BIA 656 Final Project Sarah WolbergIntroduction Renewable energy is hotly debated topic in every part of the world today. I downloaded a data set from Kaggle labeled solar weather. The dataset has 17 columns having 196776 data points. Out of theses 16 columns 1 column is 'Object' type, 6 columns are of 'Float64' type and rest 10 columns are of 'Int64' type. In this project, I plan to analyze the data set using various statistical and machine learning techniques to identify patterns and trends in the data that can help others to develop predictive models that can be used to optimize renewable energy production.About the Data The dataset has 17 columns having 196776 data points. Out of theses 16 columns 1 column is 'Object' type, 6 columns are of 'Float64' type and rest 10 columns are of 'Int64' type.   It is also a tabular dataset that includes information on various weather conditions, as well as the energy production of a renewable energy system. The dataset contains columns such as "Time" (timestamp), "Energy delta[Wh]" (energy production), "GHI" (global horizontal irradiance), "temp" (temperature), "pressure", "humidity", "wind_speed", "rain_1h", "snow_1h", "clouds_all", "isSun", "sunlightTime", "dayLength", "SunlightTime/daylength", "weather_type", "hour", and "month". The data was likely collected over a period using sensors and other measuring devices and can be used to gain insights into the relationship between weather conditions and renewable energy production.The methodology I will follow is as follows: 1. Download data set. 2. Import it into python using Pandas. 3 Clean data. 4. Split the data into train and test the data set. 5. Inspect the data and look at overall statistics. 6. Look at the split features from labels. 7. Normalize the data, 8) develop the model. 8. Inspect the model and see to see if the model can predict future values. 9. Train the model. 10. Validate the model accuracy using visualization techniques. 11. Make predictions. 12. Describe results. 13. Discuss future research opportunities.

In [1]:
```
import pandas as pd
import os
from sklearn.model_selection import train_test_split
```

In [2]:
```
pd.__version__
```

Out[2]:
```
'2.0.1'
```

In [3]:
```
#os.getcwd()
os.getcwd()
```

Out[3]:
```
'/Users/sarahrwolberg/Desktop'
```

path = os.listdir(os.getcwd()) path print(path)

In [4]:
```
import pandas as pd


# # parser.py (built-in file in pandas) file has this implementation
# read_csv = _make_parser_function('read_csv', sep=',')
# read_csv = Appender(_read_csv_doc)(read_csv)
# data_frame = pd.read_csv("data.csv", sep=";")
# data_frame
#df = pd.read_csv('bank.csv')
#df = pd.read_excel(r'C:\Users\swolberg\Desktop\bank.xlsx',sep=",")
#print(df)#data = pd.read_csv('C:\\Users\\swolberg\\Desktop\\BIA 656\\bank-ful
#data=pd.read_csv(r'/Users/sarahrwolberg/Desktop/BIA656/bank-full.csv')
data=pd.read_csv('solar_weather.csv')
data
```

Out[4]:

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	c
0	2017-01-01 00:00:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	
1	2017-01-01 00:15:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	
2	2017-01-01 00:30:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	
3	2017-01-01 00:45:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	
4	2017-01-01 01:00:00	0	0.0	1.7	1020	100	5.2	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
196771	2022-08-31 16:45:00	118	23.7	18.6	1023	57	3.8	0.0	0.0	
196772	2022-08-31 17:00:00	82	15.6	18.5	1023	61	4.2	0.0	0.0	
196773	2022-08-31 17:15:00	51	8.0	18.5	1023	61	4.2	0.0	0.0	
196774	2022-08-31 17:30:00	24	2.1	18.5	1023	61	4.2	0.0	0.0	
196775	2022-08-31 17:45:00	0	0.0	18.5	1023	61	4.2	0.0	0.0	

196776 rows × 17 columns

In [5]:
```
data.head()
```

Out[5]:

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	clouds_
0	2017-01-01 00:00:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	1
1	2017-01-01 00:15:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	1
2	2017-01-01 00:30:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	1
3	2017-01-01 00:45:00	0	0.0	1.6	1021	100	4.9	0.0	0.0	1
4	2017-01-01 01:00:00	0	0.0	1.7	1020	100	5.2	0.0	0.0	1

In [6]:

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.9/site-pa
ckages (1.7.5)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.9/site-pack
ages (from xgboost) (1.22.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.9/site-pack
ages (from xgboost) (1.7.3)
Note: you may need to restart the kernel to use updated packages.
```

In [7]:

```
import warnings
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from tqdm import tqdm
from xgboost import XGBRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.metrics import mean_absolute_percentage_error
```

In [8]:

```
%matplotlib inline

random_state = 123
np.random.seed(random_state)

warnings.simplefilter('ignore')
np.set_printoptions(precision=5, suppress=True)

sns.set_theme()
```

Results and Analysis The dataset has 17 columns having 196776 data points. Out of theses 16 columns 1 column is 'Object' type, 6 columns are of 'Float64' type and rest 10 columns are of 'Int64' type.

In [45]:

```
df = pd.DataFrame(data)
df.head()
```

Out[45]:

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	clouds_all
0	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
1	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
2	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
3	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
4	2017-01-01	0	0.0	1.7	1020	100	5.2	0.0	0.0	100

The statistical information of the data is also given below for data.describe. The mean, min, quartiles, max and standard deviation are calculated.

In [51]:
```
data.describe()
```

Out[51]:

	Time	Energy delta[Wh]	GHI	temp	pressure	
count	196776	196776.000000	196776.000000	196776.000000	196776.000000	1967
mean	2019-10-29 10:51:46.037321472	573.008228	32.596538	9.790521	1015.292780	
min	2017-01-01 00:00:00	0.000000	0.000000	-16.600000	977.000000	2
25%	2018-06-02 00:00:00	0.000000	0.000000	3.600000	1010.000000	7
50%	2019-10-28 00:00:00	0.000000	1.600000	9.300000	1016.000000	8
75%	2021-03-24 00:00:00	577.000000	46.800000	15.700000	1021.000000	9
max	2022-08-31 00:00:00	5020.000000	229.200000	35.800000	1047.000000	10
std	NaN	1044.824047	52.172018	7.995428	9.585773	1

I also try to figure out what type of data the data set is composed up. I use dat.info() to figure this out. The output is given below.

In [52]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 196776 entries, 0 to 196775
Data columns (total 17 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Time                    196776 non-null  datetime64[ns]
 1   Energy delta[Wh]        196776 non-null  int64
 2   GHI                     196776 non-null  float64
 3   temp                    196776 non-null  float64
 4   pressure                196776 non-null  int64
 5   humidity                196776 non-null  int64
 6   wind_speed              196776 non-null  float64
 7   rain_1h                 196776 non-null  float64
 8   snow_1h                 196776 non-null  float64
 9   clouds_all              196776 non-null  int64
 10  isSun                   196776 non-null  int64
 11  sunlightTime            196776 non-null  int64
 12  dayLength               196776 non-null  int64
 13  SunlightTime/daylength  196776 non-null  float64
 14  weather_type            196776 non-null  int64
 15  hour                    196776 non-null  int64
 16  month                   196776 non-null  object
dtypes: datetime64[ns](1), float64(6), int64(9), object(1)
memory usage: 25.5+ MB
```

I also need to check the data for missing values.

In [53]:
```
data.isnull().sum()
```

Out[53]:
```
Time                      0
Energy delta[Wh]          0
GHI                       0
temp                      0
pressure                  0
humidity                  0
wind_speed                0
rain_1h                   0
snow_1h                   0
clouds_all                0
isSun                     0
sunlightTime              0
dayLength                 0
SunlightTime/daylength    0
weather_type              0
hour                      0
month                     0
dtype: int64
```

The data does not have any missing values which means that we do not have to clean the data and so on.Next step is to conduct a data manipulation of the data that I currently have. First we will look at Time.

In [70]:
```
#pip install seaborn
```

In [55]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')
sns.set_palette('coolwarm')
```

In [56]:
```
data['Time'] = pd.to_datetime(data['Time'])
data.head()
```

Out[56]:

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	clouds_all
0	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
1	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
2	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
3	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
4	2017-01-01	0	0.0	1.7	1020	100	5.2	0.0	0.0	100

I then relook at the data information to see if there are any significant changes. I use data.info() again. This can be seen.

In [57]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 196776 entries, 0 to 196775
Data columns (total 17 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Time                    196776 non-null  datetime64[ns]
 1   Energy delta[Wh]        196776 non-null  int64
 2   GHI                     196776 non-null  float64
 3   temp                    196776 non-null  float64
 4   pressure                196776 non-null  int64
 5   humidity                196776 non-null  int64
 6   wind_speed              196776 non-null  float64
 7   rain_1h                 196776 non-null  float64
 8   snow_1h                 196776 non-null  float64
 9   clouds_all              196776 non-null  int64
 10  isSun                   196776 non-null  int64
 11  sunlightTime            196776 non-null  int64
 12  dayLength               196776 non-null  int64
 13  SunlightTime/daylength  196776 non-null  float64
 14  weather_type            196776 non-null  int64
 15  hour                    196776 non-null  int64
 16  month                   196776 non-null  object
dtypes: datetime64[ns](1), float64(6), int64(9), object(1)
memory usage: 25.5+ MB
```

In [58]:
```
data['Time'] = data['Time'].apply(lambda x: x.date())
data.head()
```

Out[58]:

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	clouds_all
0	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
1	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
2	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
3	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
4	2017-01-01	0	0.0	1.7	1020	100	5.2	0.0	0.0	100

I then want to manipulate the month sot hat insteaf of 1,2,3,.. the data is showing January, February, March, ... This will help later on when graphs are used to describe the data.

In [59]:

```
data['month']=data['month'].replace({1:'January',2:'February',3:'March',4:'Apr:
data.head()
```

Out[59]:

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	clouds_all
0	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
1	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
2	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
3	2017-01-01	0	0.0	1.6	1021	100	4.9	0.0	0.0	100
4	2017-01-01	0	0.0	1.7	1020	100	5.2	0.0	0.0	100

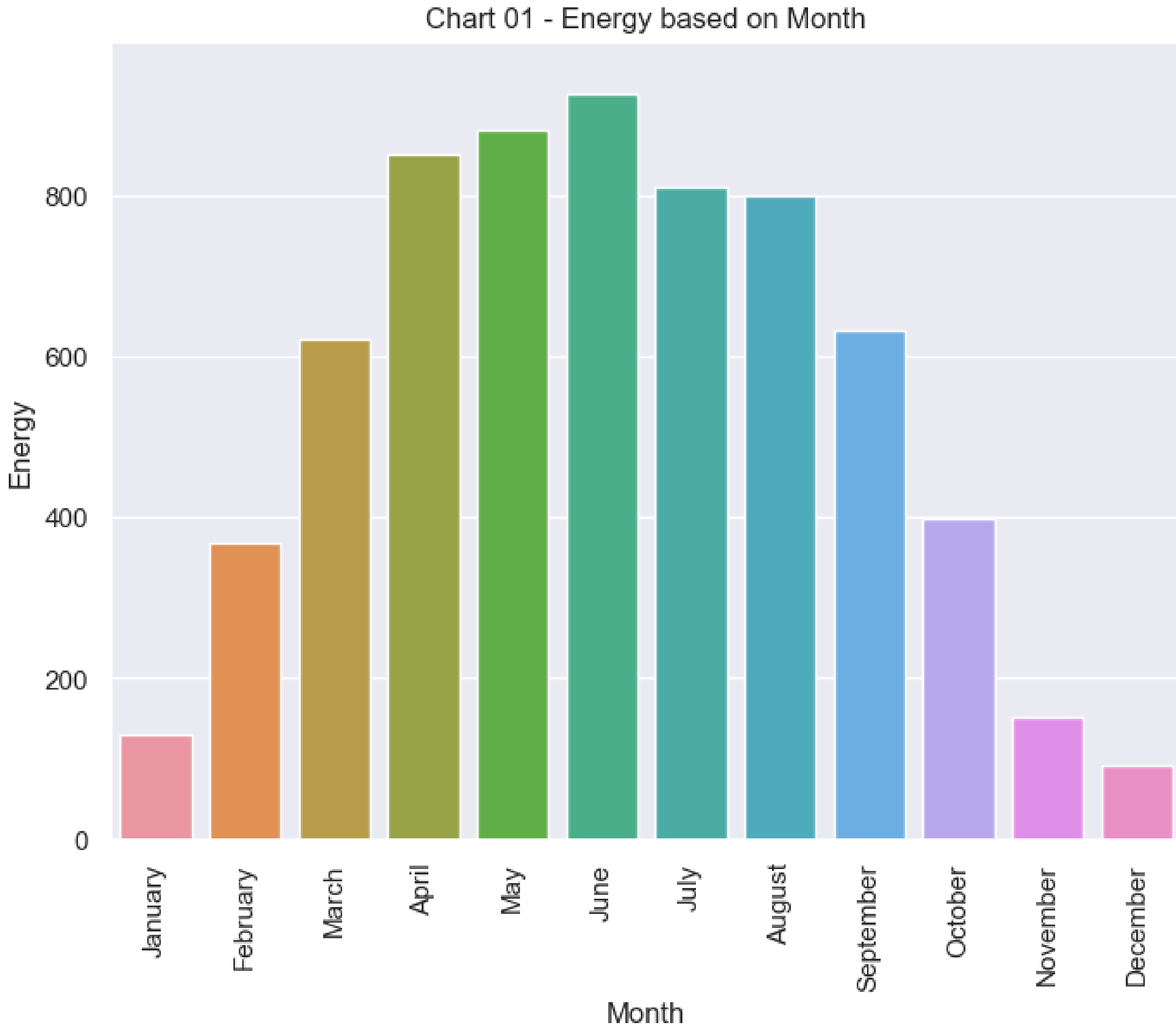
Data Interpretation

In [60]:

```
plt.figure(figsize=(8,6))
sns.barplot(x=data['month'], y=data['Energy delta[Wh]'], errwidth=0)
plt.title('Chart 01 - Energy based on Month')
plt.xlabel('Month')
plt.xticks(rotation=90)
plt.ylabel('Energy')
```

Out[60]:

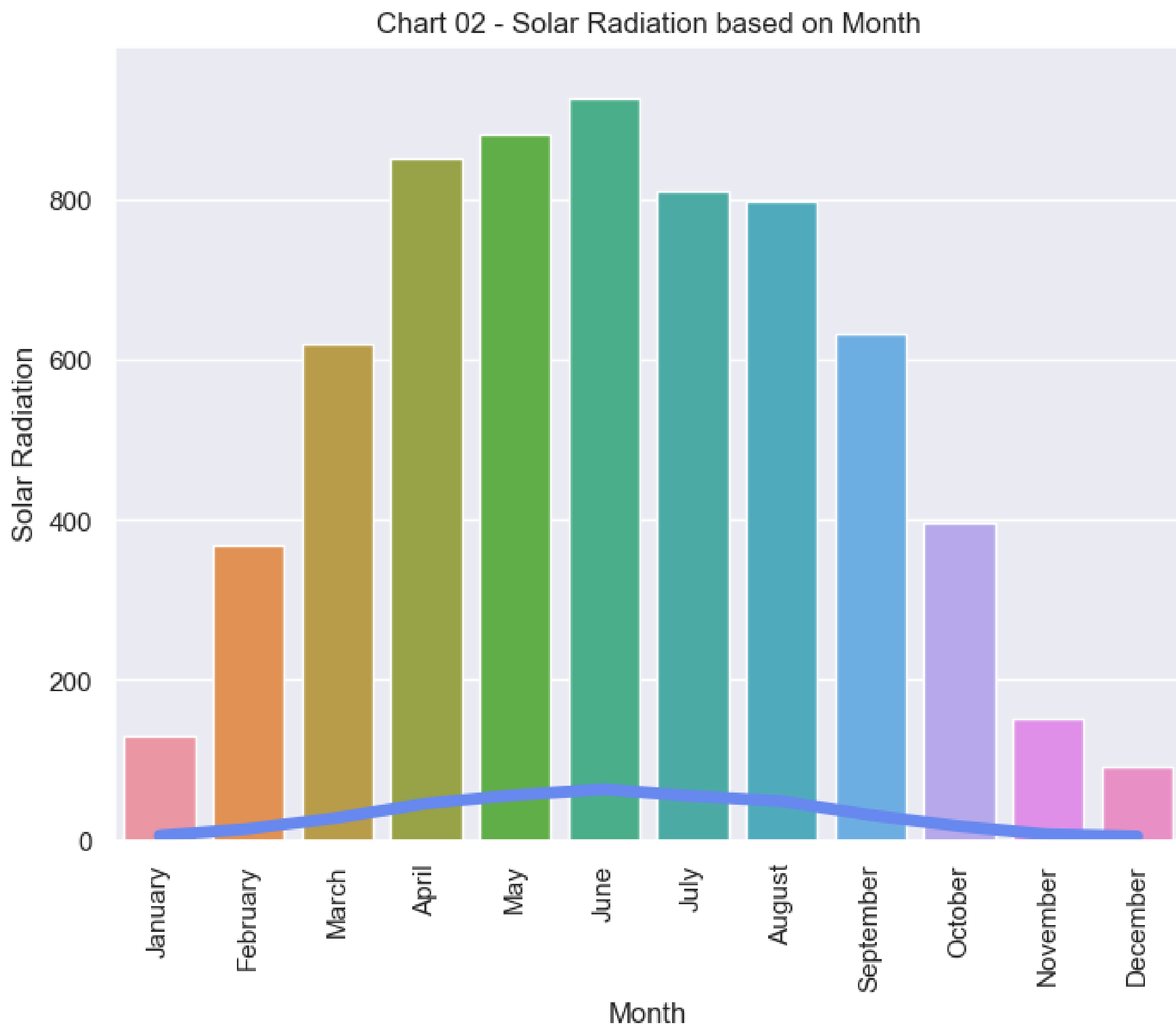
```
Text(0, 0.5, 'Energy')
```



The lowest energy producing months are January, November and December. The Highest energy producing month is June and besides that April, May, July and August are nicely energy producing months.

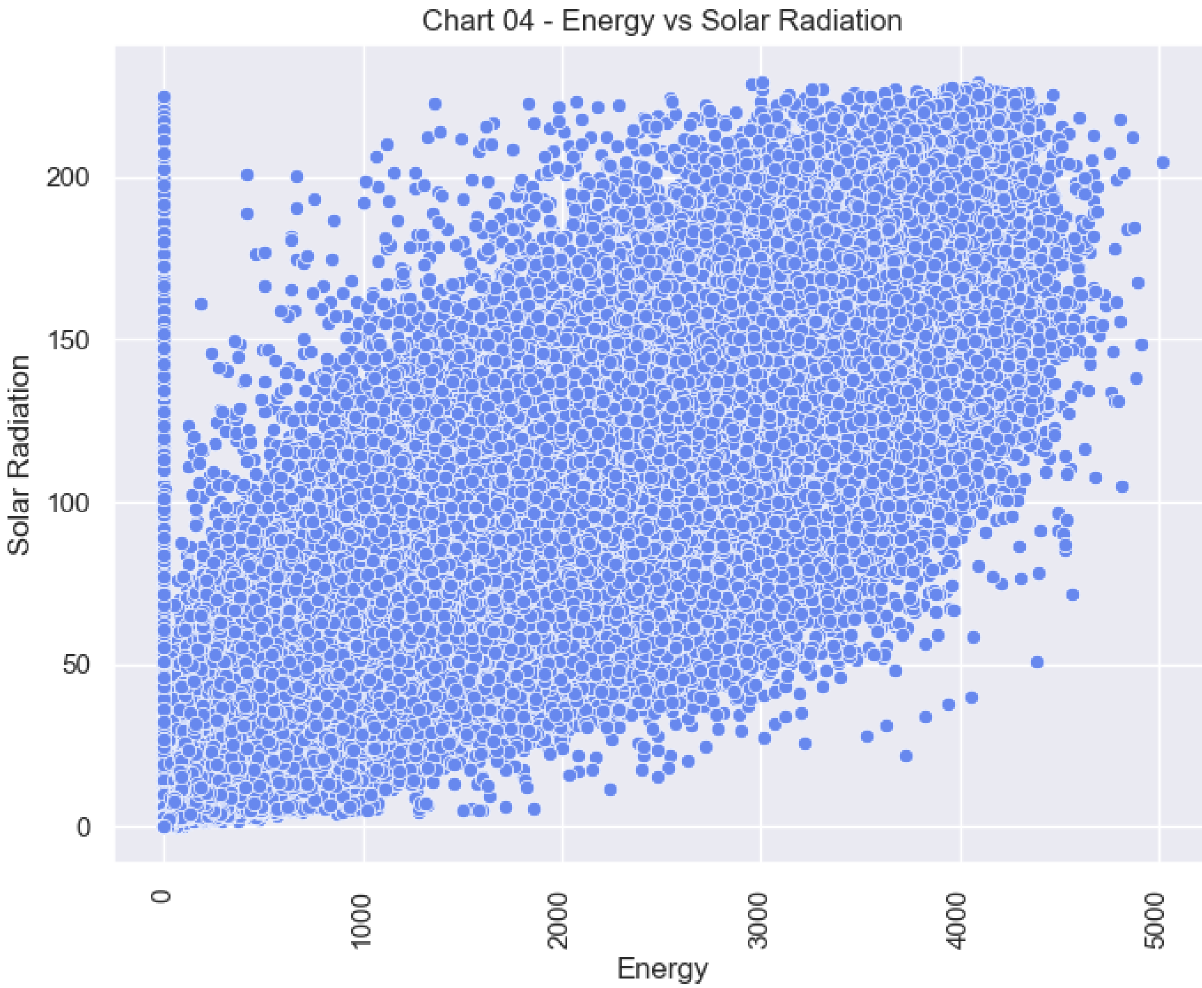
In [61]:
```
plt.figure(figsize=(8,6))
sns.lineplot(x=data['month'],y=data['GHI'],linewidth=5,errorbar=None)
sns.barplot(x=data['month'], y=data['GHI'],errwidth=0)
sns.barplot(x=data['month'], y=data['Energy delta[Wh]'], errwidth=0)
plt.title('Chart 02 - Solar Radiation based on Month')
plt.xlabel('Month')
plt.xticks(rotation=90)
plt.ylabel('Solar Radiation')
```

Out[61]:
```
Text(0, 0.5, 'Solar Radiation')
```



In [62]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x=data['Energy delta[Wh]'], y=data['GHI'])
plt.title('Chart 04 - Energy vs Solar Radiation')
plt.xlabel('Energy')
plt.xticks(rotation=90)
plt.ylabel('Solar Radiation')
```

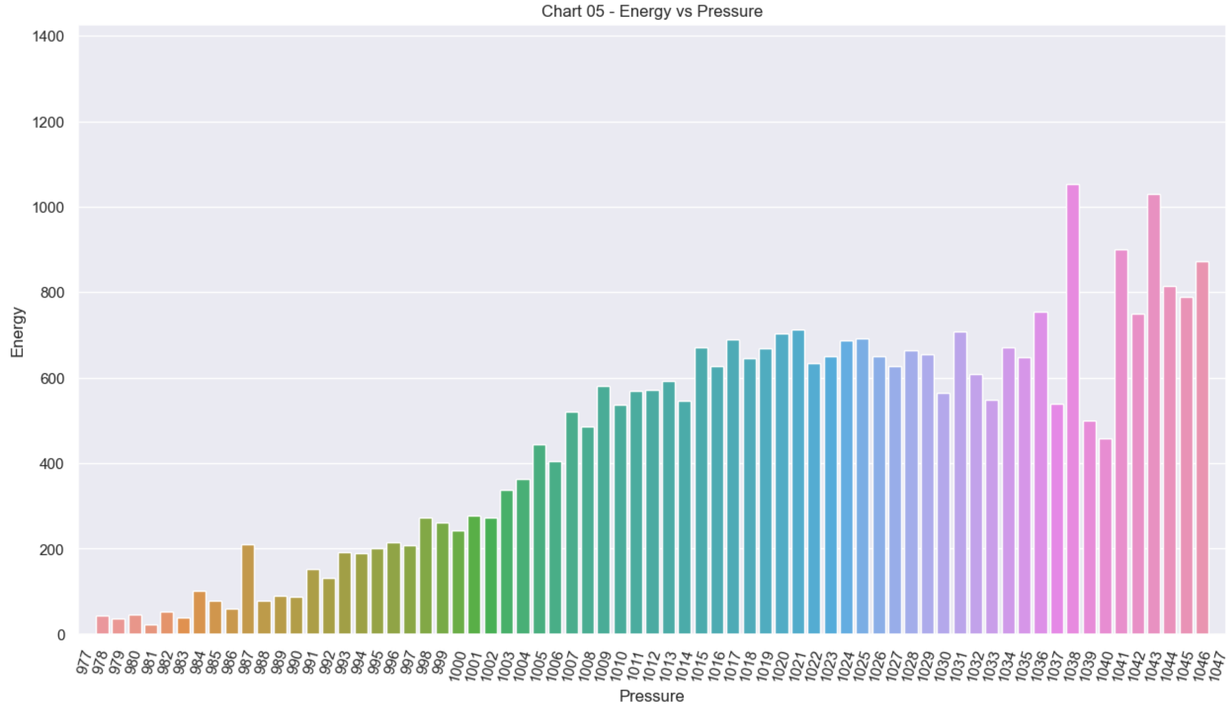
Out[62]:
```
Text(0, 0.5, 'Solar Radiation')
```



We can clearly see the relationship between Energy and Solar Radiation. The level of radiation goes up as the level of energy goes up and vice-versa.

In [63]:
```
plt.figure(figsize=(15,8))
plt.title('Chart 05 - Energy vs Pressure')
sns.barplot(x=data['pressure'], y=data['Energy delta[Wh]'], errwidth=0)
plt.xlabel('Pressure')
plt.xticks(rotation=75)
plt.ylabel('Energy')
```

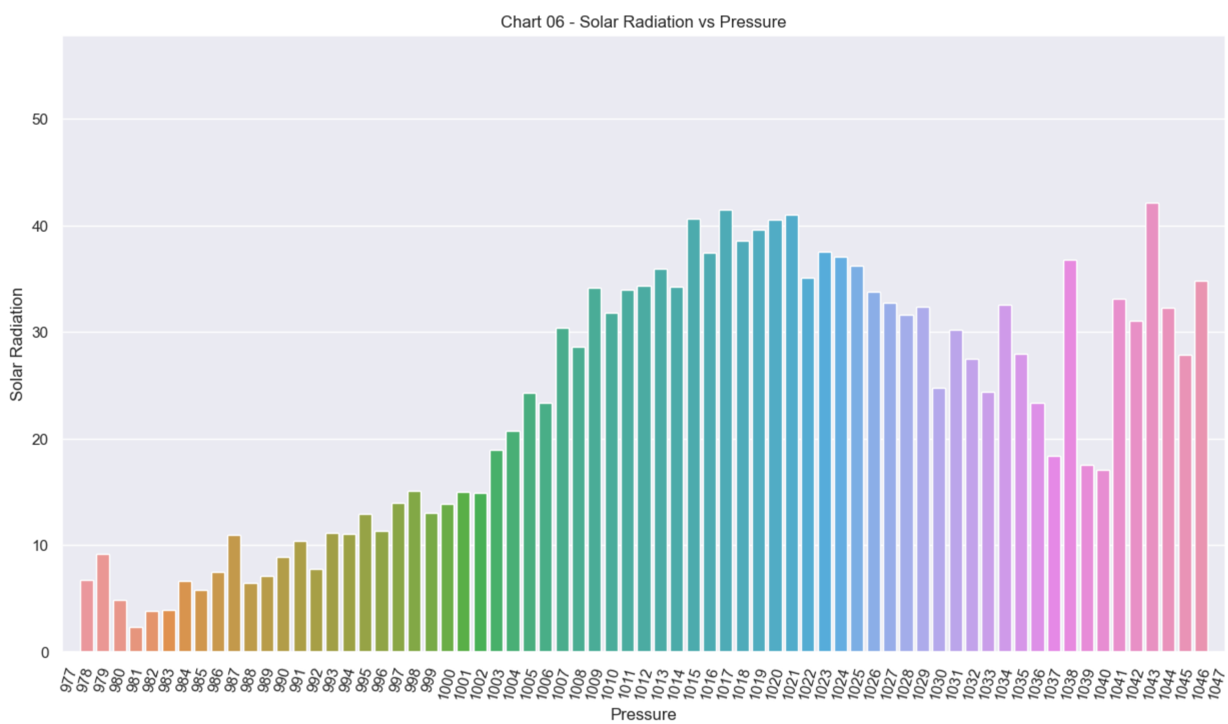
Out[63]:
```
Text(0, 0.5, 'Energy')
```



We can see there is no energy for pressure level 977 and 1047 besides that we can not see any type of relation between energy and pressure. But we can see that the energy level is not very much when the pressure is low. The highest energy we received at the pressure level 1038

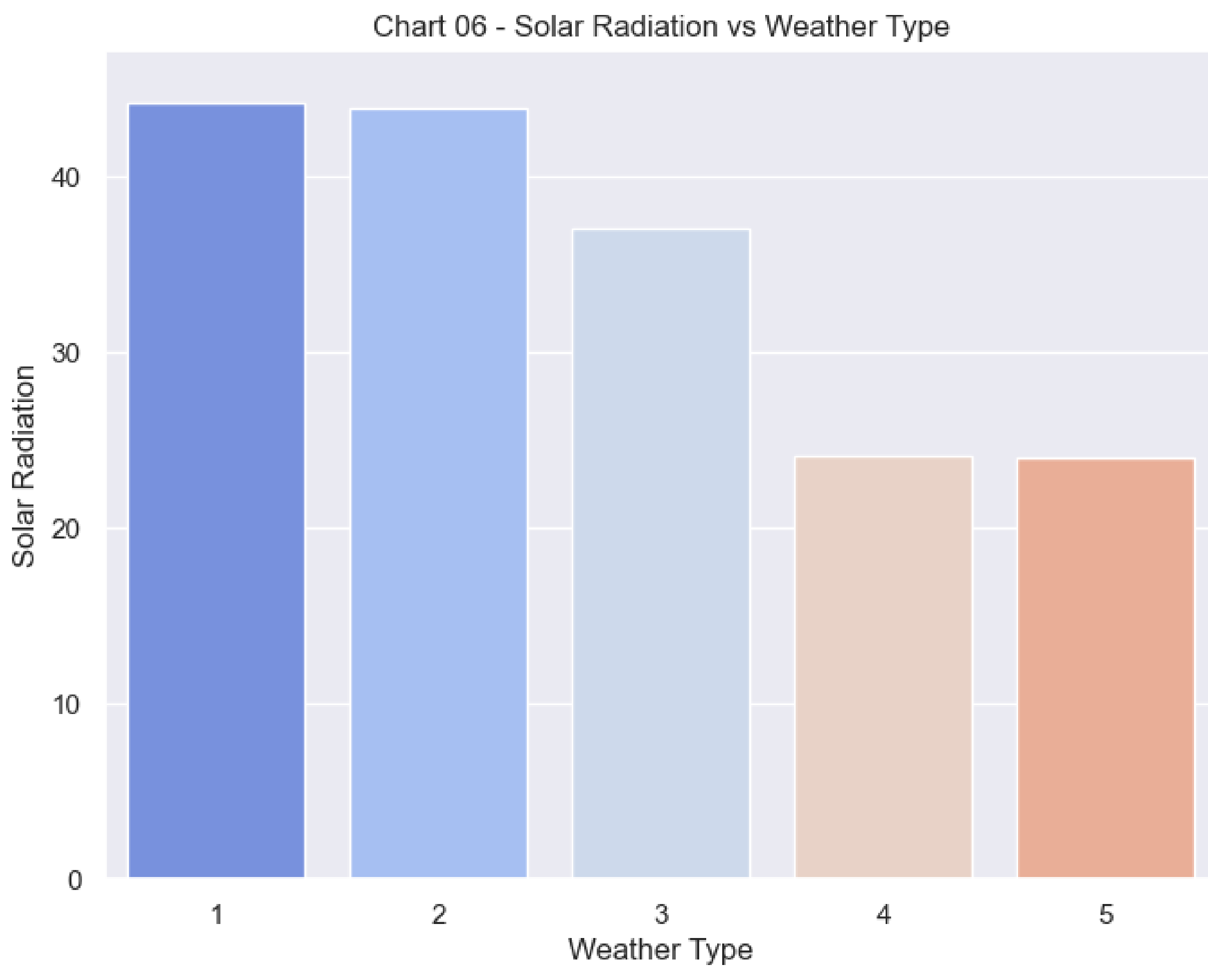
In [64]:
```
plt.figure(figsize=(15,8))
sns.barplot(x=data['pressure'], y=data['GHI'], errwidth=0)
plt.title('Chart 06 - Solar Radiation vs Pressure')
plt.xlabel('Pressure')
plt.xticks(rotation=75)
plt.ylabel('Solar Radiation')
```

Out[64]:
```
Text(0, 0.5, 'Solar Radiation')
```



In [65]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['GHI'], errwidth=0)
plt.title('Chart 06 - Solar Radiation vs Weather Type')
plt.xlabel('Weather Type')
plt.ylabel('Solar Radiation')
```

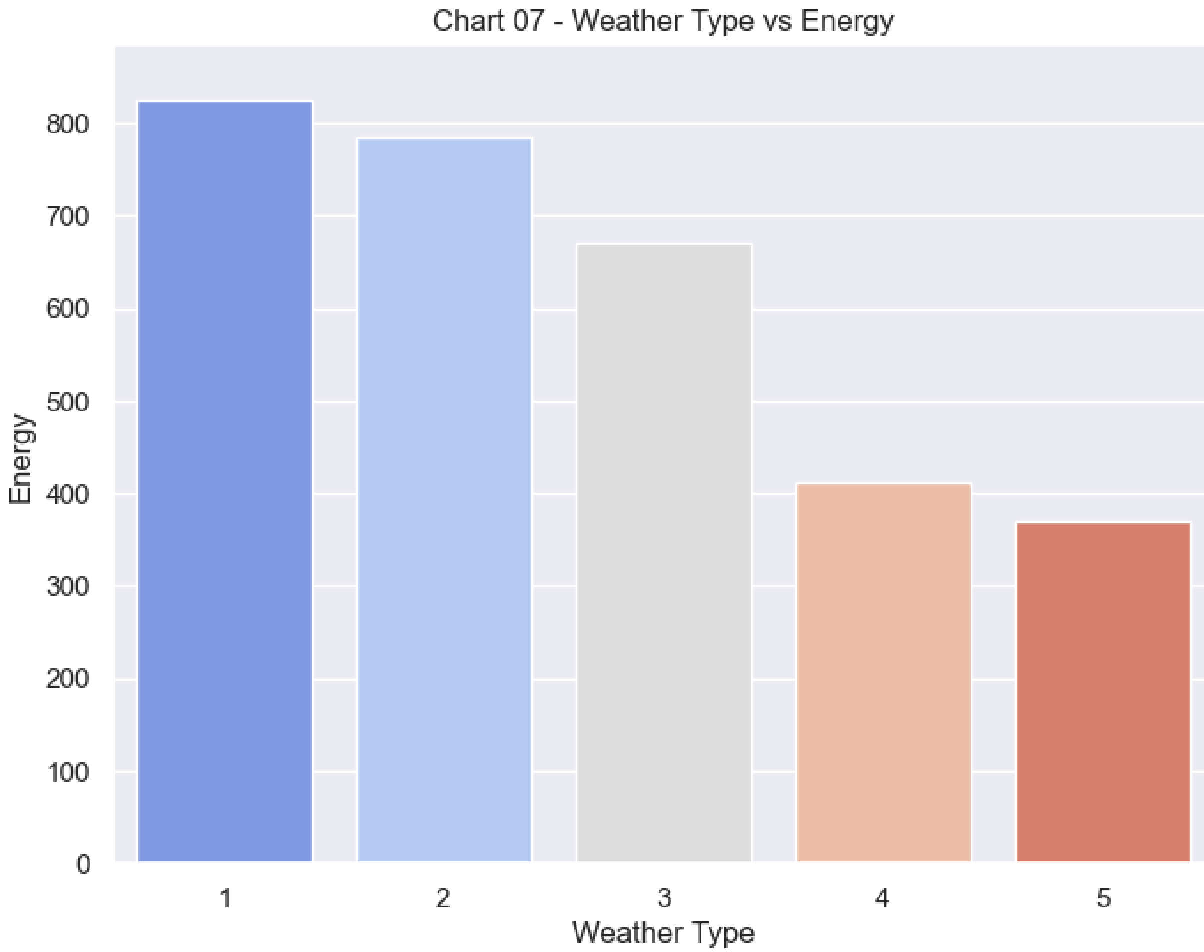
Out[65]:
```
Text(0, 0.5, 'Solar Radiation')
```



The solar radiation level is very high in weather type 1 and 2 whereas it is at its lowest in weather type 4 and 5.

In [66]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['Energy delta[Wh]'], errwidth=0, pal
plt.title('Chart 07 - Weather Type vs Energy')
plt.xlabel('Weather Type')
plt.ylabel('Energy')
```

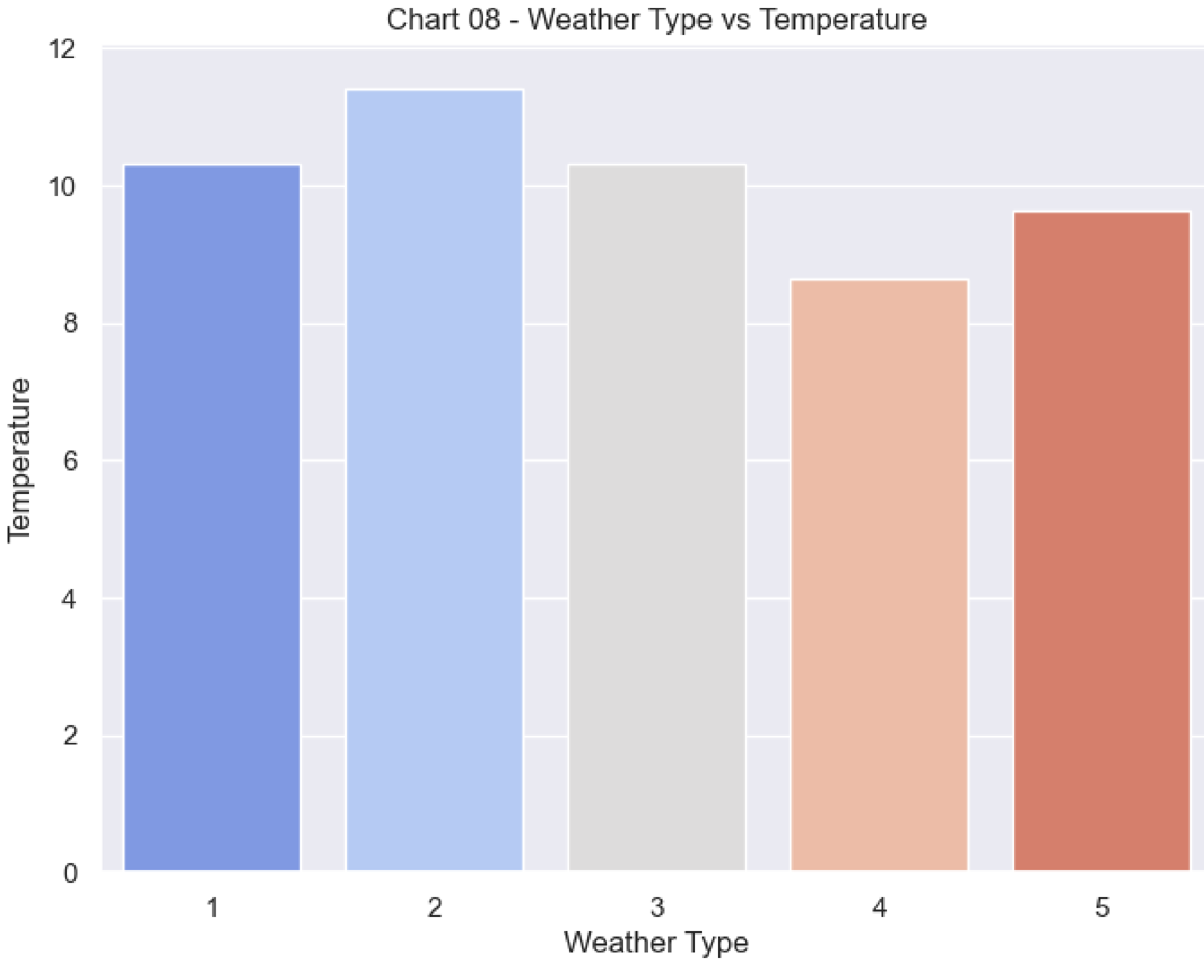
Out[66]:
```
Text(0, 0.5, 'Energy')
```



The highest energy produced in weather type 1 and weather type 5 is producing the lowest energy.

In [67]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['temp'], errwidth=0, palette='coolwa
plt.title('Chart 08 - Weather Type vs Temperature')
plt.xlabel('Weather Type')
plt.ylabel('Temperature')
```

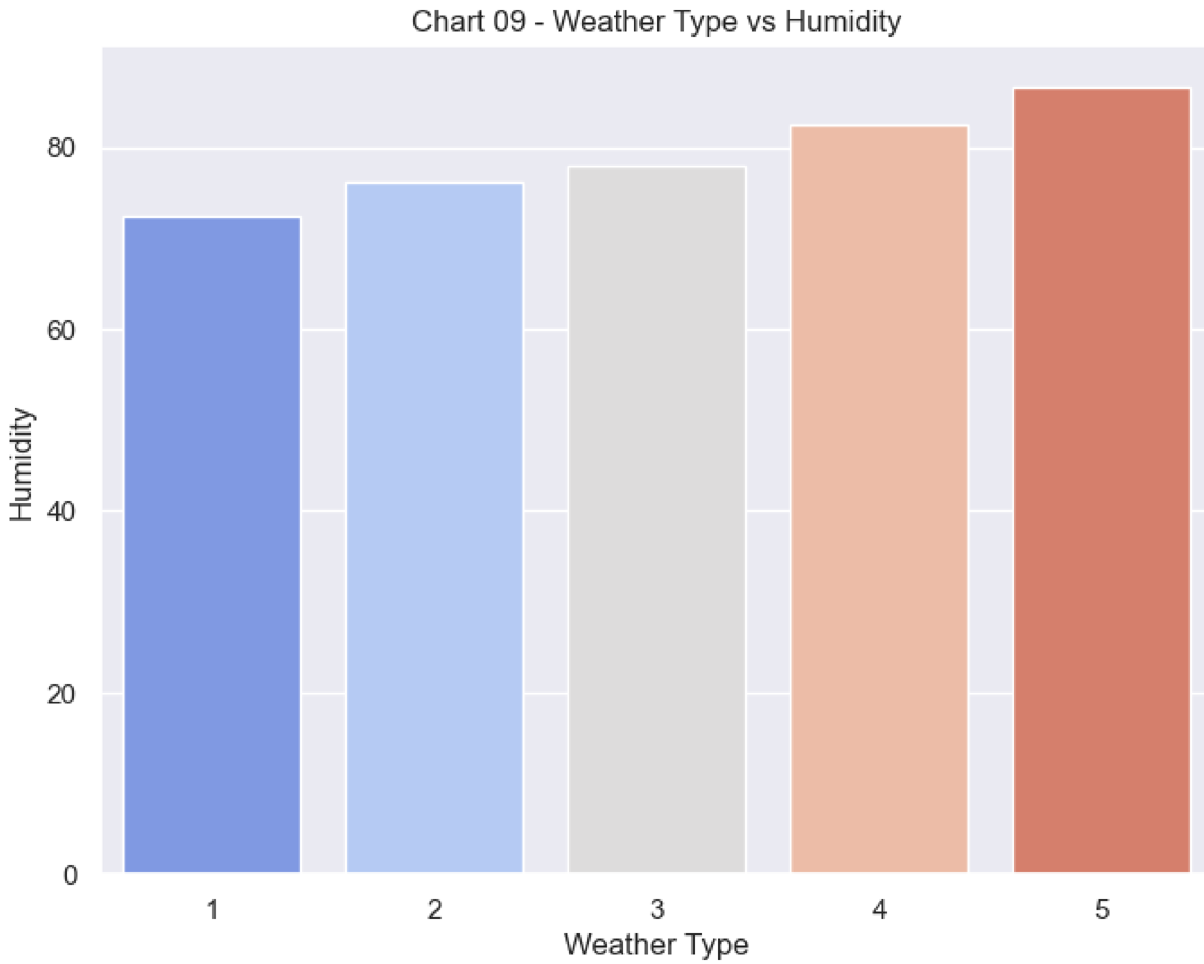
Out[67]:
```
Text(0, 0.5, 'Temperature')
```



The temerature level in all weather type is between 8-10 so we can say that the temperature doesn't vary much between different weathers.

In [68]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['humidity'], errwidth=0, palette='co
plt.title('Chart 09 - Weather Type vs Humidity')
plt.xlabel('Weather Type')
plt.ylabel('Humidity')
```

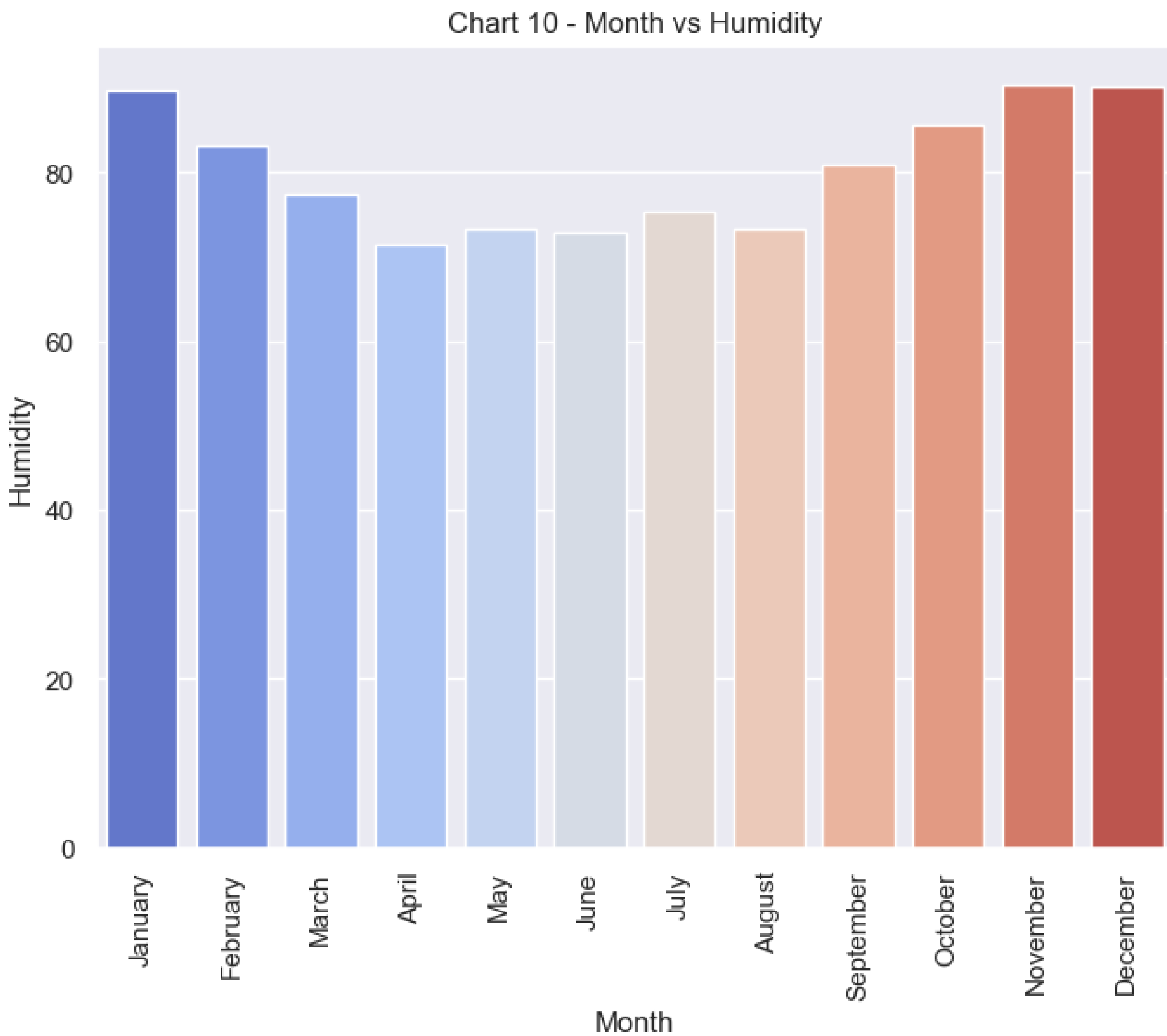
Out[68]:
```
Text(0, 0.5, 'Humidity')
```



We can see from the above chart that as the weather type goes from 1 to 5 the level of humidity also goes up.

In [27]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['month'], y=data['humidity'], errwidth=0, palette='coolwarm
plt.title('Chart 10 - Month vs Humidity')
plt.xlabel('Month')
plt.xticks(rotation=90)
plt.ylabel('Humidity')
```

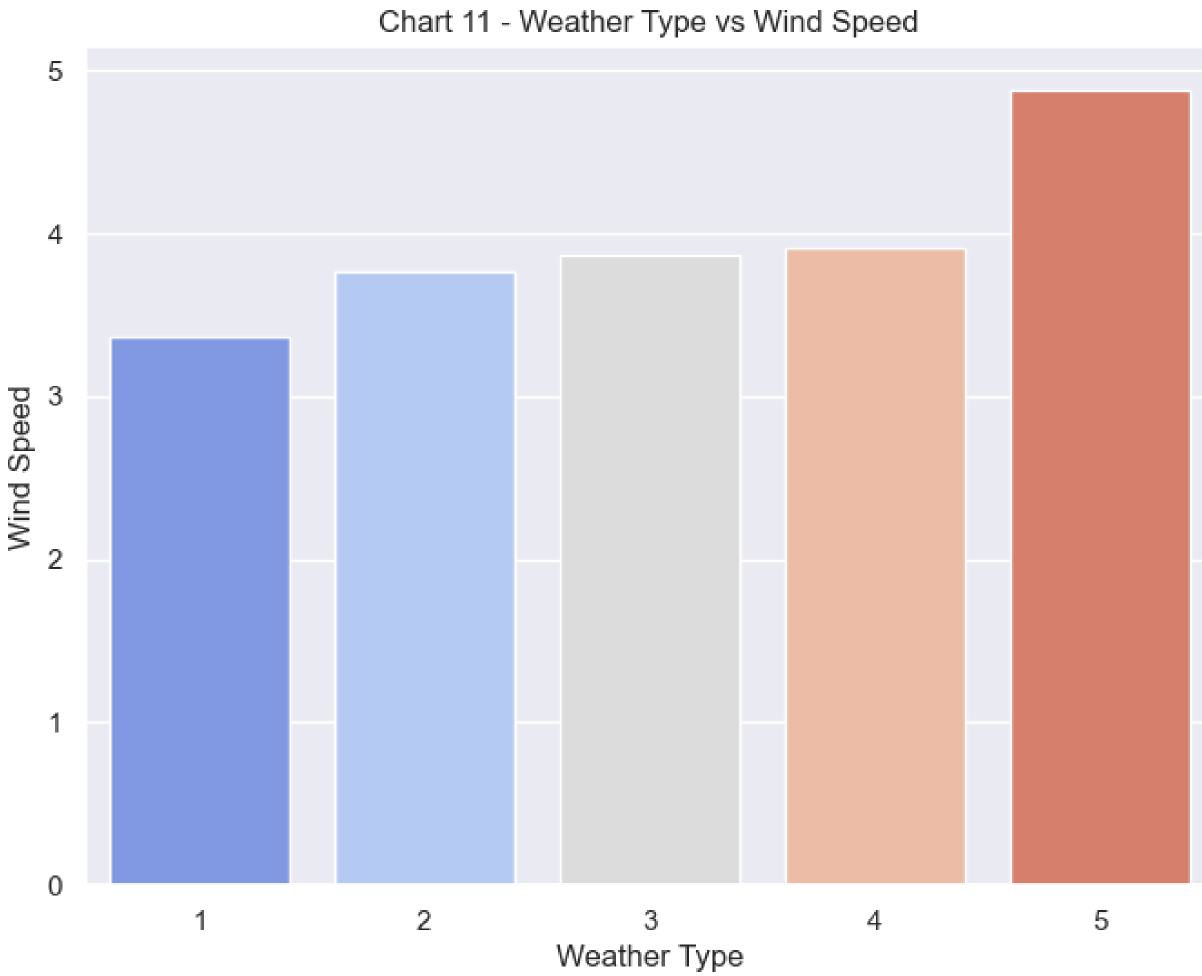
Out[27]:
```
Text(0, 0.5, 'Humidity')
```



The first and last quarter have highest humidity level than other month

In [28]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['wind_speed'], errwidth=0, palette=
plt.title('Chart 11 - Weather Type vs Wind Speed')
plt.xlabel('Weather Type')
plt.ylabel('Wind Speed')
```

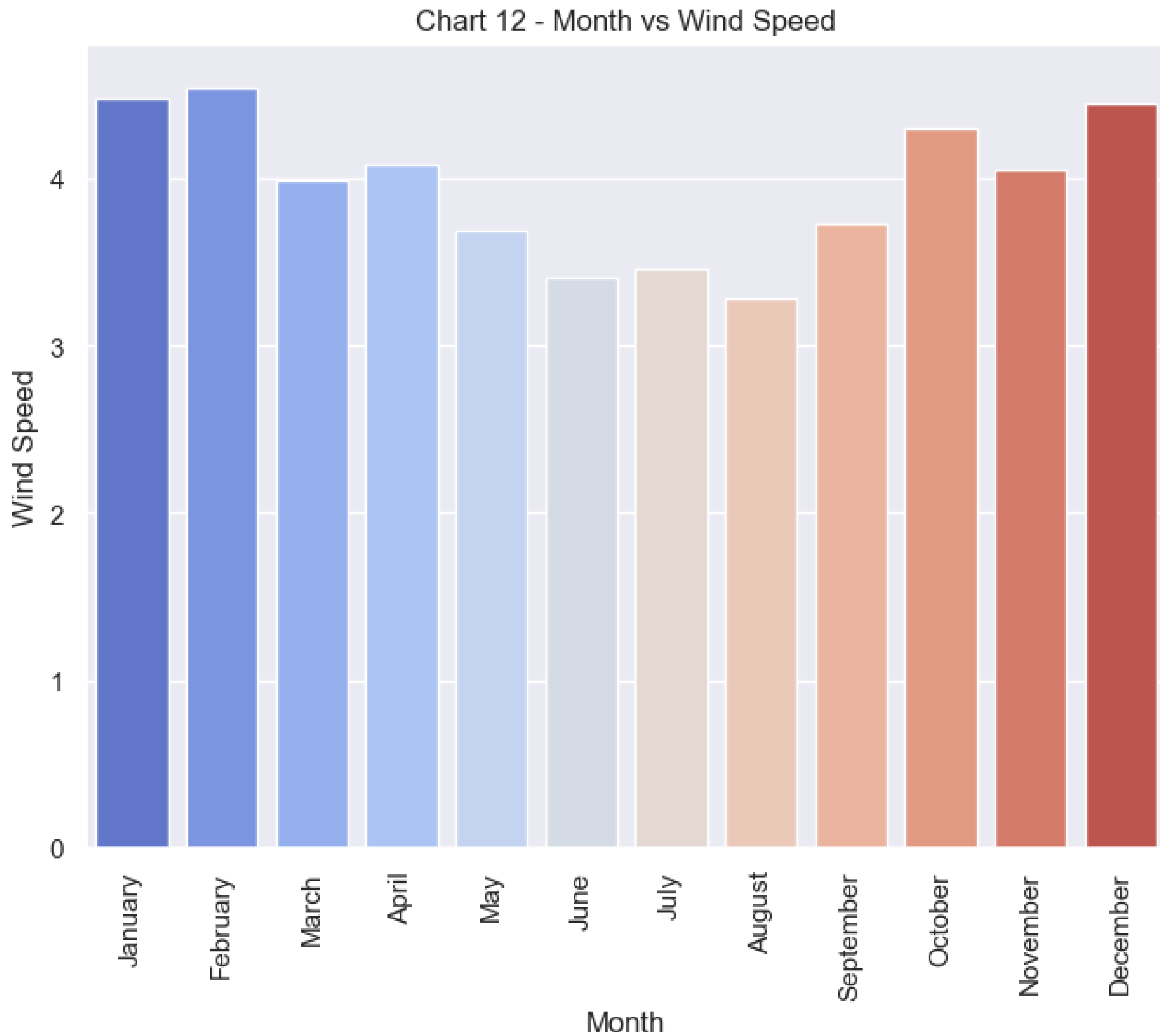
Out[28]:
```
Text(0, 0.5, 'Wind Speed')
```



The highest wind level is in weather type 5 and the lowest is in 1.

In [29]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['month'], y=data['wind_speed'], errwidth=0, palette='coolwar
plt.title('Chart 12 - Month vs Wind Speed')
plt.xlabel('Month')
plt.xticks(rotation=90)
plt.ylabel('Wind Speed')
```

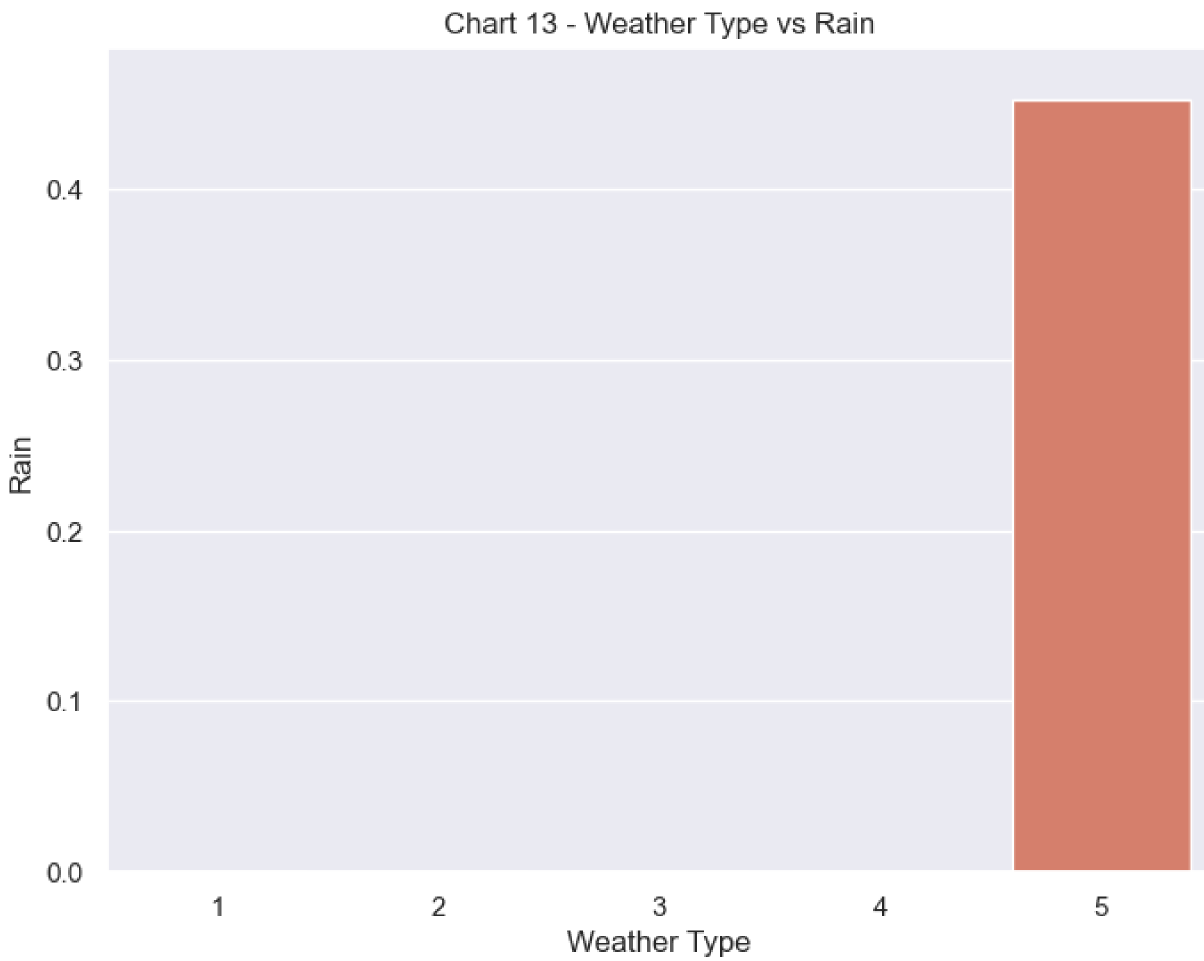
Out[29]:
```
Text(0, 0.5, 'Wind Speed')
```



January, February and the last quarter have much highesr wind speed than other months

In [30]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['rain_1h'], errwidth=0, palette='coo
plt.title('Chart 13 - Weather Type vs Rain')
plt.xlabel('Weather Type')
plt.ylabel('Rain')
```

Out[30]:
```
Text(0, 0.5, 'Rain')
```

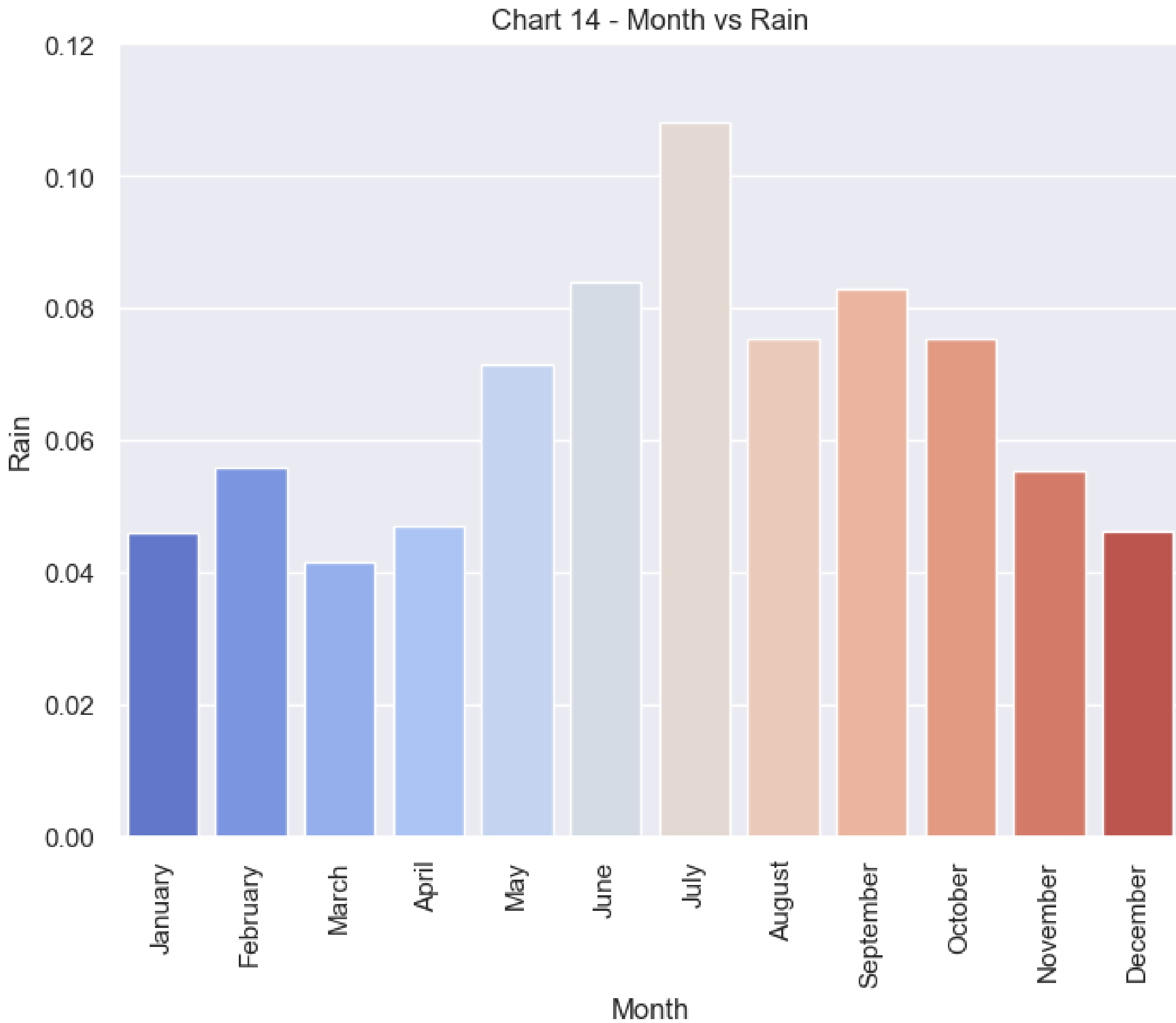


In [ ]:

In [ ]:

In [31]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['month'], y=data['rain_1h'], errwidth=0, palette='coolwarm'
plt.title('Chart 14 - Month vs Rain')
plt.xlabel('Month')
plt.xticks(rotation=90)
plt.ylabel('Rain')
```

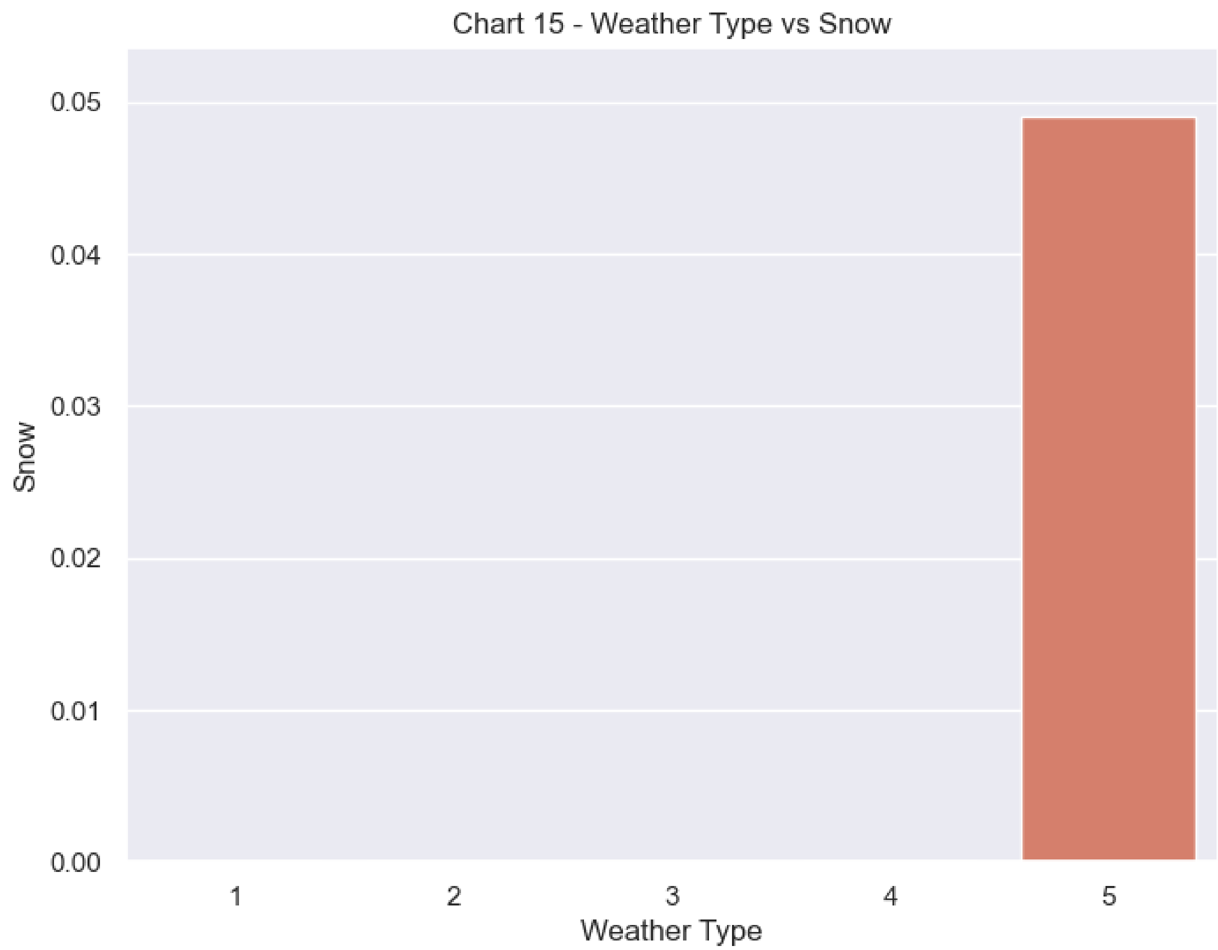
Out[31]:
```
Text(0, 0.5, 'Rain')
```



t is quite normal that rainy season have high number of rain than other months.

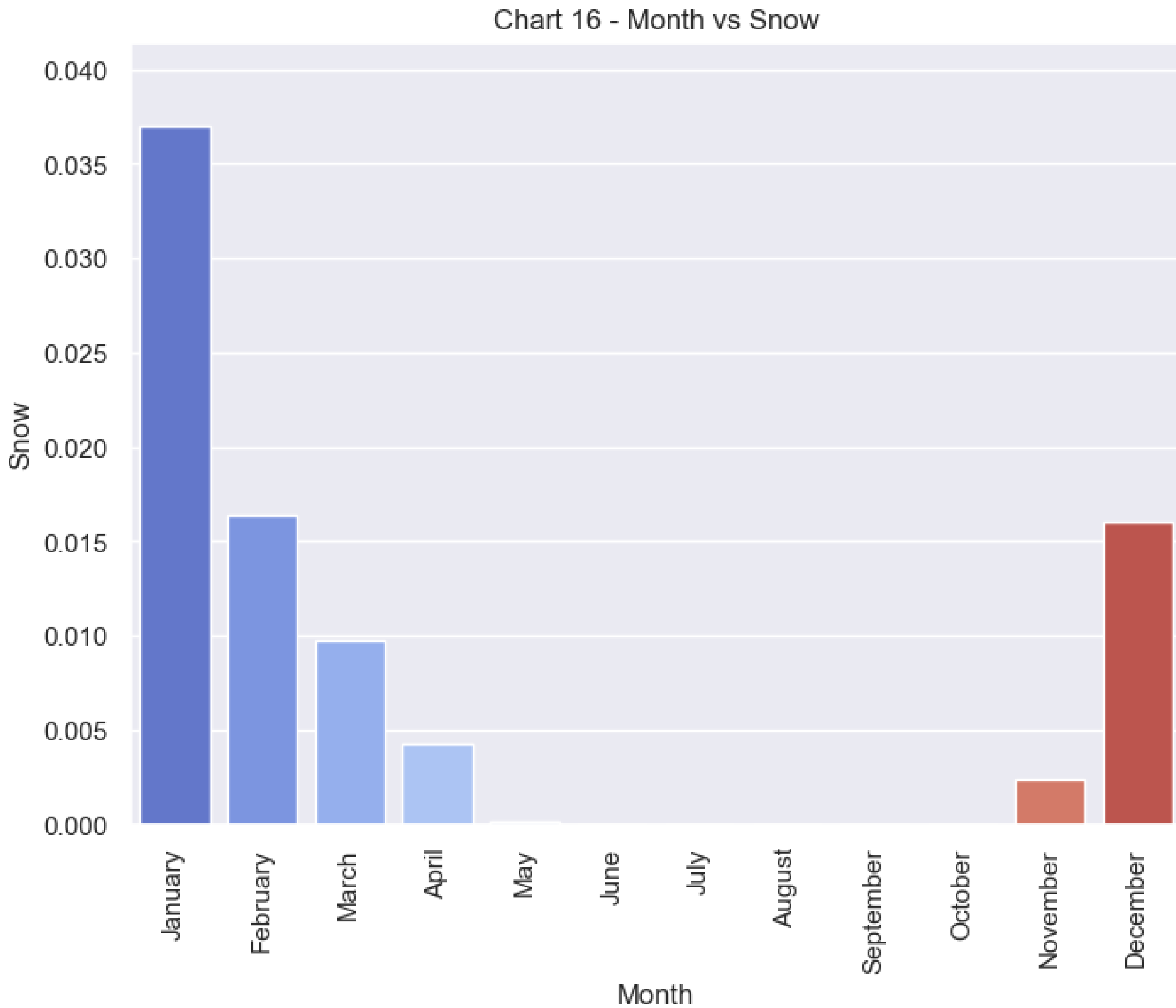
In [32]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['weather_type'], y=data['snow_1h'], errwidth=0, palette='coo
plt.title('Chart 15 - Weather Type vs Snow')
plt.xlabel('Weather Type')
plt.ylabel('Snow')
```

Out[32]:
```
Text(0, 0.5, 'Snow')
```



In [33]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['month'], y=data['snow_1h'], errwidth=0, palette='coolwarm')
plt.title('Chart 16 - Month vs Snow')
plt.xlabel('Month')
plt.xticks(rotation=90)
plt.ylabel('Snow')
```

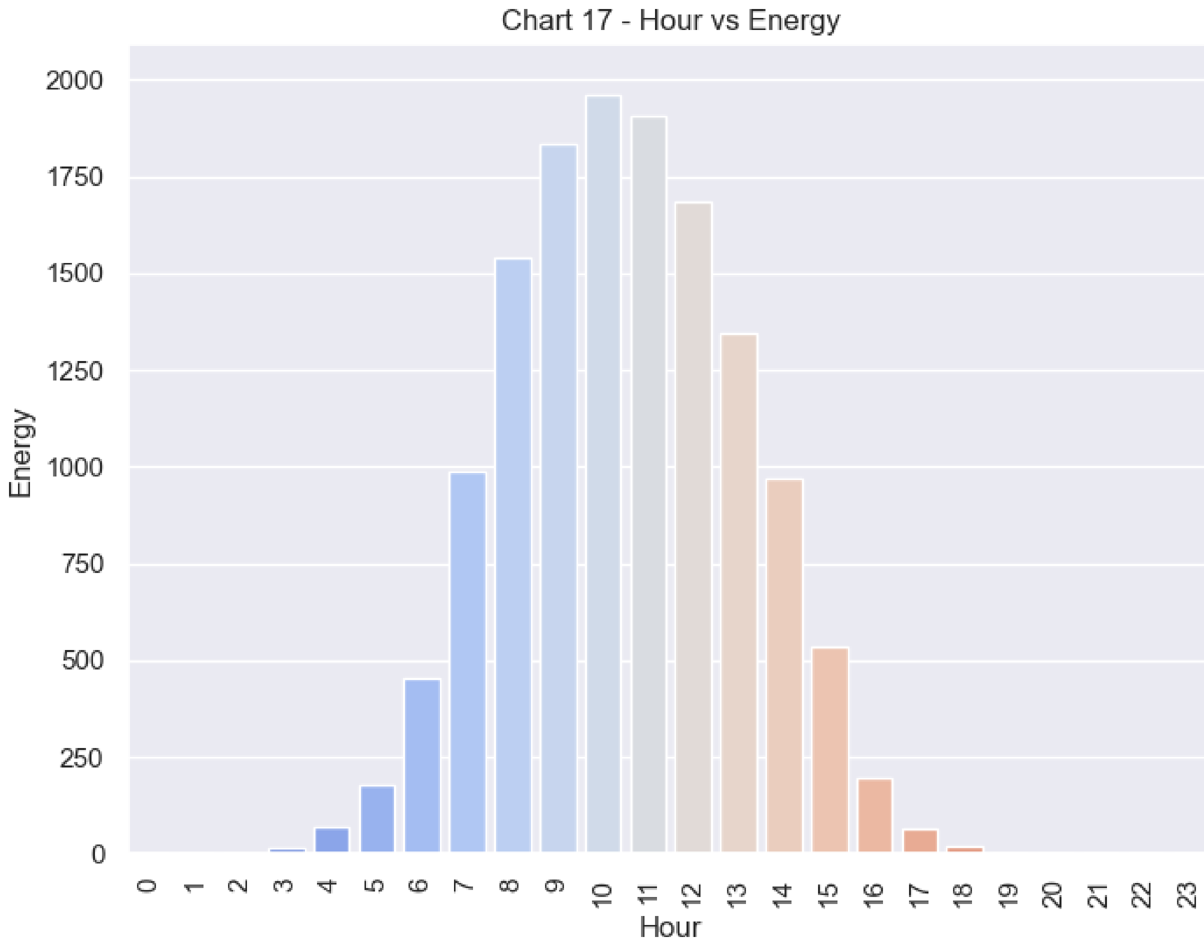
Out[33]: `Text(0, 0.5, 'Snow')`



The winter season have all the snow but out of that January month has highest snow than others.

In [34]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['hour'], y=data['Energy delta[Wh]'], errwidth=0, palette='co
plt.title('Chart 17 - Hour vs Energy')
plt.xlabel('Hour')
plt.xticks(rotation=90)
plt.ylabel('Energy')
```

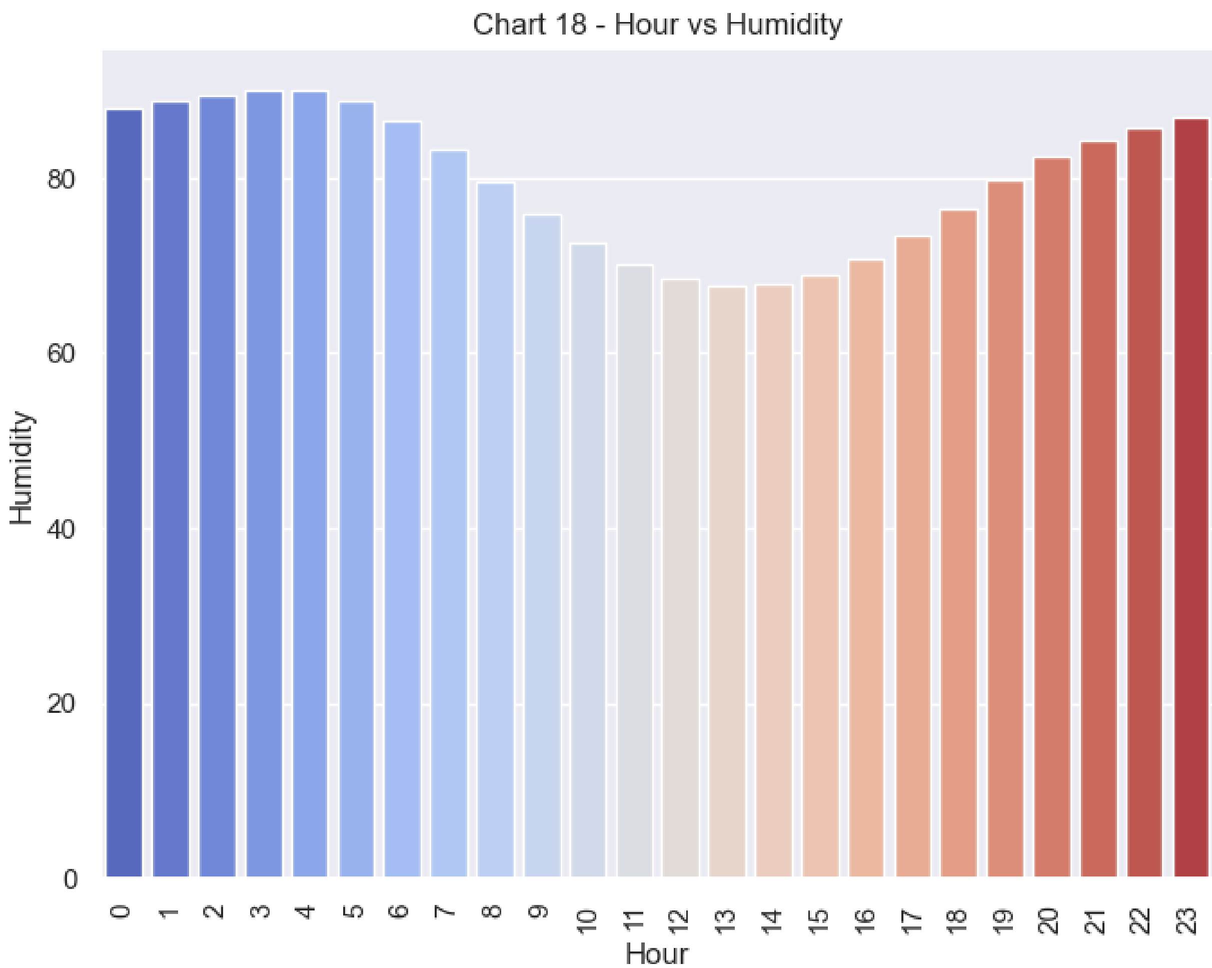
Out[34]:
```
Text(0, 0.5, 'Energy')
```



The energy curve is a clean bell curve. We can see as the hour increase from 0 the level of energy increases till hour 10 and from hour 11 it start decreasing all the way to the end.

In [35]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['hour'], y=data['humidity'], errwidth=0, palette='coolwarm')
plt.title('Chart 18 - Hour vs Humidity')
plt.xlabel('Hour')
plt.xticks(rotation=90)
plt.ylabel('Humidity')
```

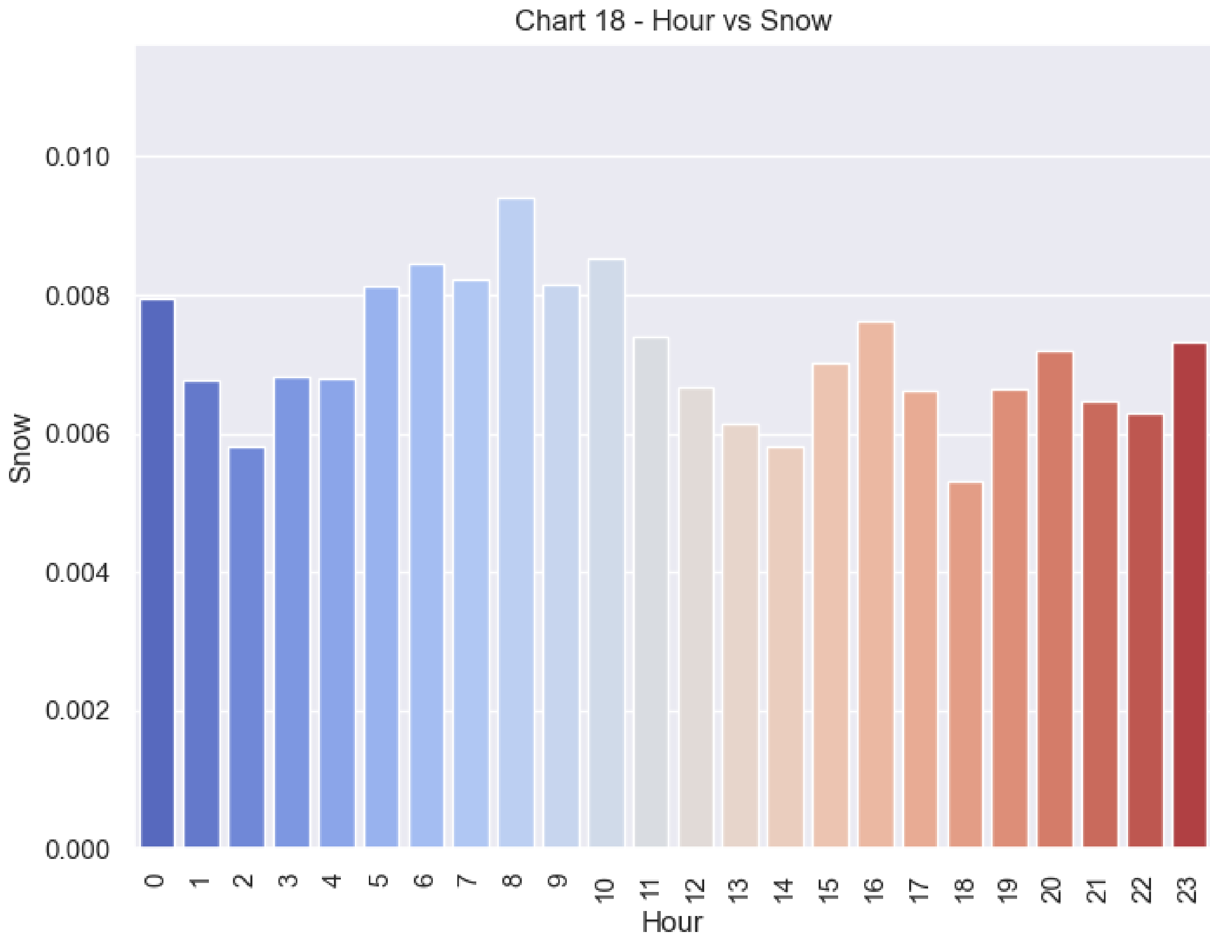
Out[35]:
```
Text(0, 0.5, 'Humidity')
```



After looking into the chart we find out that humidity level increase very slowly from hour 0 to hour 4 and after that it start decreasing all the way to hour 14. From hour 15 it again start increasing to the end.

In [36]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['hour'], y=data['snow_1h'], errwidth=0, palette='coolwarm')
plt.title('Chart 18 - Hour vs Snow')
plt.xlabel('Hour')
plt.xticks(rotation=90)
plt.ylabel('Snow')
```

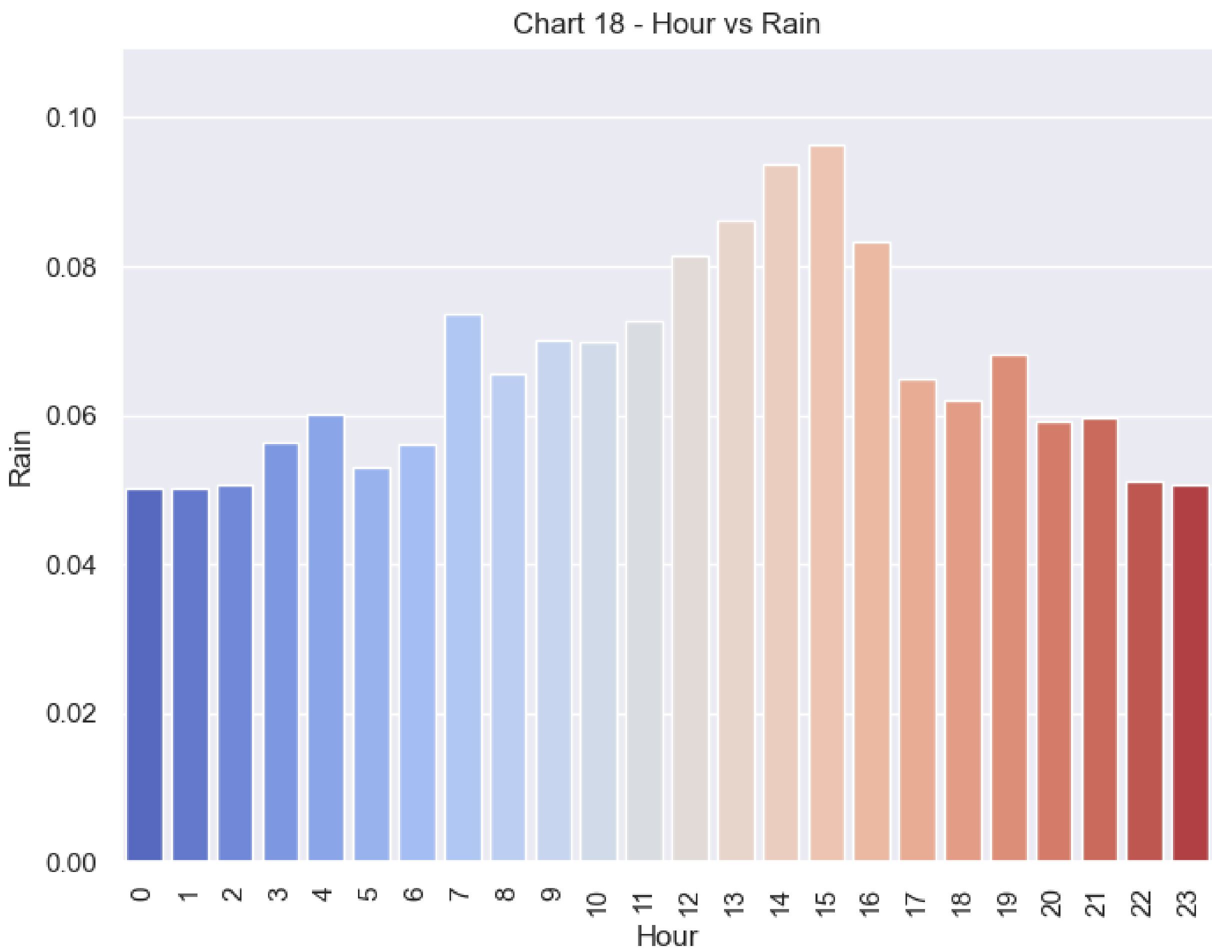
Out[36]:
```
Text(0, 0.5, 'Snow')
```



We can't say which hour will the snow fall but it is clear from the chart that the 8th hour has highest snow fall than others.

In [37]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['hour'], y=data['rain_1h'], errwidth=0, palette='coolwarm')
plt.title('Chart 18 - Hour vs Rain')
plt.xlabel('Hour')
plt.xticks(rotation=90)
plt.ylabel('Rain')
```

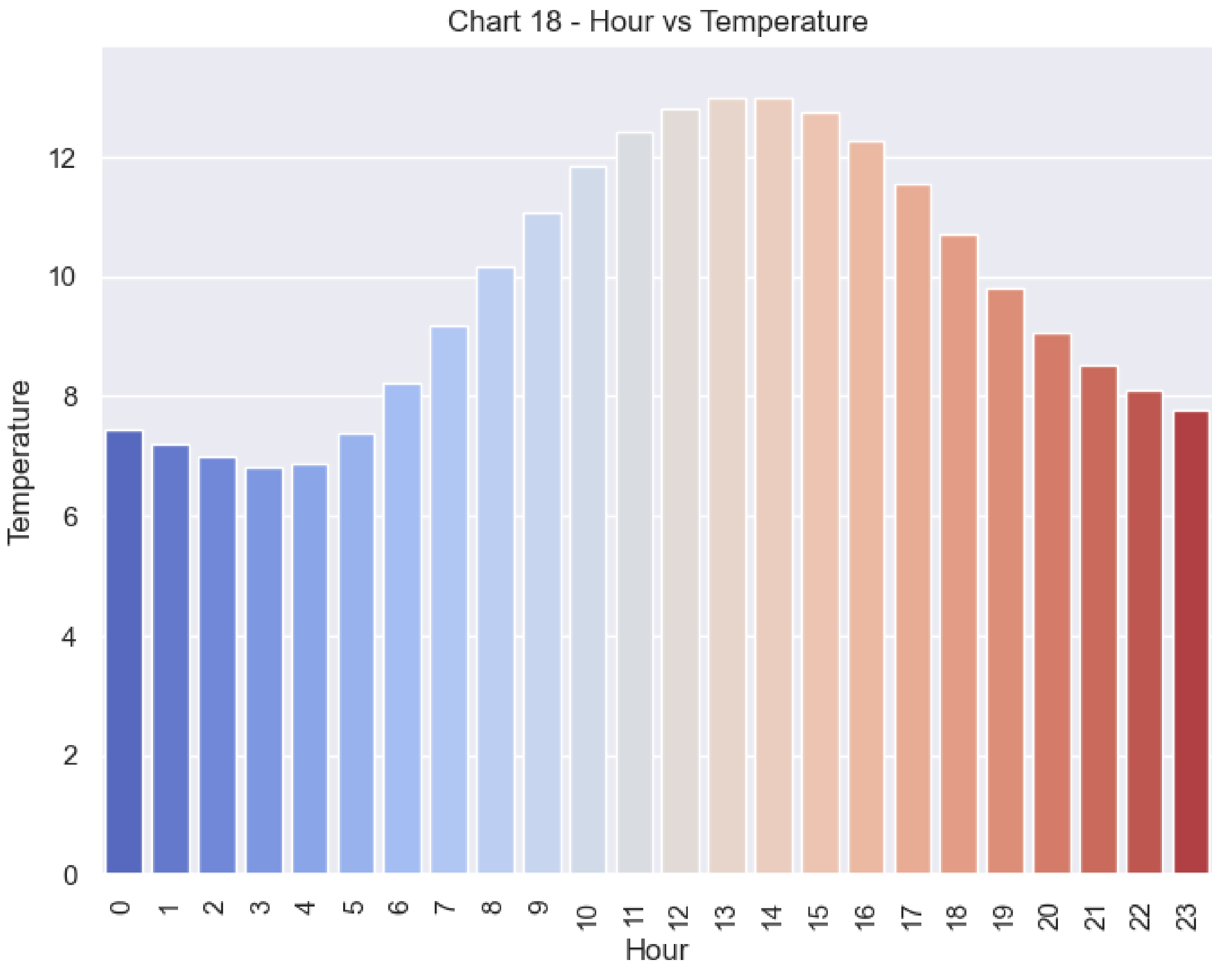
Out[37]:
```
Text(0, 0.5, 'Rain')
```



The 15th hour is peak hour for rain.

In [38]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=data['hour'], y=data['temp'], errwidth=0, palette='coolwarm')
plt.title('Chart 18 - Hour vs Temperature')
plt.xlabel('Hour')
plt.xticks(rotation=90)
plt.ylabel('Temperature')
```

Out[38]:
```
Text(0, 0.5, 'Temperature')
```



The relation between temperature and hour is understandable. The temperature will be less in night and morning but it will rise to its highest in noon and after that it will start decreasing in evening.

In [ ]:

In [ ]:

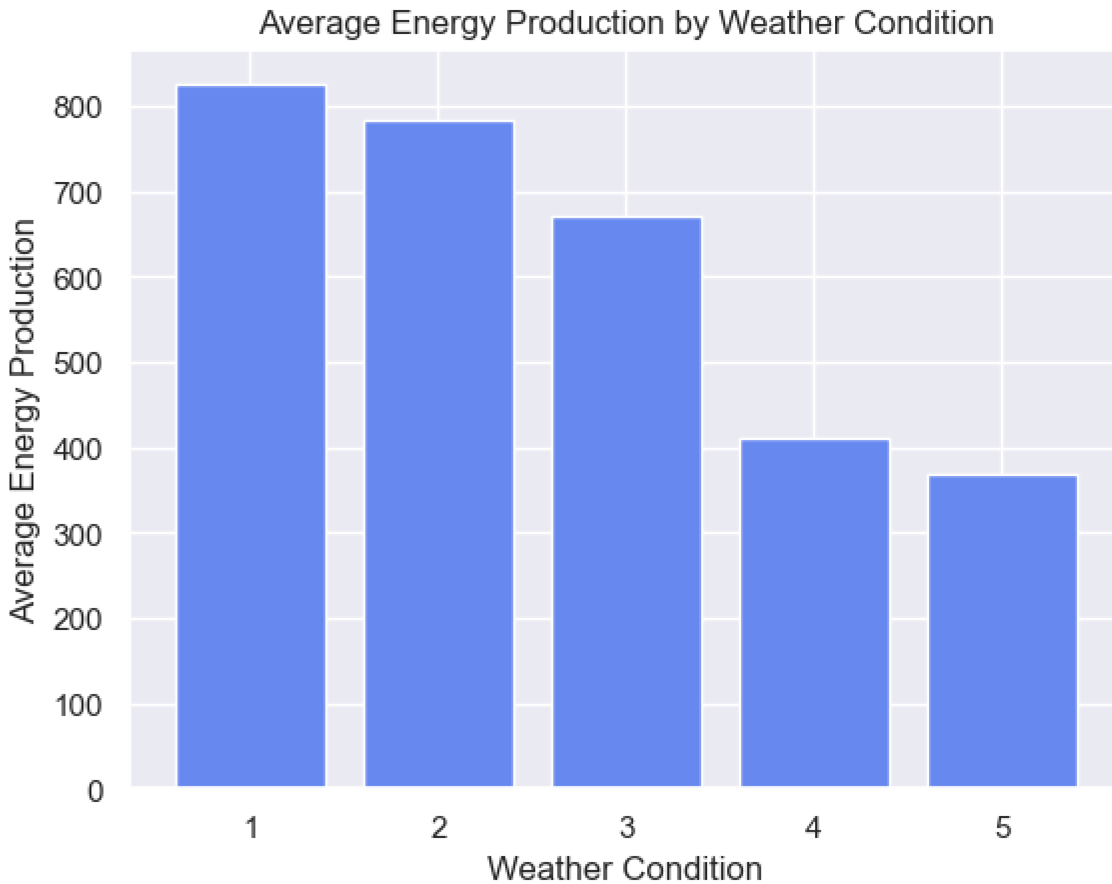
In [ ]:

In [ ]:

Here are the average energy production by weather condition. As you can see, weather condition 1,2, and 3 have higher average energy production than the other weather conditions.

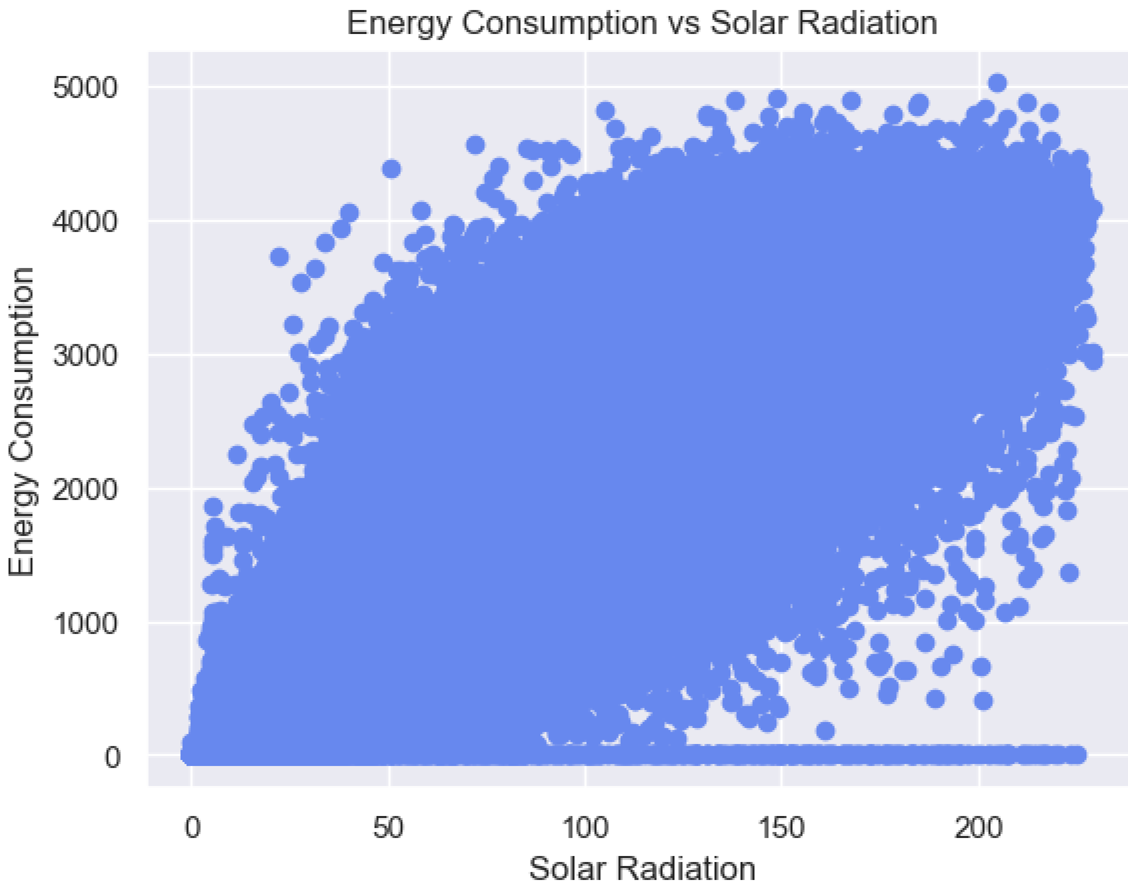
In [39]:

```
avg_energy = data.groupby("weather_type")["Energy delta[Wh]"].mean()


plt.bar(avg_energy.index, avg_energy.values)
plt.title("Average Energy Production by Weather Condition")
plt.xlabel("Weather Condition")
plt.ylabel("Average Energy Production")
plt.show()
```



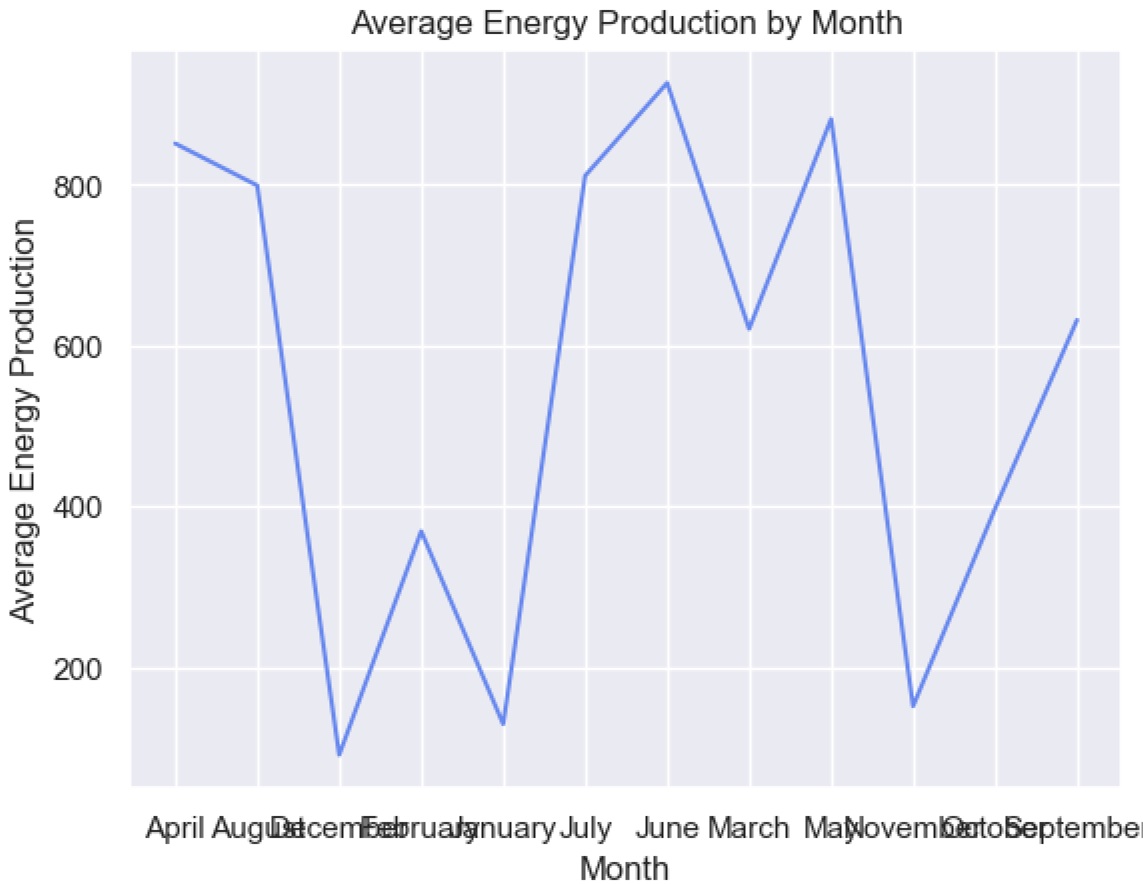
Next, I will check the relationship between energy consumption and solar radiation. The scatter plot below shows a positive correlation between these two factors.

In [40]:
```
data["Time"] = pd.to_datetime(data["Time"])


plt.scatter(data["GHI"], data["Energy delta[Wh]"])
plt.title("Energy Consumption vs Solar Radiation")
plt.xlabel("Solar Radiation")
plt.ylabel("Energy Consumption")
plt.show()
```



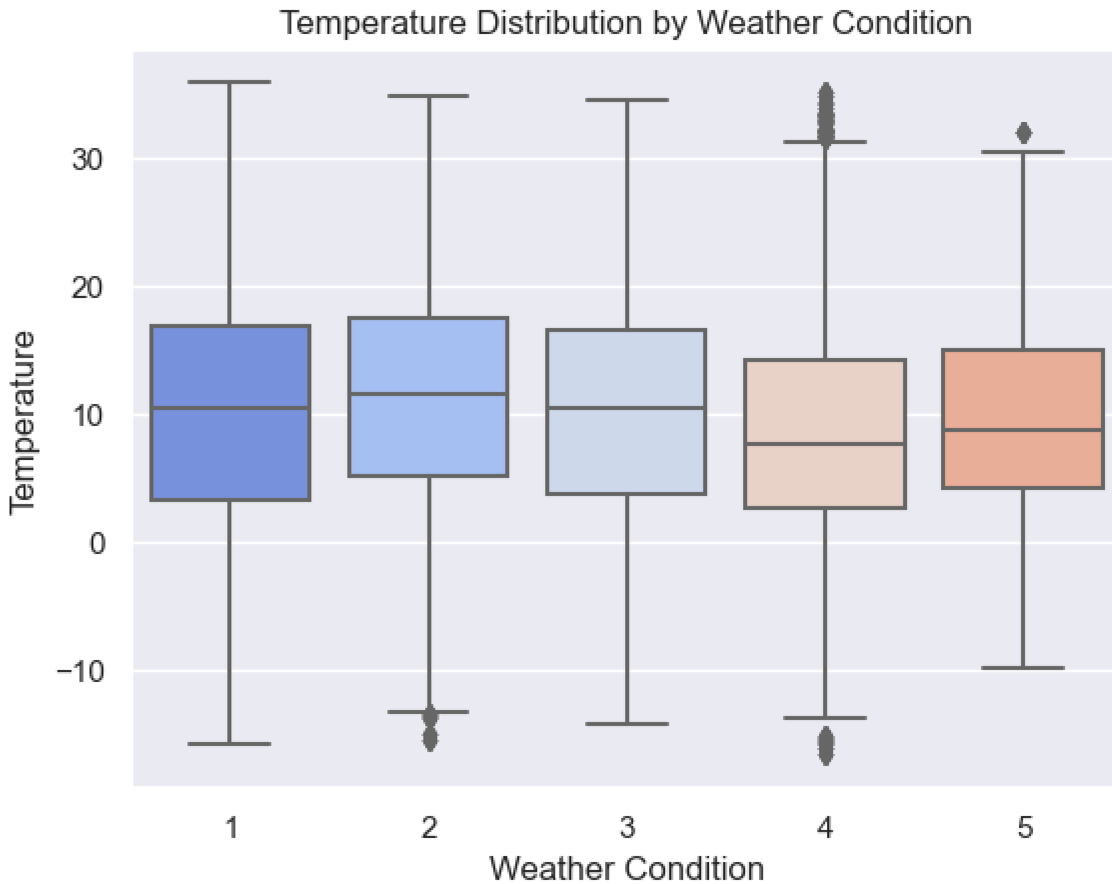
To visualize the average energy production by month, I will plot this data.

In [72]:
```
avg_energy = data.groupby("month")["Energy delta[Wh]"].mean()
plt.plot(avg_energy.index, avg_energy.values)
plt.title("Average Energy Production by Month")
plt.xlabel("Month")
plt.ylabel("Average Energy Production")
plt.show()
```



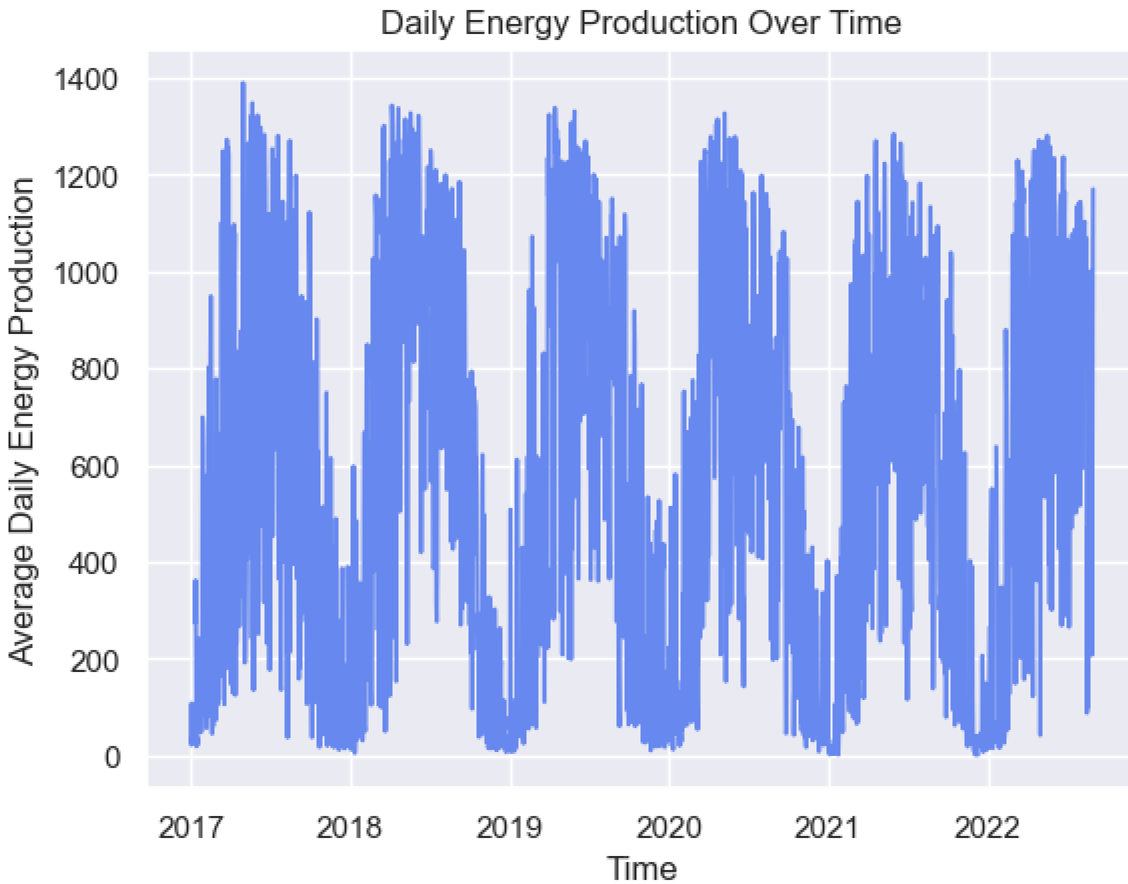
The graph above tells us that during the 6th month, this is when the highest levels of energy producton are done. Months 2 and 12 are the lowest. Below we can see the distribution of temperature during different weather conditions.

In [83]:
```
sns.boxplot(x="weather_type", y="temp", data=data)
plt.title("Temperature Distribution by Weather Condition")
plt.xlabel("Weather Condition")
plt.ylabel("Temperature")
plt.show()
```



Below is the daily energy production over time.

In [75]:
```
data["Time"] = pd.to_datetime(data["Time"])

# Set the "Time" column as the index of the DataFrame
data.set_index("Time", inplace=True)

# Resample the data to a daily frequency and calculate the average energy produ
daily_energy = data["Energy delta[Wh]"].resample("D").mean()

# Plot the daily energy production over time
plt.plot(daily_energy.index, daily_energy.values)
plt.title("Daily Energy Production Over Time")
plt.xlabel("Time")
plt.ylabel("Average Daily Energy Production")
plt.show()
```



Challenges of this Project I came into this class with little to no knowledge of Python. This course used Python to conduct Advanced Data Analytics as well as learn machine learning techniques. The challenges of life changing events and lack of Python knowledge, made this difficult. However, I am proud to say that i have learned this skill and very thankful for Professor Taylor for the excellent instructorship. I will continue to learn and use Python for work, school, and any other projects that I come across.Conclusion The analysis of the data proved to be very useful and a perfect start for statiscal analysis that will help future research. Using various statistical and machine learning techniques I did identify patterns and trends in the data that can help others to develop predictive models that can be used to optimize renewable energy production. The other possible research exploration areas that this analysis can assist are the following: a. Finding out where to place energy production plants. b. Finding an optimal way to supplement different types of energy so that renewable energy is most used. c. Optimize energy redundancy to prevent power outages in various sized populations (e.g. large, small, etc.). d. Optimize and subsidize enough renewable energy so that smaller forms of energy like coal, oil, gas, etc. are being minimally used. This would supposedly decrease the CO_2 emissions. Thanks to professor Taylor for making this course such an amazing course.