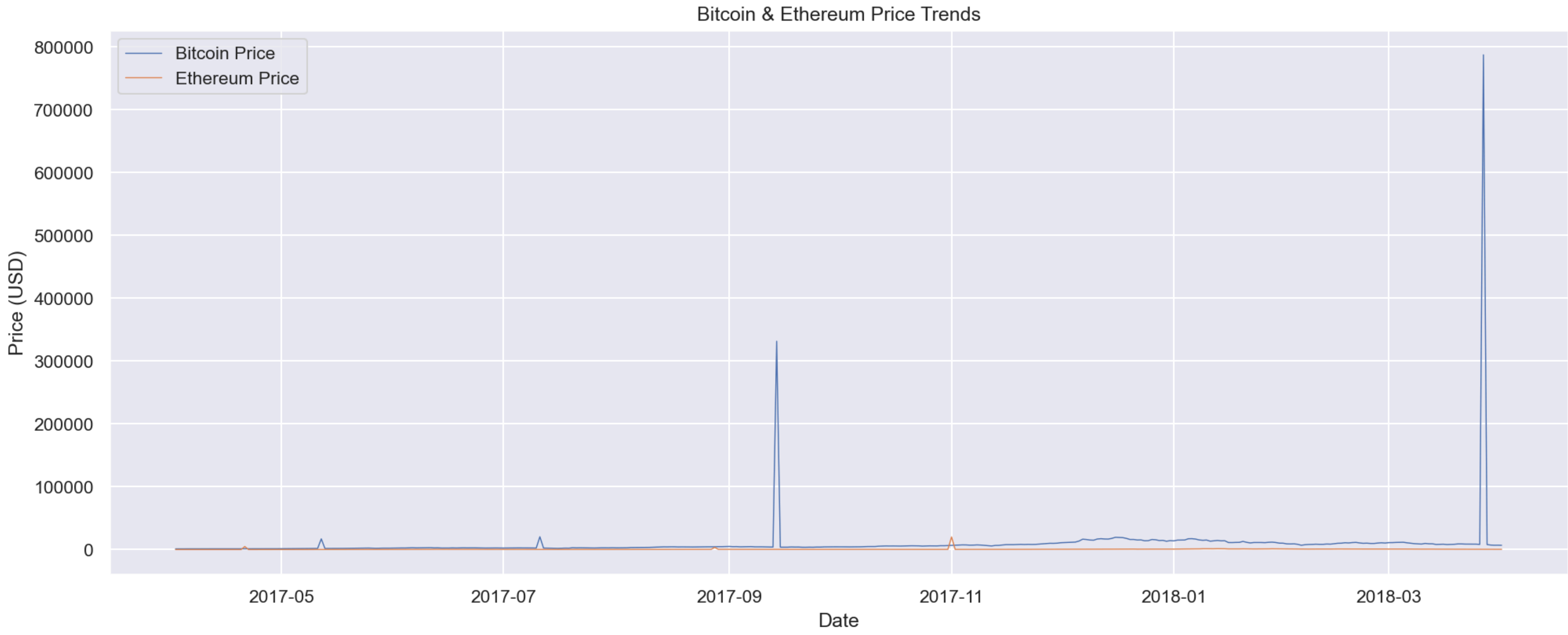
Out[44]:

	Bitcoin	Ether	Year	Month	Weekday Name
Timestamp					
2017-04-02	1099.169125	48.55	2017	4	<bound method _inherit_from_data.<locals>.meth...
2017-04-03	1141.813000	44.13	2017	4	<bound method _inherit_from_data.<locals>.meth...
2017-04-04	1141.600363	44.43	2017	4	<bound method _inherit_from_data.<locals>.meth...
2017-04-05	1133.079314	44.90	2017	4	<bound method _inherit_from_data.<locals>.meth...
2017-04-06	1196.307937	43.23	2017	4	<bound method _inherit_from_data.<locals>.meth...
...	...	...	...	...	...
2017-12-26	15999.048333	753.40	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-27	15589.321667	739.94	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-28	14380.581667	716.69	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-29	14640.140000	739.60	2017	12	<bound method _inherit_from_data.<locals>.meth...
2017-12-30	13215.574000	692.99	2017	12	<bound method _inherit_from_data.<locals>.meth...

273 rows × 5 columns

In [45]:

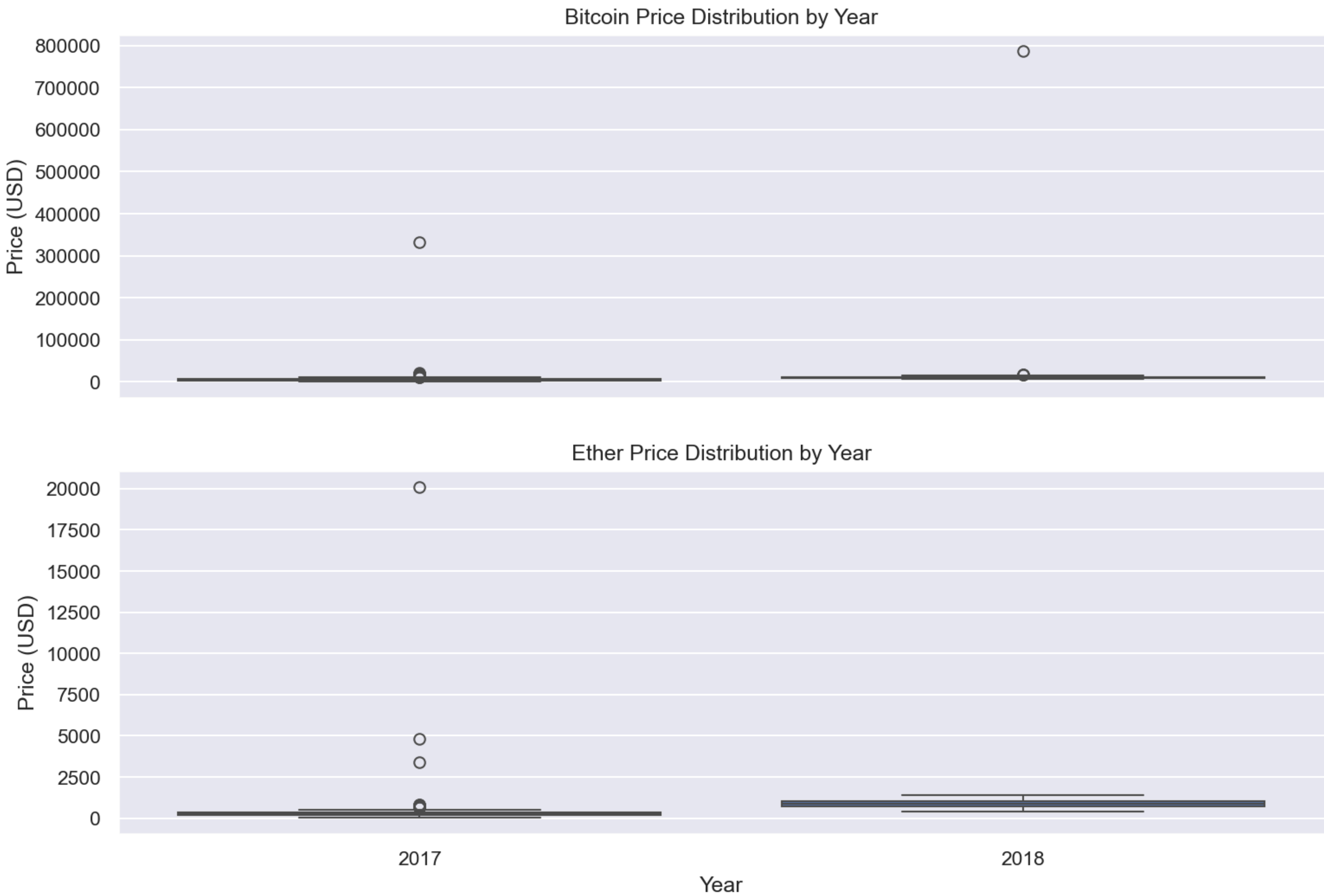
```
plt.figure(figsize=(16, 6))
plt.plot(df_bit.index, df_bit['Bitcoin'], label="Bitcoin Price", linewidth=0.7)
plt.plot(df_bit.index, df_bit['Ether'], label="Ethereum Price", linewidth=0.7)
plt.xlabel("Date")
plt.ylabel("Price (USD)")
plt.title("Bitcoin & Ethereum Price Trends")
plt.legend()
plt.show()
```



Price Trends Over Time:

The time series plot of Bitcoin and Ethereum prices shows substantial price fluctuations. Bitcoin prices exhibit more extreme spikes and crashes compared to Ethereum, indicating higher volatility.

In [46]:
```
fig, axes = plt.subplots(2, 1, figsize=(12, 8), sharex=True)
for name, ax in zip(['Bitcoin', 'Ether'], axes):
    sns.boxplot(data=df_bit, x='Year', y=name, ax=ax)
    ax.set_ylabel('Price (USD)')
    ax.set_title(f"{name} Price Distribution by Year")
plt.show()
```

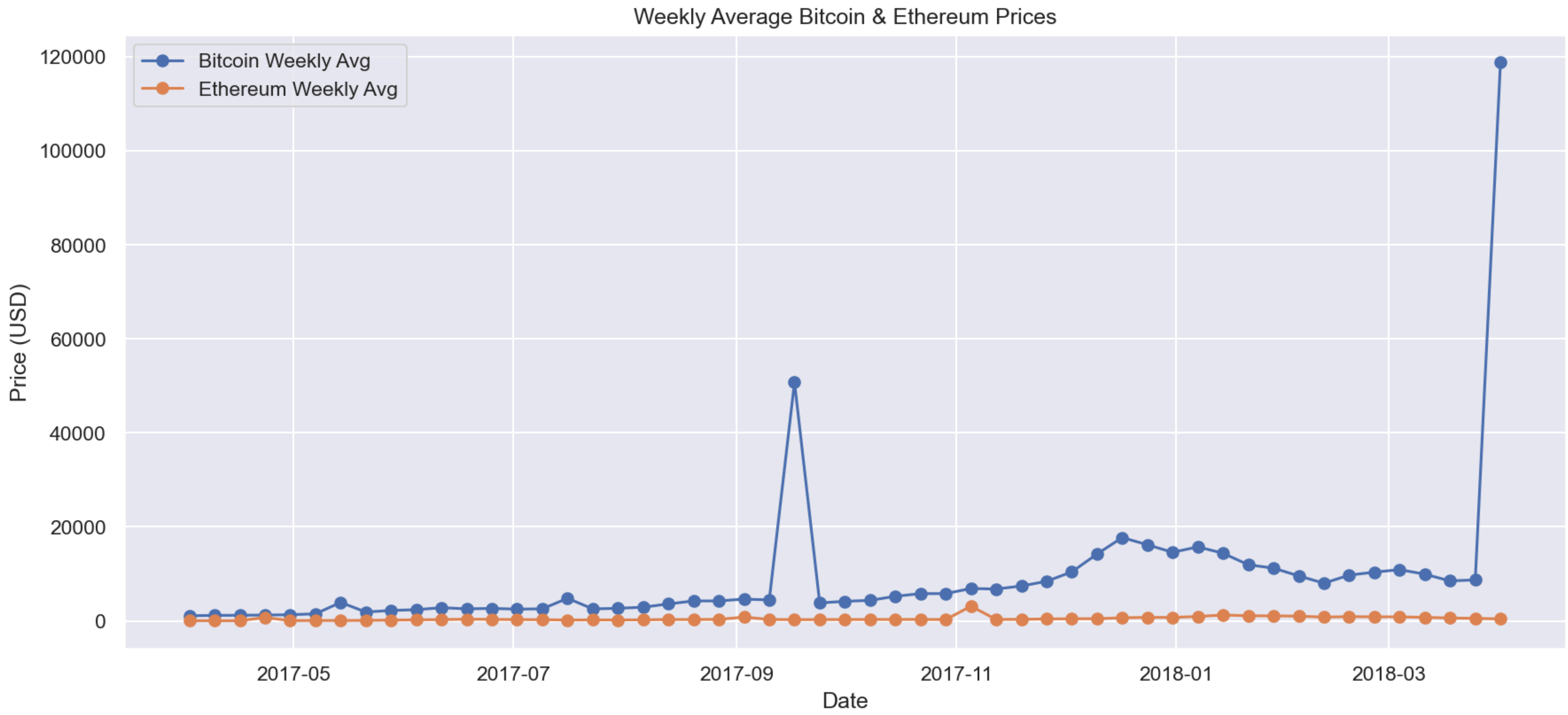


Yearly Price Distribution:

The boxplot analysis of price distribution across different years highlights that Bitcoin experienced significant price surges, especially in late 2017. Ethereum also showed an upward trend but with less drastic spikes.

In [47]:
```
# Resample data to weekly average
weekly_prices = df_bit[['Bitcoin', 'Ether']].resample('W').mean()

# Plot weekly averages
plt.figure(figsize=(14, 6))
plt.plot(weekly_prices.index, weekly_prices['Bitcoin'], marker='o', linestyle='-', label='Bitcoin Weekly Avg')
plt.plot(weekly_prices.index, weekly_prices['Ether'], marker='o', linestyle='-', label='Ethereum Weekly Avg')
plt.xlabel("Date")
plt.ylabel("Price (USD)")
plt.title("Weekly Average Bitcoin & Ethereum Prices")
plt.legend()
plt.show()
```



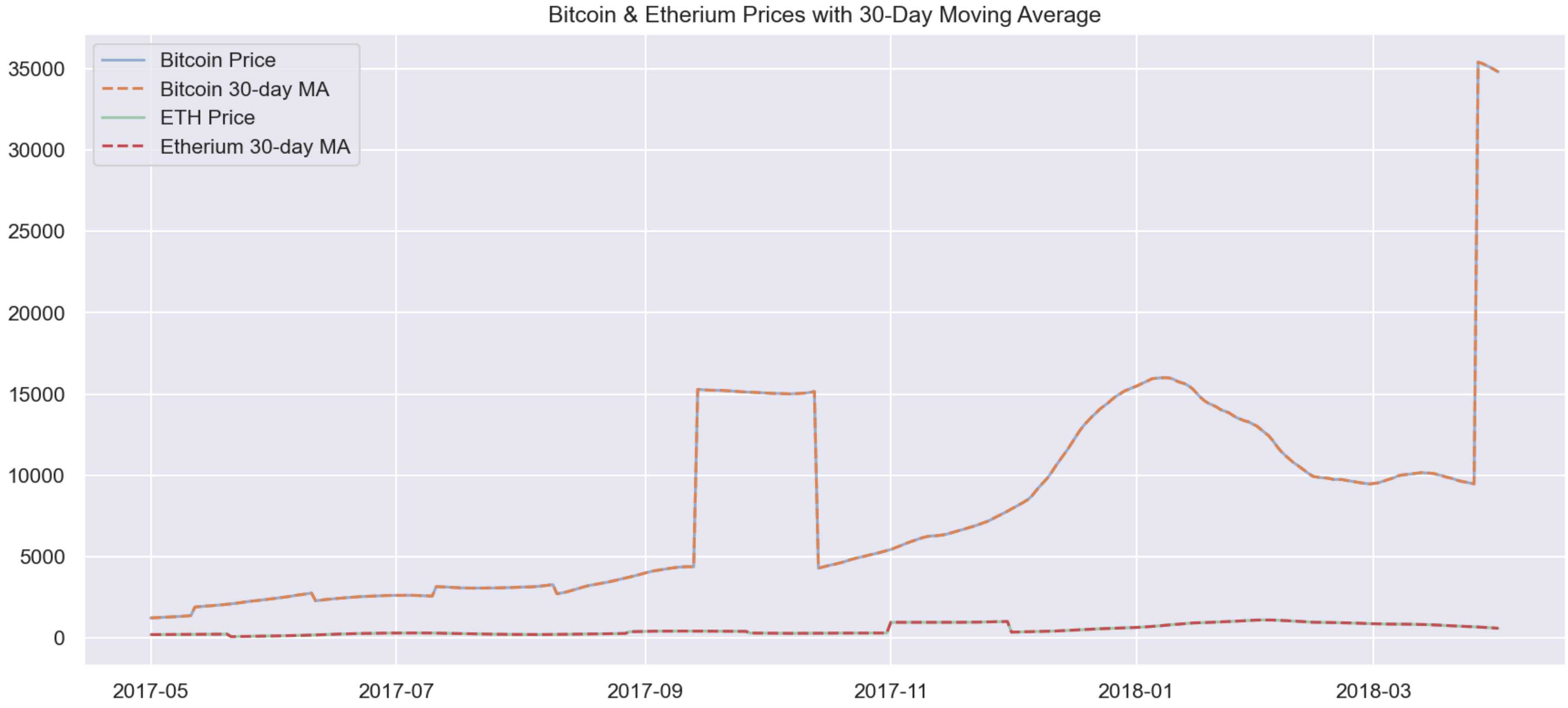
Weekly Resampling for Smoother Trend Analysis:

The weekly average resampling smooths out the daily price volatility, allowing us to identify general trends more easily. This technique helps in visualizing price cycles and long-term movements.

In [48]:
```
# Moving average
df_bit['Bitcoin'] = df_bit['Bitcoin'].rolling(window=30).mean()
df_bit['Ether'] = df_bit['Ether'].rolling(window=30).mean()

plt.figure(figsize=(14, 6))
plt.plot(df_bit.index, df_bit['Bitcoin'], alpha=0.5, label='Bitcoin Price')
plt.plot(df_bit.index, df_bit['Bitcoin'], label='Bitcoin 30-day MA', linestyle='--')
```

```
plt.plot(df_bit.index, df_bit['Ether'], alpha=0.5, label='ETH Price')
plt.plot(df_bit.index, df_bit['Ether'], label='Etherium 30-day MA', linestyle='--')
plt.legend()
plt.title('Bitcoin & Etherium Prices with 30-Day Moving Average')
plt.show()
```

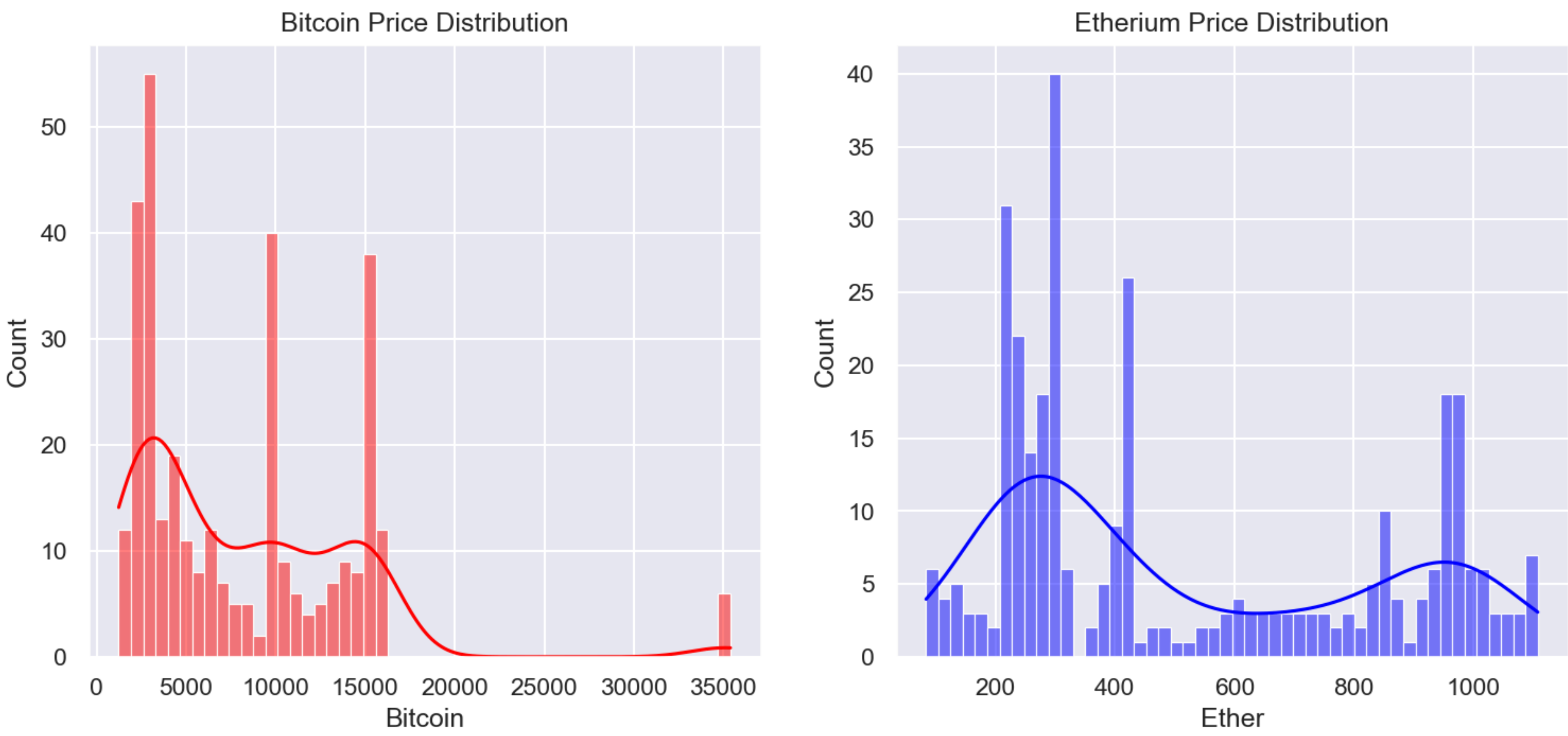


30-Day Moving Average for Trend Detection:

Applying a rolling 30-day moving average to Bitcoin and Ethereum prices reduces short-term fluctuations and highlights long-term trends. The smoothed-out price trends make it easier to identify bullish (rising) and bearish (falling) market trends.

In [49]:
```
# Histogram
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
```

```
sns.histplot(df_bit['Bitcoin'], bins=50, kde=True, ax=ax[0], color='red')
sns.histplot(df_bit['Ether'], bins=50, kde=True, ax=ax[1], color='blue')
ax[0].set_title('Bitcoin Price Distribution')
ax[1].set_title('Etherium Price Distribution')
plt.show()
```



Price Distributions of Bitcoin and Ethereum:

Histograms of Bitcoin and Ethereum prices show that Bitcoin prices are more widely spread out, indicating greater volatility. Ethereum, in contrast, has a more concentrated price distribution, suggesting relatively lower volatility.

```
In [ ]:
```