Power Consumption Analysis

About Dataset

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [10]:
```
# Load the dataset
df = pd.read_csv('powerconsumption.csv')
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52416 entries, 0 to 52415
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Datetime                52416 non-null  object
 1   Temperature             52416 non-null  float64
 2   Humidity                52416 non-null  float64
 3   WindSpeed               52416 non-null  float64
 4   GeneralDiffuseFlows     52416 non-null  float64
 5   DiffuseFlows            52416 non-null  float64
 6   PowerConsumption_Zone1  52416 non-null  float64
 7   PowerConsumption_Zone2  52416 non-null  float64
 8   PowerConsumption_Zone3  52416 non-null  float64
dtypes: float64(8), object(1)
memory usage: 3.6+ MB
```

In [14]:
```
df.shape
```

Out[14]: (52416, 9)

In [16]:
```
data.describe()
```

Out[16]:

	Temperature	Humidity	WindSpeed	GeneralDiffuseFlows	DiffuseFlows	Pc
count	52416.000000	52416.000000	52416.000000	52416.000000	52416.000000	
mean	18.810024	68.259518	1.959489	182.696614	75.028022	
std	5.815476	15.551177	2.348862	264.400960	124.210949	
min	3.247000	11.340000	0.050000	0.004000	0.011000	
25%	14.410000	58.310000	0.078000	0.062000	0.122000	
50%	18.780000	69.860000	0.086000	5.035500	4.456000	
75%	22.890000	81.400000	4.915000	319.600000	101.000000	
max	40.010000	94.800000	6.483000	1163.000000	936.000000	

In [21]:
```
# Convert 'Datetime' column to datetime type
df['Datetime'] = pd.to_datetime(df['Datetime'])
```

In [23]:
```
# Create helper columns
df['Date'] = df['Datetime'].dt.date
df['Hour'] = df['Datetime'].dt.hour
df['Day'] = df['Datetime'].dt.day_name()
df['Month'] = df['Datetime'].dt.month_name()
df['Week'] = df['Datetime'].dt.isocalendar().week
df['TotalConsumption'] = df[['PowerConsumption_Zone1', 'PowerConsumption_Zone2',
```

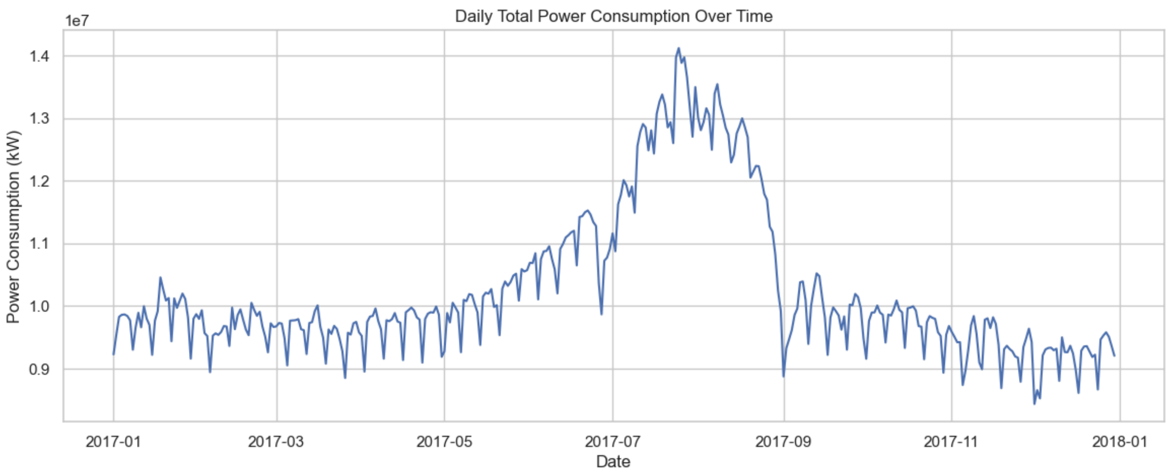
In [25]:
```
# Set style for plots
sns.set(style='whitegrid')
```

In [27]:
```
# Store all plots and outputs in a dictionary
insights_outputs = {}
```

Descriptive & Temporal Analysis

Trend of total power consumption over time (daily)

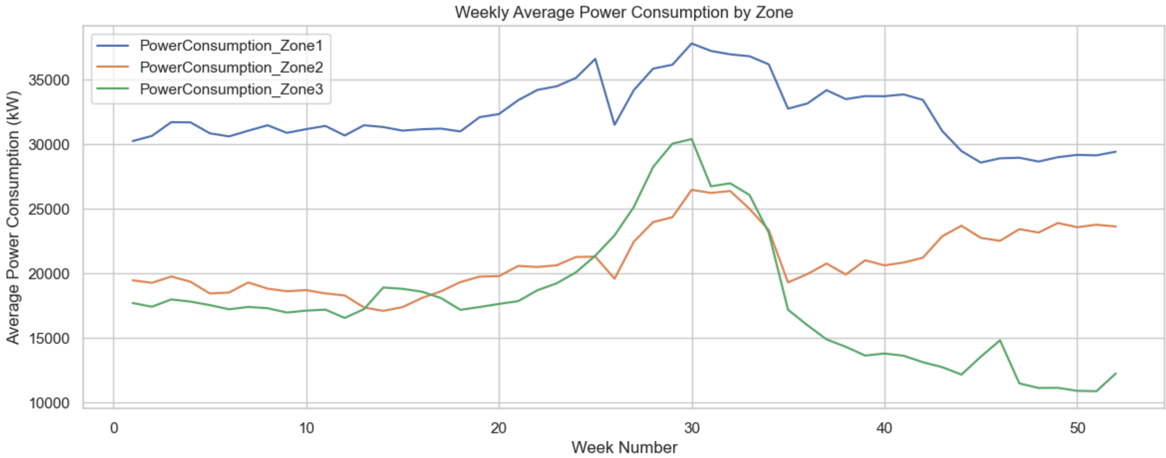
In [31]:
```
daily_total = df.groupby('Date')['TotalConsumption'].sum()
fig1, ax1 = plt.subplots(figsize=(14, 5))
daily_total.plot(ax=ax1)
ax1.set_title('Daily Total Power Consumption Over Time')
ax1.set_ylabel('Power Consumption (kW)')
ax1.set_xlabel('Date')
insights_outputs["1. Daily Total Power Consumption"] = fig1
```



- Power usage shows daily fluctuations, possibly due to operational changes or varying weather.
- Upward or downward trends can help detect seasonal or monthly consumption behavior.
- Potential to identify spikes on specific dates (e.g., holidays, maintenance days).

Compare power consumption across three zones over time (weekly average)

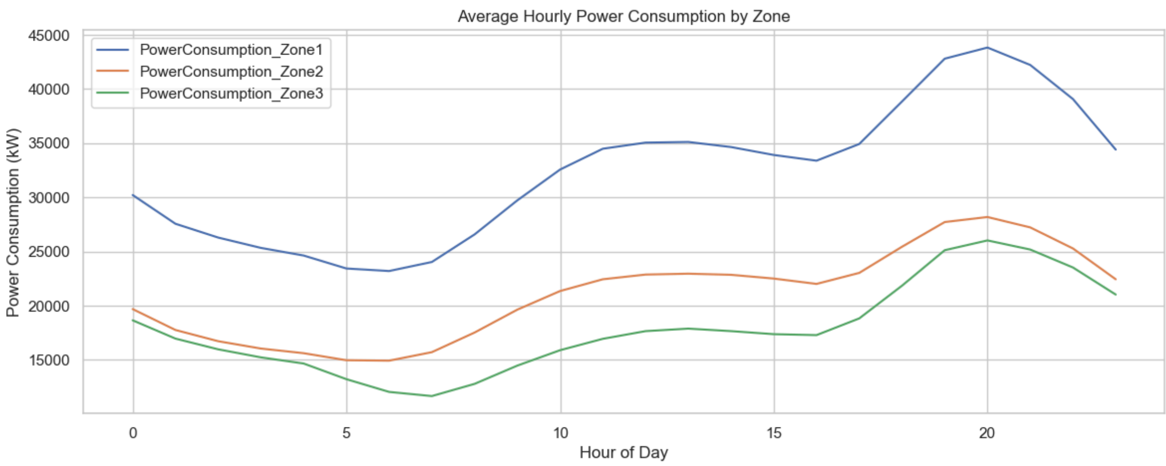
In [34]:
```
weekly_avg = df.groupby('Week')[['PowerConsumption_Zone1', 'PowerConsumption_Zon
fig2, ax2 = plt.subplots(figsize=(14, 5))
weekly_avg.plot(ax=ax2)
ax2.set_title('Weekly Average Power Consumption by Zone')
ax2.set_ylabel('Average Power Consumption (kW)')
ax2.set_xlabel('Week Number')
insights_outputs["2. Weekly Avg Zone-wise Power Consumption"] = fig2
```



- Zone 1 consistently consumes the most power, followed by Zone 2 and Zone 3.
- Weekly averages reveal stable patterns, with occasional surges indicating increased activity.
- This insight is useful for zone-wise energy planning and efficiency optimization.

Peak consumption hours during the day for each zone

In [37]:
```
hourly_avg = df.groupby('Hour')[['PowerConsumption_Zone1', 'PowerConsumption_Zon
fig3, ax3 = plt.subplots(figsize=(14, 5))
hourly_avg.plot(ax=ax3)
ax3.set_title('Average Hourly Power Consumption by Zone')
ax3.set_xlabel('Hour of Day')
ax3.set_ylabel('Power Consumption (kW)')
insights_outputs["3. Hourly Peak Consumption by Zone"] = fig3
```

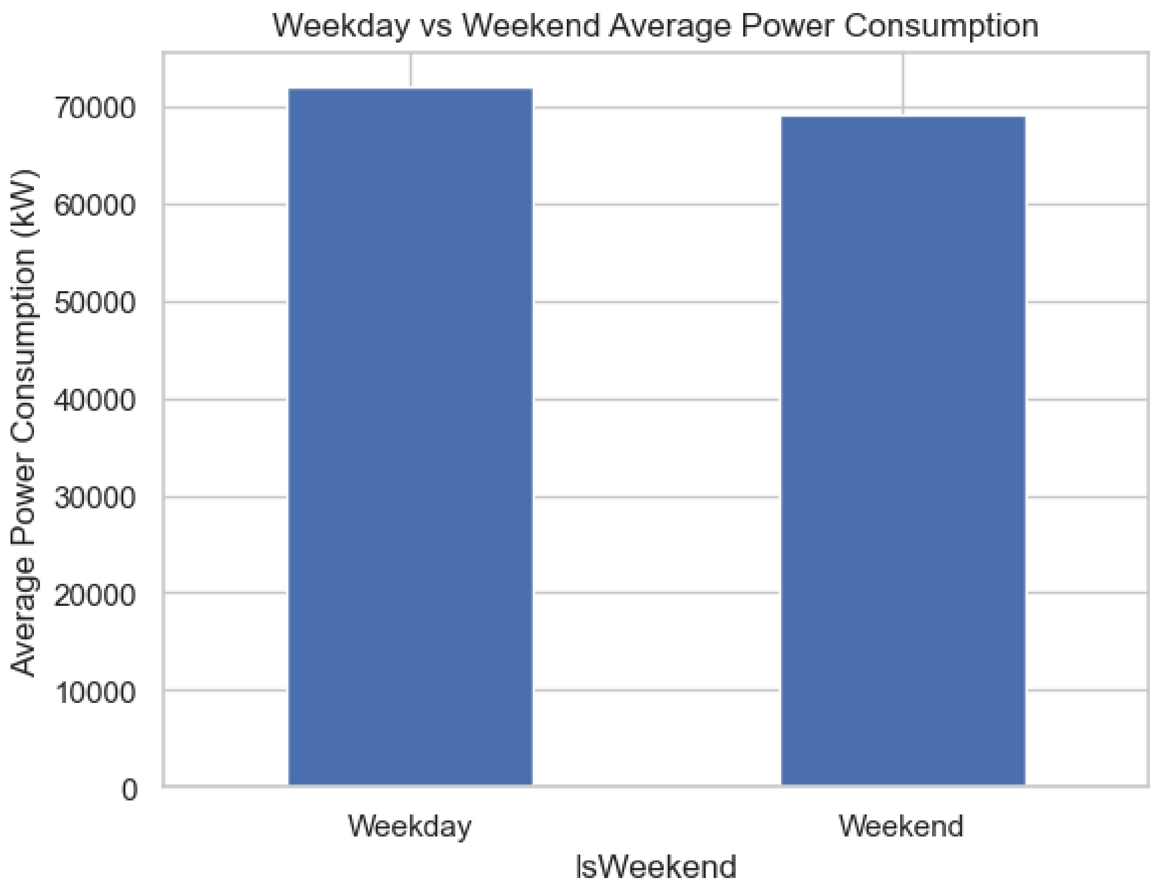


- Peak usage typically occurs during working hours (8 AM to 6 PM).
- Early morning and late-night hours show significantly lower consumption, suggesting downtime or idle hours.

- Helps in load balancing and scheduling high-energy tasks.

Weekday vs weekend usage

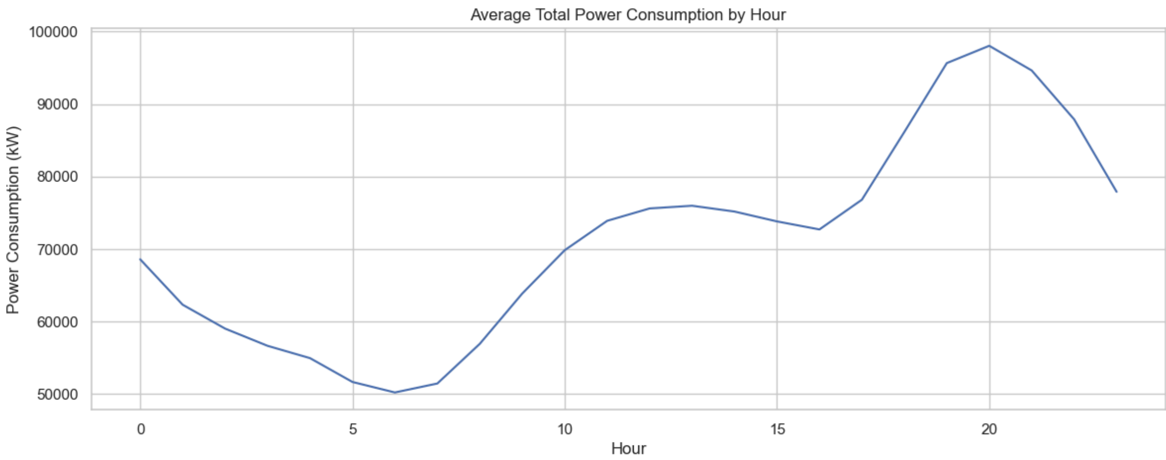
In [40]:
```
df['IsWeekend'] = df['Day'].isin(['Saturday', 'Sunday'])
weekend_avg = df.groupby('IsWeekend')['TotalConsumption'].mean()
fig4, ax4 = plt.subplots()
weekend_avg.plot(kind='bar', ax=ax4)
ax4.set_title('Weekday vs Weekend Average Power Consumption')
ax4.set_xticklabels(['Weekday', 'Weekend'], rotation=0)
ax4.set_ylabel('Average Power Consumption (kW)')
insights_outputs["4. Weekday vs Weekend Usage"] = fig4
```



- Higher average power consumption on weekdays compared to weekends.
- Implies more activity during the workweek, while weekends reflect reduced operational demand.
- Useful for differentiating load profiles and optimizing tariffs.

Time of day usage patterns

In [43]:
```
fig5, ax5 = plt.subplots(figsize=(14, 5))
df.groupby('Hour')['TotalConsumption'].mean().plot(ax=ax5)
ax5.set_title('Average Total Power Consumption by Hour')
ax5.set_xlabel('Hour')
ax5.set_ylabel('Power Consumption (kW)')
insights_outputs["5. Hourly Pattern"] = fig5
```



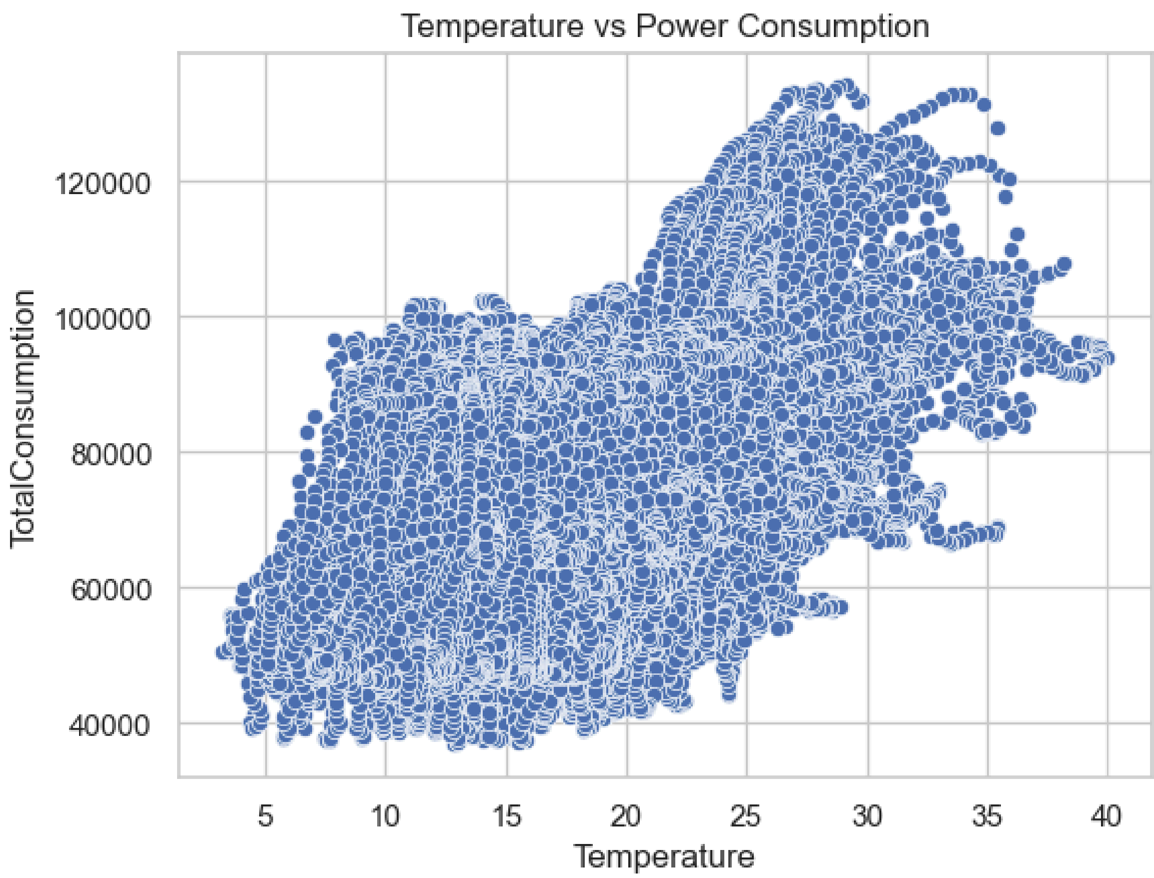
- A clear daily usage cycle is visible – ramp-up in the morning, peak during midday, and drop at night.
- Helps identify base load and plan power allocation efficiently.
- Can be used to set up energy-saving strategies during low-usage hours.

In [53]:
```
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
```

Weather vs Power Usage

Correlation between temperature and power consumption

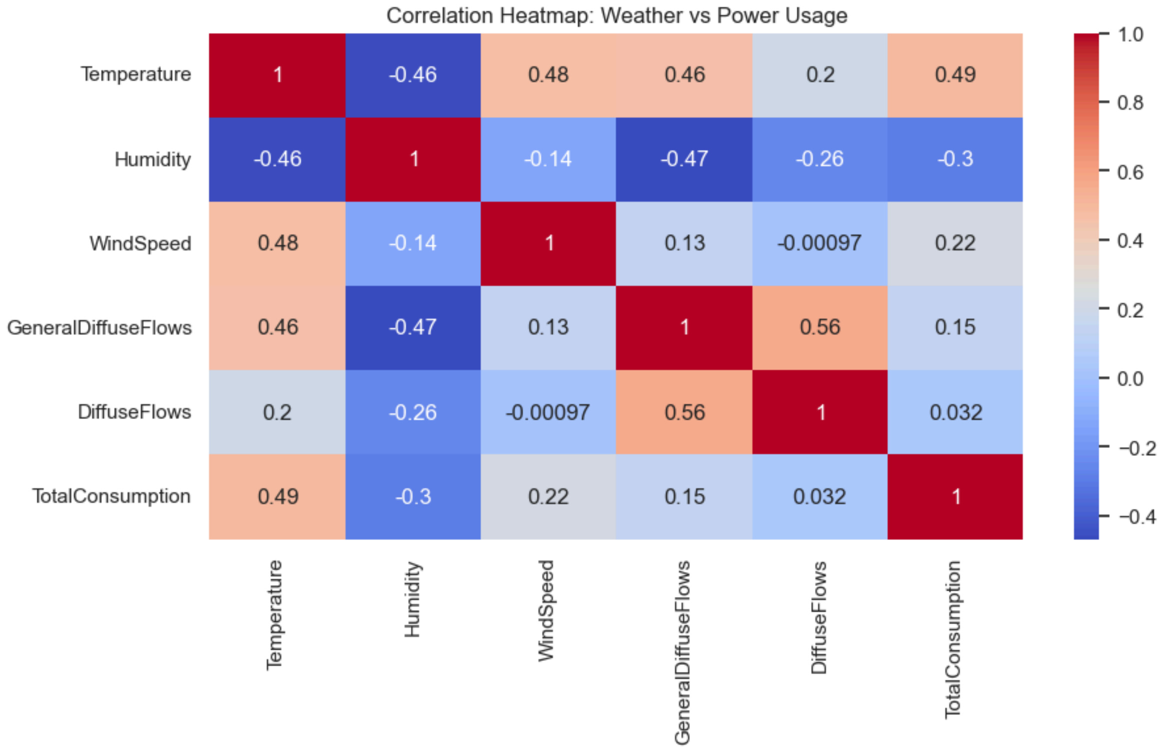
In [59]:
```
fig6, ax6 = plt.subplots()
sns.scatterplot(data=df, x='Temperature', y='TotalConsumption', ax=ax6)
ax6.set_title('Temperature vs Power Consumption')
insights_outputs["6. Temperature vs Power Usage"] = fig6
```



- A positive correlation exists between temperature and power usage.
- Higher temperatures likely increase cooling demands, driving up power consumption.
- Indicates temperature is a key driver of energy demand.

Humidity and wind speed impact

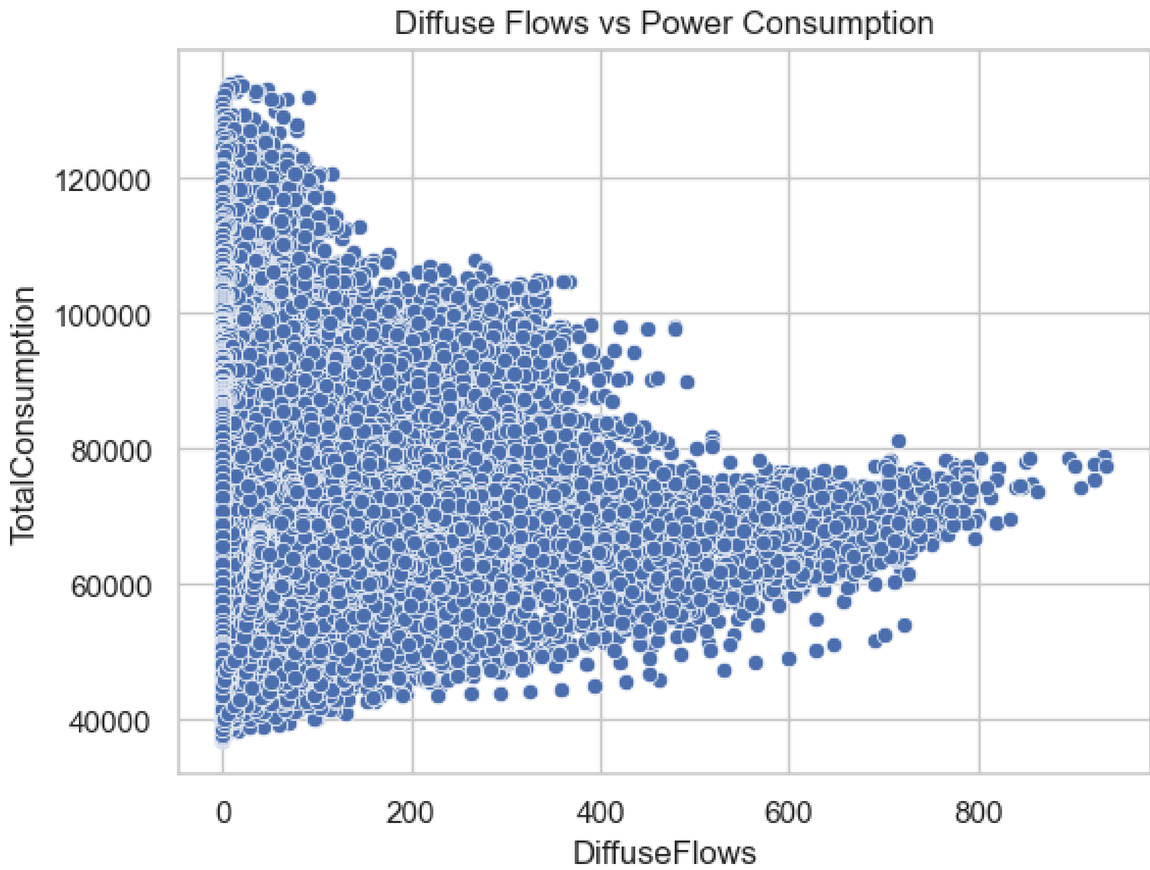
In [62]:
```
fig7, ax7 = plt.subplots(figsize=(10, 5))
sns.heatmap(df[['Temperature', 'Humidity', 'WindSpeed', 'GeneralDiffuseFlows', '
ax7.set_title('Correlation Heatmap: Weather vs Power Usage')
insights_outputs["7. Correlation Heatmap"] = fig7
```



- Temperature and Humidity are moderately correlated with total consumption.
- Diffuse Flows and Wind Speed show weaker correlations.
- Helps identify which environmental factors significantly influence power demand.

Solar irradiance impact (DiffuseFlows)

In [65]:
```
fig8, ax8 = plt.subplots()
sns.scatterplot(data=df, x='DiffuseFlows', y='TotalConsumption', ax=ax8)
ax8.set_title('Diffuse Flows vs Power Consumption')
insights_outputs["8. Diffuse Flows vs Power"] = fig8
```

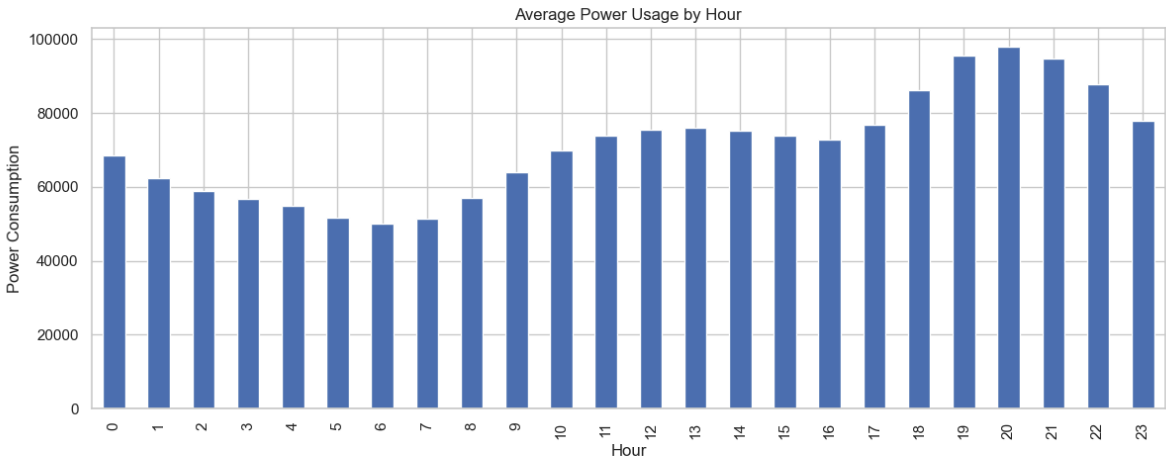


- Slight negative trend: higher Diffuse Flows (solar irradiance) may reduce power usage.
- Suggests potential for natural lighting or solar energy impact on reducing grid power use.
- Can be used to optimize lighting and solar integration in facilities.

Statistical Insights

Hourly average power usage

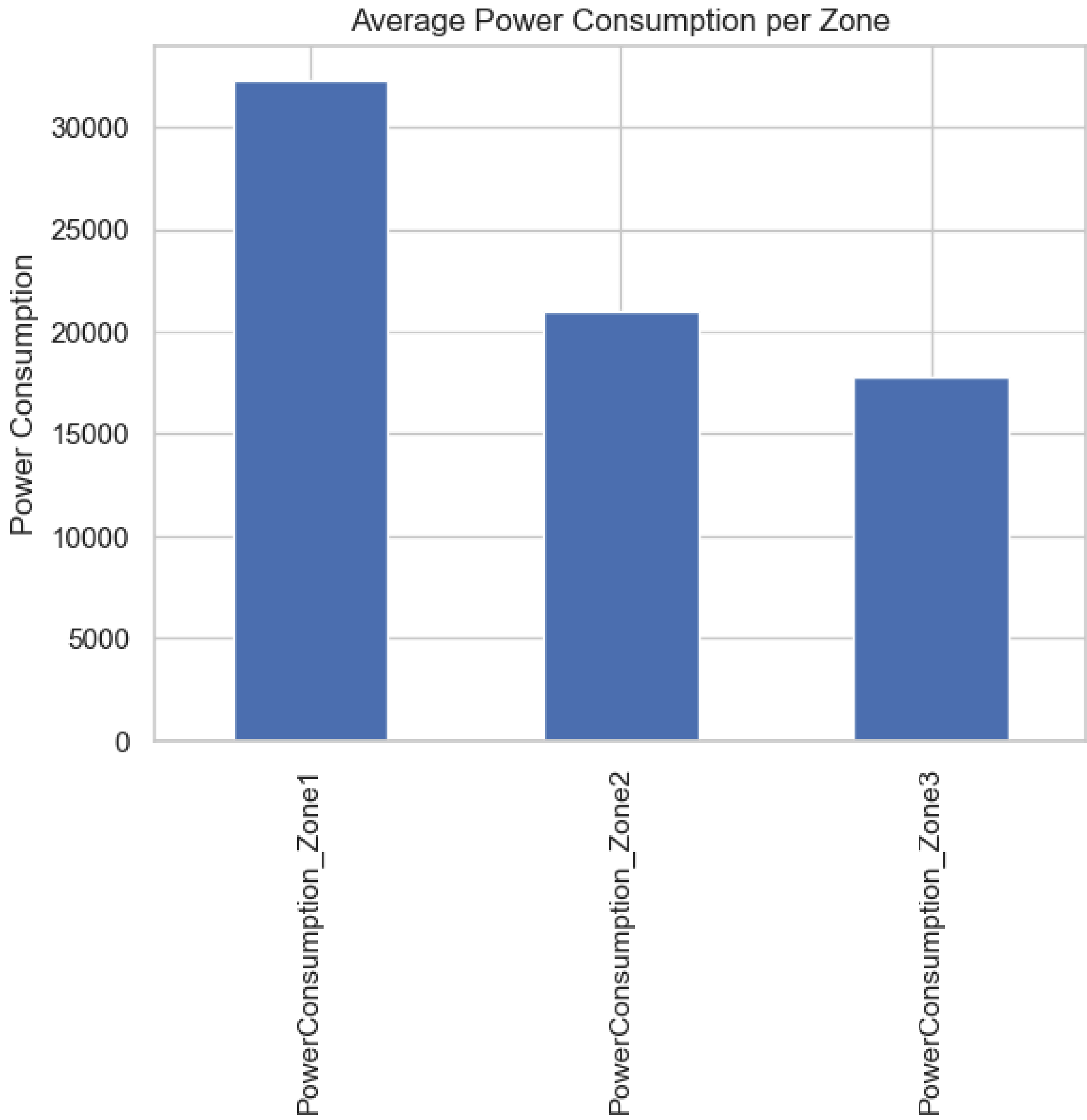
In [68]:
```
fig9, ax9 = plt.subplots(figsize=(14, 5))
df.groupby('Hour')['TotalConsumption'].mean().plot(kind='bar', ax=ax9)
ax9.set_title('Average Power Usage by Hour')
ax9.set_xlabel('Hour')
ax9.set_ylabel('Power Consumption')
insights_outputs["9. Hourly Avg Power Usage"] = fig9
```



- Power consumption is lowest during night hours (midnight to 6 AM).
- Peaks between 10 AM to 4 PM, likely indicating core working hours.
- Supports time-of-use billing and energy-saving policies during off-peak hours.

Most energy-consuming zone on average

In [71]:
```
zone_means = df[['PowerConsumption_Zone1', 'PowerConsumption_Zone2', 'PowerConsu
fig10, ax10 = plt.subplots()
zone_means.plot(kind='bar', ax=ax10)
ax10.set_title('Average Power Consumption per Zone')
ax10.set_ylabel('Power Consumption')
insights_outputs["10. Avg Power by Zone"] = fig10
```

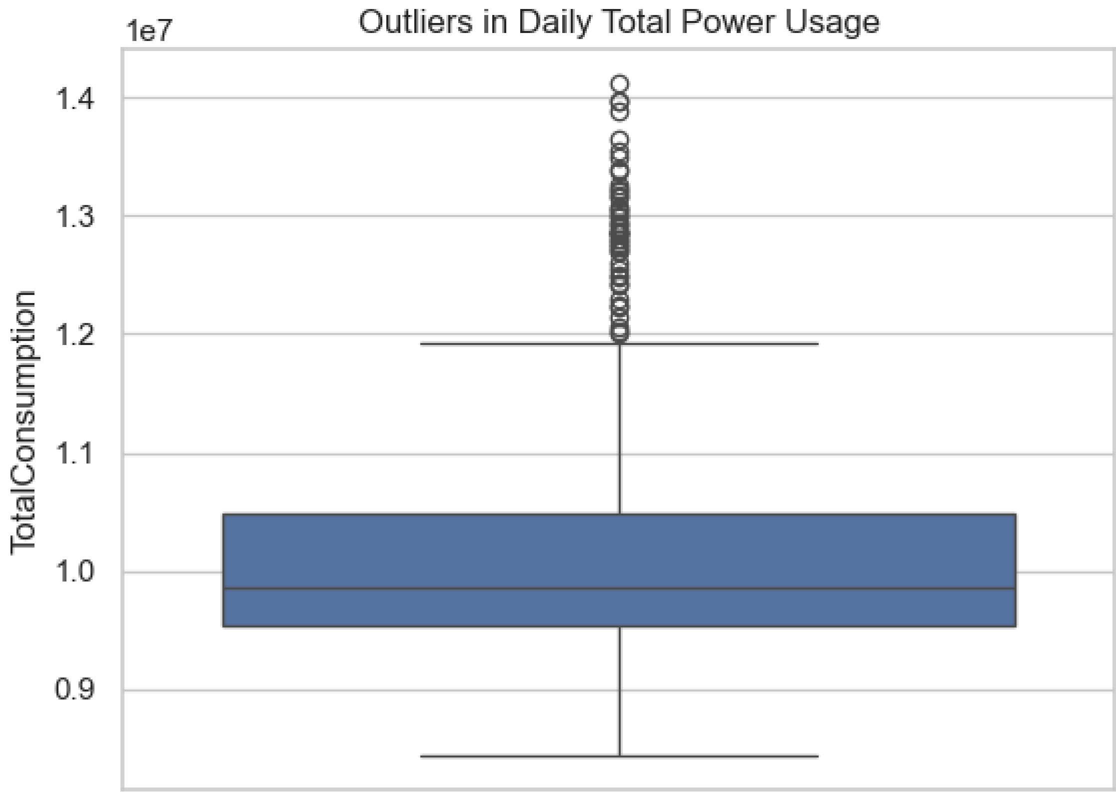


- Zone 1 is the highest consumer, followed by Zone 2, then Zone 3.
- Indicates Zone 1 might have energy-intensive operations or larger equipment load.
- Prioritizing energy audits in Zone 1 could yield significant savings.

Detect days with unusually high power usage (outliers)

In [74]:
```
daily_totals = df.groupby('Date')['TotalConsumption'].sum()
q1 = daily_totals.quantile(0.25)
q3 = daily_totals.quantile(0.75)
iqr = q3 - q1
outliers = daily_totals[(daily_totals < (q1 - 1.5 * iqr)) | (daily_totals > (q3
```

In [76]:
```
fig11, ax11 = plt.subplots()
sns.boxplot(y=daily_totals, ax=ax11)
ax11.set_title('Outliers in Daily Total Power Usage')
insights_outputs["11. Outlier Days"] = fig11
```



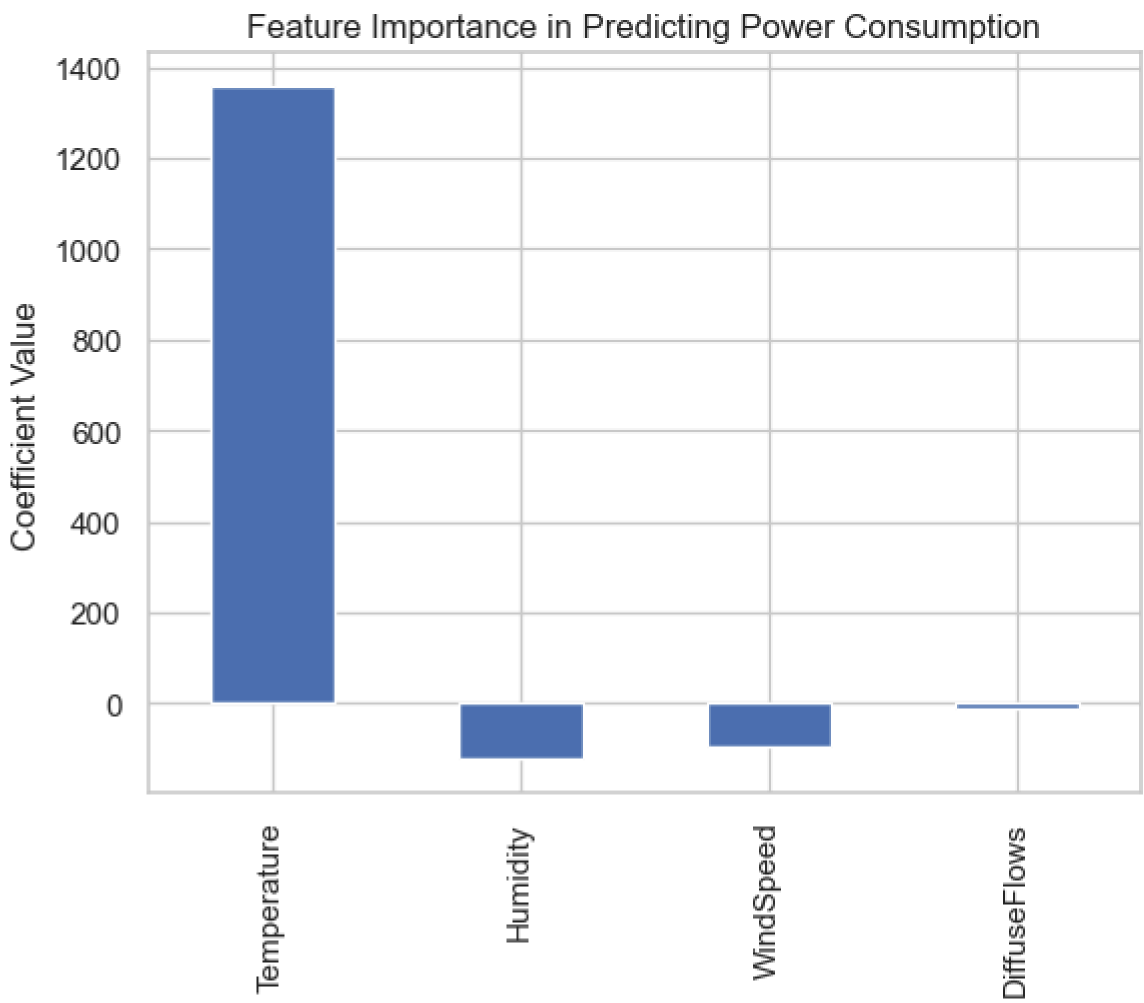
- A few days show exceptionally high or low power usage.
- Outliers may be due to equipment shutdowns, holidays, or abnormal spikes.
- Flagging these days helps in fault detection, anomaly tracking, and operational review.

Predictive and Pattern Detection

Simple regression model to predict total power consumption

In [81]:
```
features = df[['Temperature', 'Humidity', 'WindSpeed', 'DiffuseFlows']]
target = df['TotalConsumption']
model = LinearRegression().fit(features, target)
coeffs = pd.Series(model.coef_, index=features.columns)
```

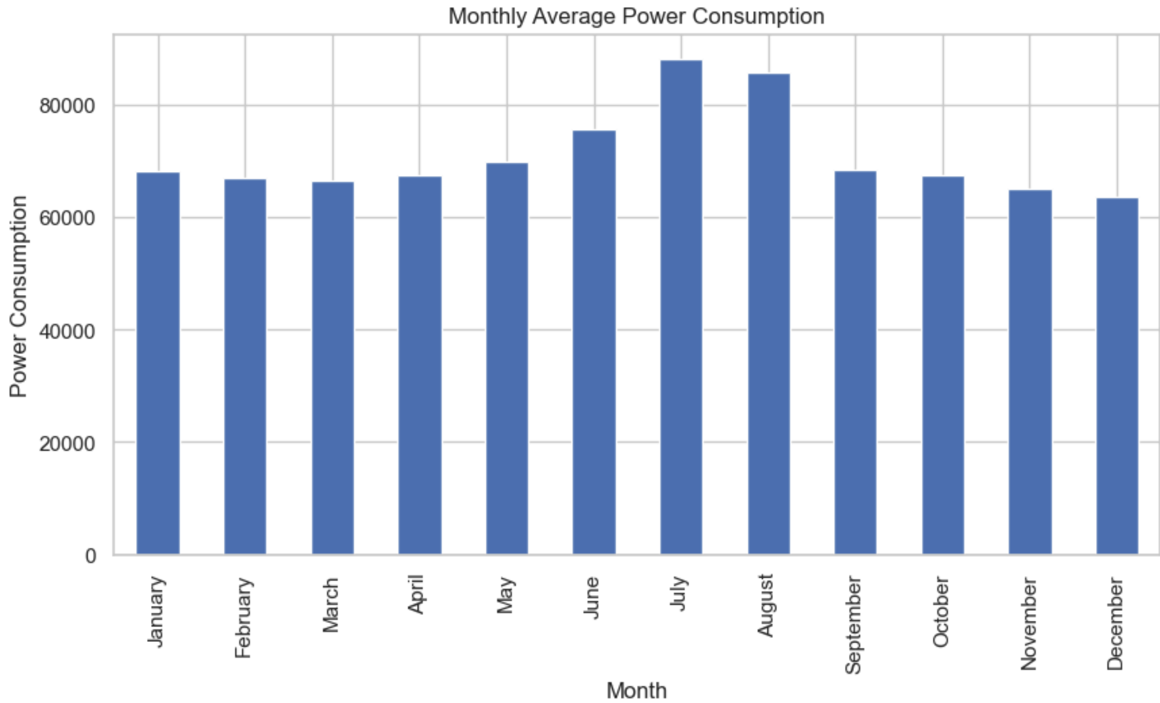
In [83]:
```
fig12, ax12 = plt.subplots()
coeffs.plot(kind='bar', ax=ax12)
ax12.set_title('Feature Importance in Predicting Power Consumption')
ax12.set_ylabel('Coefficient Value')
insights_outputs["12. Regression Feature Importance"] = fig12
```



- Temperature has a strong positive impact on total power consumption.
- Humidity and Wind speed show moderate influence, while Diffuse Flows has minimal effect.
- Insightful for predictive modeling and weather-based demand forecasting.

Seasonal variation (monthly averages)

In [88]:
```
fig13, ax13 = plt.subplots(figsize=(10, 5))
monthly_avg = df.groupby('Month')['TotalConsumption'].mean().reindex([
    'January', 'February', 'March', 'April', 'May', 'June',
    'July', 'August', 'September', 'October', 'November', 'December'
])
monthly_avg.plot(kind='bar', ax=ax13)
ax13.set_title('Monthly Average Power Consumption')
ax13.set_ylabel('Power Consumption')
insights_outputs["13. Seasonal Variation"] = fig13
```



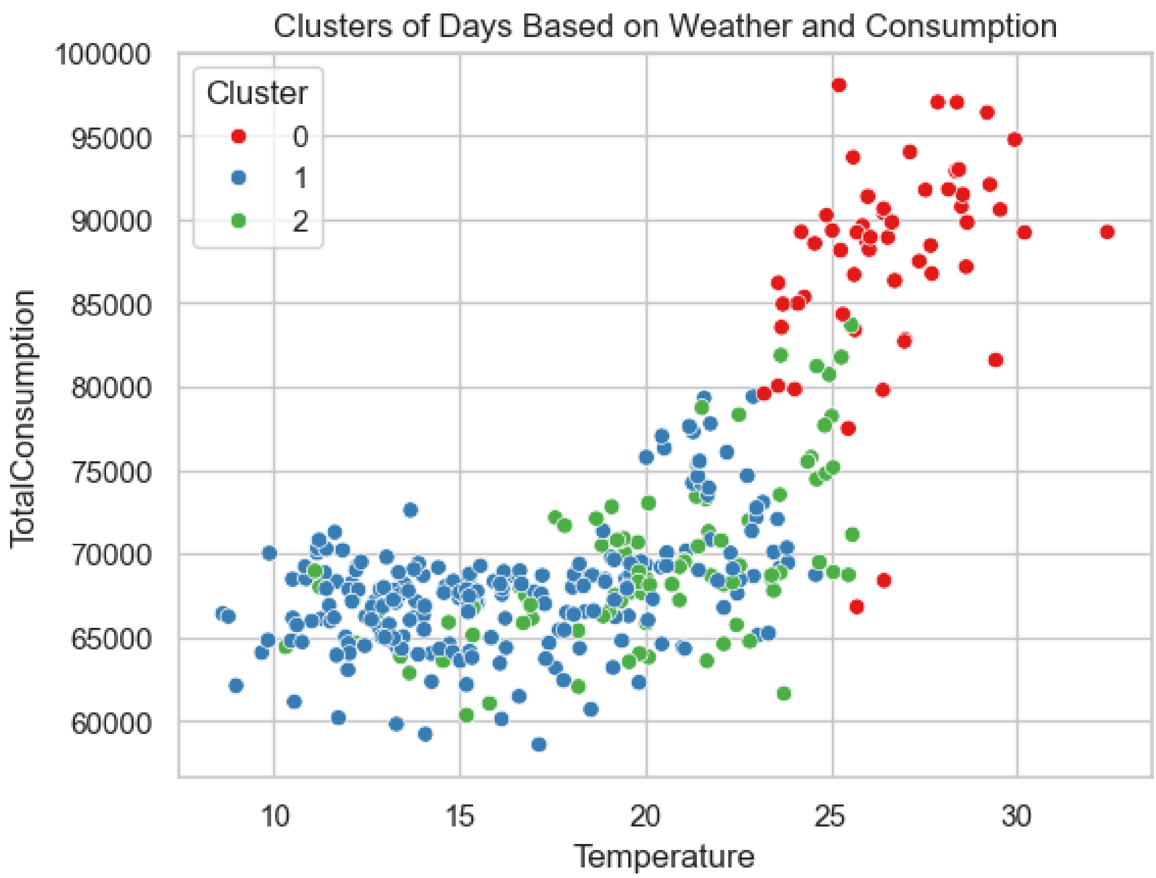
- Power usage is higher in summer months, likely due to cooling systems.
- Lower during winter months, indicating seasonal dependency.
- Useful for capacity planning and setting energy budgets per quarter.

Cluster days based on weather & power usage

In [90]:
```
daily_features = df.groupby('Date')[['Temperature', 'Humidity', 'WindSpeed', 'Tc
scaler = StandardScaler()
X_scaled = scaler.fit_transform(daily_features)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X_scaled)
daily_features['Cluster'] = kmeans.labels_
```

```
C:\Users\Nishita Bala\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:144
6: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when t
here are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

In [92]:
```
fig14, ax14 = plt.subplots()
sns.scatterplot(data=daily_features, x='Temperature', y='TotalConsumption', hue=
ax14.set_title('Clusters of Days Based on Weather and Consumption')
insights_outputs["14. Clustering Days"] = fig14
```

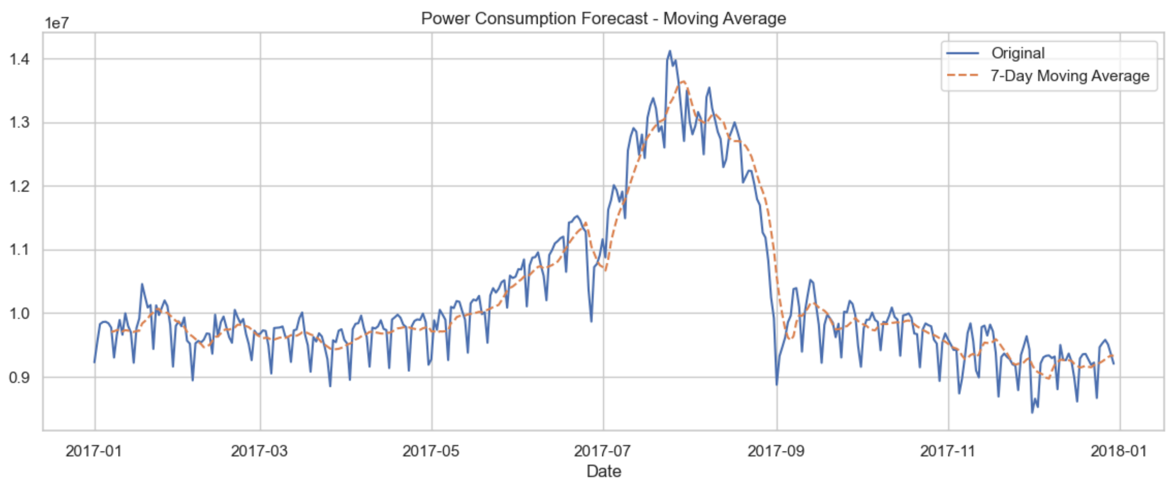


- Days grouped into 3 distinct clusters based on temperature and total power usage.
- Helps identify similar usage patterns, such as high-load summer days or low-usage weekends.
- Can assist in targeted energy management strategies.

Simple time series forecast using moving average

In [95]:

```
daily_series = daily_total.rolling(window=7).mean()
fig15, ax15 = plt.subplots(figsize=(14, 5))
daily_total.plot(ax=ax15, label='Original')
daily_series.plot(ax=ax15, label='7-Day Moving Average', linestyle='--')
ax15.set_title('Power Consumption Forecast - Moving Average')
ax15.legend()
insights_outputs["15. Moving Average Forecast"] = fig15
```



- 7-day moving average smooths out short-term fluctuations.
- Shows overall trends and seasonality, making it easier to forecast future demand.
- Useful for short-term load forecasting and inventory planning.