In [1]:

```
import datetime
import time
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
```

In [2]:

```
file_path=input("Enter the path to your csv file: ")
file_path=str(file_path)
#/Users/thapliyaa/Desktop/CS_IA/DOM VA Power Detailed Energy Usage.csv
cost_KWH=input("How much do you pay per KWH? Insert your number as a decimal. ")
cost_KWH=float(cost_KWH)
total_system_cost=input("How much did you pay for your solar panels? Insert your number as a whole number with no commas. ")
total_system_cost=int(total_system_cost)
```

```
Enter the path to your csv file: /Users/thapliyaa/Desktop/CS_IA/DOM VA Power Detailed Energy Usage.csv
How much do you pay per KWH? Insert your number as a decimal. 0.11
How much did you pay for your solar panels? Insert your number as a whole number with no commas. 16723
```

In [3]:

```
#Reads the usage data csv file and turns it into a dataframe
energydata=pd.read_csv(file_path, dtype='unicode')

#Deletes unnecessary columns of data
energydata.drop(['Account No', 'Recorder ID', 'Day', 'Month', 'Year','New Date'], axis=1, inplace=True)

#Deletes unnecessary rows of data
energydata.drop(index=energydata.iloc[0:21].index.tolist(), inplace=True)

#Changes the date column to a datetime format
energydata['DATE']=pd.to_datetime(energydata['Date'].values)
del energydata['Date']

energydata.rename(columns={'TOTAL Consumption':'KWH Consumption', 'TOTAL Production':'KWH Production'}, inplace=True)

#Converts data type from float64 to float32 to reduce memory
energydata['KWH Consumption']=energydata['KWH Consumption'].astype(np.float32)
energydata['KWH Production']=energydata['KWH Production'].astype(np.float32)
```

In [4]:

```
#Prints the format of dataframe: columns names and first five rows
energydata.head()
```

Out[4]:

	KWH Consumption	KWH Production	DATE
21	11.100000	15.874	2016-01-27
22	14.900000	16.194	2016-01-28
23	19.700001	14.296	2016-01-29
24	10.800000	20.319	2016-01-30
25	9.600000	20.886	2016-01-31

In [5]:

```
#Prints general statistical information about data
energydata.describe()
```

Out[5]:

	KWH Consumption	KWH Production
count	1757.000000	1757.000000
mean	19.087934	18.717537
std	8.901297	8.996354
min	0.400000	0.000000
25%	12.800000	11.549000
50%	16.900000	20.045000
75%	23.500000	26.108000
max	71.199997	35.613998

In [ ]:

```
#Prints data type and data size information
#energydata.info()
```

In [7]:

```
#Splits production data into separate dataframe for prediction
productionY=energydata['KWH Production']
productionY.index=energydata['DATE']

#Features that can be predicted
productionX = energydata[['KWH Consumption', 'KWH Production']]
#Target Variable: what is getting predicted
productionY = energydata['KWH Production']

from sklearn.model_selection import train_test_split
#Tuple unpacking to grab train and test data sets
#Test size is the % of data set that is allocated to the test set
X_train, X_test, y_train, y_test = train_test_split(productionX, product
ionY, test_size=0.7)

from sklearn.linear_model import LinearRegression
from sklearn import linear_model
#Creates Linear Regression object
production_linearmodel = LinearRegression()
#Fits linear model
production_linearmodel.fit(X_train, y_train)

#Array of predictions of KWH Production
production_predictions=production_linearmodel.predict(X_test)

#Added predicted production data to a separate csv file
productiondata=pd.read_csv("/Users/thapliyaa/Desktop/CS_IA/PredictedProd
uction.csv", dtype='unicode')
productiondata['Prediction']=production_predictions.tolist()
productiondata['DATE']=pd.to_datetime(productiondata['11/21/20'].values)
del productiondata['11/21/20']
```

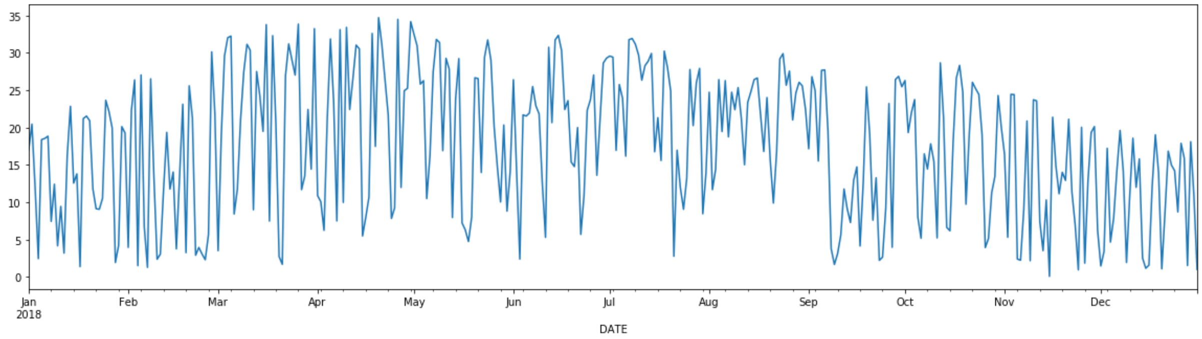
In [8]:

```
#Splits consumption data into separate dataframe for prediction
consumptionY=energydata['KWH Consumption']
consumptionY.index=energydata['DATE']
consumptionY.head()

#Features that can be predicted
consumptionX = energydata[['KWH Consumption', 'KWH Production']]
#Target Variable: what is getting predicted
consumptionY = energydata['KWH Consumption']

#Tuple unpacking to grab train and test data sets
#Test size is the % of data set that is allocated to the test set
X_train, X_test, y_train, y_test = train_test_split(consumptionX, consum
ptionY, test_size=0.7)

#Creates Linear Regression object
consumption_linearmodel = LinearRegression()
#Fits linear model
consumption_linearmodel.fit(X_train, y_train)

#Array of predictions of KWH Consumption
consumption_predictions=consumption_linearmodel.predict(X_test)

#Added predicted consumption data to a separate csv file
consumptiondata=pd.read_csv("/Users/thapliyaa/Desktop/CS_IA/PredictedCon
sumption.csv", dtype='unicode')
consumptiondata['Prediction']=consumption_predictions.tolist()
consumptiondata['DATE']=pd.to_datetime(consumptiondata['11/21/20'].value
s)
del consumptiondata['11/21/20']
```

In [9]:

```
data=input("Which data would you like to see? (production or consumption)")
data=str(data)
graph=input("What kind of graph would you like to see? (line, scatter, or bar) ")
graph=str(graph)
ans=input("Would you like to specify with time intervals? (yes or no)")
ans=str(ans)

if data=="production":
    if graph=="line":
            if ans=="yes":
                start=input("What day would you like to start on? (YYYY-MM-DD)")
                start=str(start)
                end=input("What day would you like to end on? (YYYY-MM-DD)")
                end=str(end)
                productionY.loc[start:end].plot(figsize=(20,5))
    else:
                 productionY.plot(figsize=(20,5))
    if graph=="scatter":
        plt.scatter(energydata['DATE'], productionY)
    if graph=="bar":
        plt.bar(energydata['DATE'], productionY)

if data=="consumption":
    if graph=="line":
        if ans=="yes":
                start=input("What day would you like to start on? (YYYY-MM-DD) ")
                start=str(start)
                end=input("What day would you like to end on? (YYYY-MM-DD) ")
                end=str(end)
                consumptionY.loc[start:end].plot(figsize=(20,5))
    else:
                 consumptionY.plot(figsize=(20,5))
    if graph=="scatter":
        plt.scatter(energydata['DATE'], consumptionY)
    if graph=="bar":
        plt.bar(energydata['DATE'], consumptionY)
```

```
Which data would you like to see? (production or consumption)production
What kind of graph would you like to see? (line, scatter, or bar) line
Would you like to specify with time intervals? (yes or no)yes
What day would you like to start on? (YYYY-MM-DD)2018-01-01
What day would you like to end on? (YYYY-MM-DD)2018-12-31
```



In [ ]:

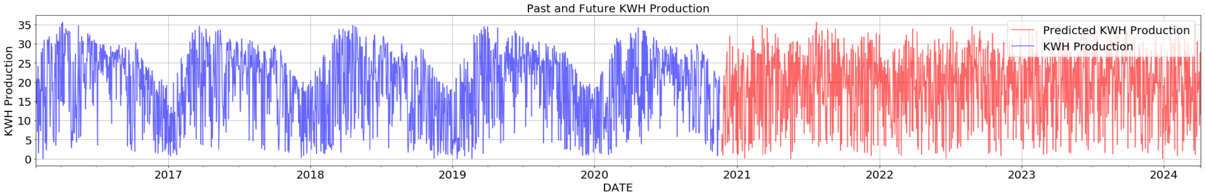
```python
#from sklearn import metrics
#print("Mean absolute error: "+str(metrics.mean_absolute_error(y_test, production_predictions)))
#print("Mean squared error: "+str(metrics.mean_squared_error(y_test, production_predictions)))
#print("Square root of mean squared error: "+str(np.sqrt(metrics.mean_squared_error(y_test, production_predictions))))
#metrics.r2_score
#plt.scatter(y_test, production_predictions)
```

In [10]:

```python
y1=productiondata['Prediction']
y1.index=productiondata['DATE']
p_graph1=y1.plot(figsize=(30,5), color='Red', alpha=0.6, fontsize=20)
p_graph1.set_ylabel('Predicted KWH Consumption')
p_graph2=productionY.plot(figsize=(30,5), title='Past and Future KWH Production', grid=True, fontsize=20, alpha=0.6, color='Blue')
p_graph2.set_xlabel('DATE', fontsize=20)
p_graph2.set_ylabel('KWH Production', fontsize=20)
p_graph2.title.set_size(20)
plt.tight_layout()
p_graph2.legend(['Predicted KWH Production','KWH Production'], fontsize=20, loc='upper right')
```

Out[10]:

```
<matplotlib.legend.Legend at 0x10e9e9710>
```

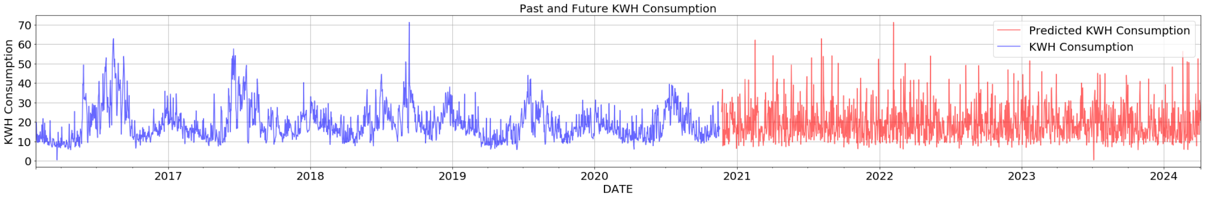


In [11]:

```
y2=consumptiondata['Prediction']
y2.index=consumptiondata['DATE']
c_graph1=y2.plot(figsize=(30,5), color='Red', alpha=0.6, fontsize=20)
c_graph1.set_ylabel('Predicted KWH Consumption')
c_graph2=consumptionY.plot(figsize=(30,5), title='Past and Future KWH Consumption', grid=True, fontsize=20, alpha=0.6, color='Blue')
c_graph2.set_xlabel('DATE', fontsize=20)
c_graph2.set_ylabel('KWH Consumption', fontsize=20)
c_graph2.title.set_size(20)
plt.tight_layout()
c_graph2.legend(['Predicted KWH Consumption','KWH Consumption'], fontsize=20, loc='upper right')
```

Out[11]:

```
<matplotlib.legend.Legend at 0x10ea22860>
```



In [12]:

```
cumulative_df=pd.DataFrame(energydata, columns=['DATE'])
consumption_col=energydata.loc[:,'KWH Consumption']
consumption_arr=consumption_col.values
consumption_arr=consumption_arr.astype(int)

production_col=energydata.loc[:, 'KWH Production']
production_arr=production_col.values

for i in range(1,1757):
    production_arr[i]=(production_arr[i-1]+production_arr[i])
    #print(production_arr[i])
for i in range(1,1757):
    consumption_arr[i]=(consumption_arr[i-1]+consumption_arr[i])
    #consumption_arr[i]=consumption_arr[i]*cost_KWH
    #print(consumption_arr[i])
```

In [13]:

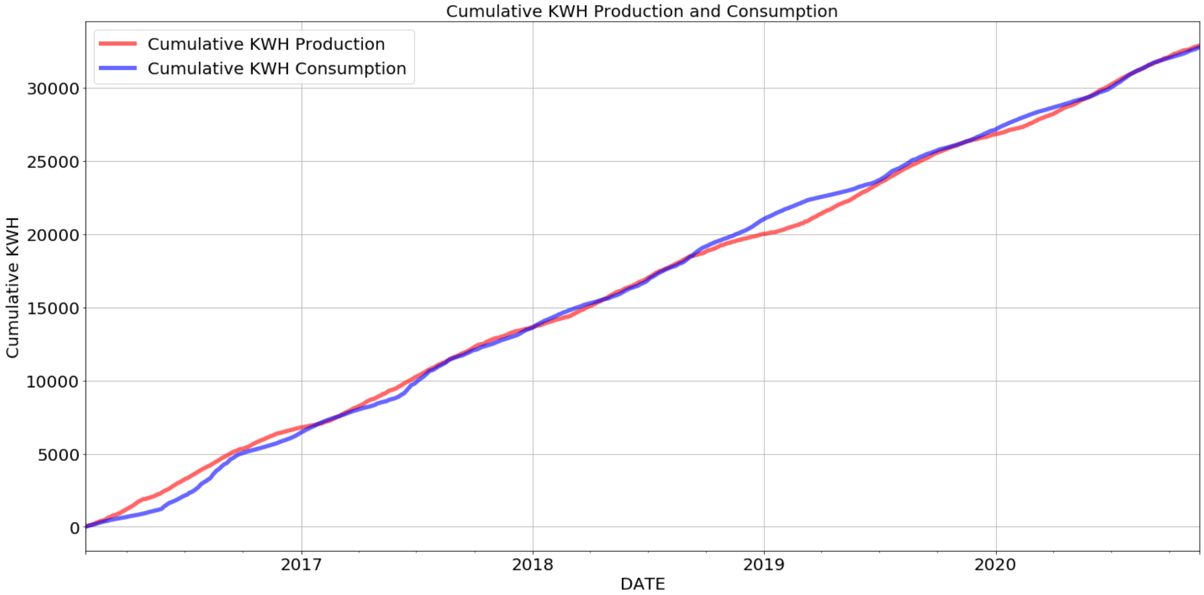
```
cumulative_df['Cumulative Consumption']=consumption_arr.tolist()
cumulative_df['Cumulative Production']=production_arr.tolist()
```

In [14]:

```
totalY=cumulative_df['Cumulative Consumption']
totalY.index=cumulative_df['DATE']
totalY.head()

y3=cumulative_df['Cumulative Production']
y3.index=cumulative_df['DATE']
t_graph1=y3.plot(figsize=(20,10), color='Red', alpha=0.6, fontsize=20, linewidth=5)
t_graph2=totalY.plot(figsize=(20,10), title='Cumulative KWH Production and Consumption',
                     grid=True, fontsize=20, alpha=0.6, color='Blue', linewidth=5)
t_graph2.set_xlabel('DATE', fontsize=20)
t_graph2.set_ylabel('Cumulative KWH', fontsize=20)
t_graph2.title.set_size(20)
plt.tight_layout()
t_graph2.legend(['Cumulative KWH Production','Cumulative KWH Consumption'], fontsize=20, loc='upper left')
```

Out[14]:

```
<matplotlib.legend.Legend at 0x11fb1a0f0>
```

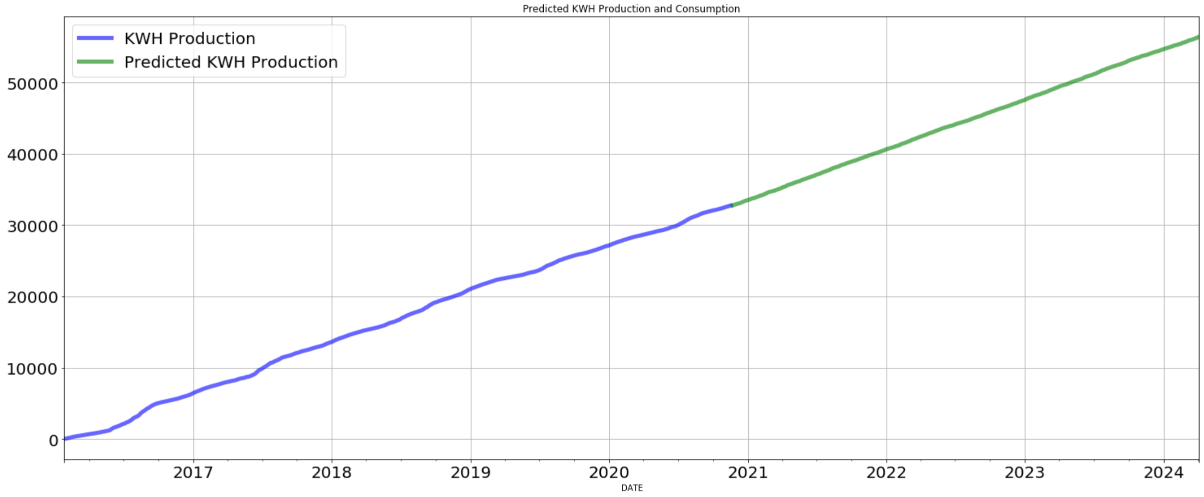


In [15]:

```
predictions_df=pd.DataFrame(consumptiondata, columns=['DATE'])

pred_consumption=consumptiondata.loc[:,'Prediction']
pred_consumption_arr=pred_consumption.values

pred_production=productiondata.loc[:,'Prediction']
pred_production_arr=pred_production.values

pred_consumption_arr[0]=consumption_arr[1756]
pred_production_arr[0]=production_arr[1756]
#predictions_df
for i in range(1,1230):
    pred_production_arr[i]=(pred_production_arr[i-1]+pred_production_arr[i])
    #print(production_arr[i])

for i in range(1,1230):
    pred_consumption_arr[i]=(pred_consumption_arr[i-1]+pred_consumption_arr[i])
    #consumption_arr[i]=consumption_arr[i]*cost_KWH
    #print(consumption_arr[i])

predictions_df['Predicted Consumption']=pred_consumption_arr.tolist()
predictions_df['Predicted Production']=pred_production_arr.tolist()
#predictions_df
```

In [16]:
```
totalY=cumulative_df['Cumulative Consumption']
totalY.index=cumulative_df['DATE']
t_graph2=totalY.plot(figsize=(25,10), title='Cumulative KWH Production a
nd Consumption',
                     grid=True, fontsize=20, alpha=0.6, color='Blue', li
newidth=5)
#y3=cumulative_df['Cumulative Production']
#y3.index=cumulative_df['DATE']
#t_graph1=y3.plot(figsize=(20,10), color='Red', alpha=0.6, fontsize=20,
 linewidth=5)

#t_graph2.set_xlabel('DATE', fontsize=20)
#t_graph2.set_ylabel('Cumulative KWH', fontsize=20)
#t_graph2.title.set_size(20)
#plt.tight_layout()

predY=predictions_df['Predicted Consumption']
predY.index=predictions_df['DATE']
t_graph3=predY.plot(figsize=(25,10), title='Predicted KWH Production and
Consumption',
                    grid=True, fontsize=20, alpha=0.6, color='Green', li
newidth=5)

y4=predictions_df['Predicted Production']
y4.index=predictions_df['DATE']
#t_graph4=y4.plot(figsize=(20,15), grid=True, fontsize=20, alpha=0.6, co
lor='Orange', linewidth=5)

t_graph3.legend(['KWH Production','Predicted KWH Production'], fontsize=
20, loc='upper left')
```

Out[16]:
```
<matplotlib.legend.Legend at 0x11fa40a90>
```



In [17]:

```
saved=int(production_arr[1756]*cost_KWH)
print("Total saved from 1/27/2016 to 11/20/2020: $"+str(saved))
pred_saved=int((pred_production_arr[1229]-production_arr[1756])*cost_KWH
)
print("Total predicted to save from 11/21/2020 to 4/24/2024: $"+str(pred
_saved))
total_saved=int(saved+pred_saved)
print("Total savings in 7.4 years: $"+str(total_saved))
total_system_cost=16723
payback_period=float(round((total_system_cost/total_saved), 2)*7.4)
print("Client will break even in "+str(payback_period)+" years")
```

```
Total saved from 1/27/2016 to 11/20/2020: $3617
Total predicted to save from 11/21/2020 to 4/24/2024: $2535
Total savings in 7.4 years: $6152
Client will break even in 20.128000000000004 years
```

In [ ]: