In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
#importing data into the notebook using pandas
df = pd.read_csv(r'C:\Users\SamDutse\Documents\ANALYSIS\EnergyD202.csv')
```

In [4]:

```
#performing a check on data to ensure readiness for use
df.head(2)
```

Out[4]:

	TYPE	DATE	START TIME	END TIME	USAGE	UNITS	COST($)	NOTES
0	Electric usage	10/22/2016	0:00	0:14	0.01	kWh	0.0	NaN
1	Electric usage	10/22/2016	0:15	0:29	0.01	kWh	0.0	NaN

In [6]:

```
df.tail(2)
```

Out[6]:

	TYPE	DATE	START TIME	END TIME	USAGE	UNITS	COST($)	NOTES
70366	Electric usage	10/24/2018	23:30	23:44	0.03	kWh	0.01	NaN
70367	Electric usage	10/24/2018	23:45	23:59	0.03	kWh	0.01	NaN

In [7]:

```
df.size
```

Out[7]:

```
562944
```

In [8]:

```
df.shape
```

Out[8]:

```
(70368, 8)
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70368 entries, 0 to 70367
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   TYPE        70368 non-null  object 
 1   DATE        70368 non-null  object 
 2   START TIME  70368 non-null  object 
 3   END TIME    70368 non-null  object 
 4   USAGE       70368 non-null  float64
 5   UNITS       70368 non-null  object 
 6   COST($)     70368 non-null  float64
 7   NOTES       0 non-null      float64
dtypes: float64(3), object(5)
memory usage: 4.3+ MB
```

In [11]:

```
df.describe()
```

Out[11]:

	USAGE	COST($)	NOTES
count	70368.000000	70368.000000	0.0
mean	0.121941	0.024684	NaN
std	0.210507	0.042646	NaN

min	0.000000	0.000000	NaN
25%	0.030000	0.010000	NaN
50%	0.050000	0.010000	NaN
75%	0.120000	0.020000	NaN
max	2.360000	0.650000	NaN

In [12]:
```
df.describe().transpose()
```

Out[12]:

	count	mean	std	min	25%	50%	75%	max
USAGE	70368.0	0.121941	0.210507	0.0	0.03	0.05	0.12	2.36
COST($)	70368.0	0.024684	0.042646	0.0	0.01	0.01	0.02	0.65
NOTES	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
TYPE                0
DATE                0
START TIME          0
END TIME            0
USAGE               0
UNITS               0
COST($)             0
NOTES           70368
dtype: int64
```

In [14]:
```
#All the values in the NOTE column are nulls so we have to drop the column
#dropping the null NOTE column
df=df.drop(['NOTES'], axis=1)
```

In [16]:
```
df.head(2)
```

Out[16]:

	TYPE	DATE	START TIME	END TIME	USAGE	UNITS	COST($)
0	Electric usage	10/22/2016	0:00	0:14	0.01	kWh	0.0
1	Electric usage	10/22/2016	0:15	0:29	0.01	kWh	0.0

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
TYPE          0
DATE          0
START TIME    0
END TIME      0
USAGE         0
UNITS         0
COST($)       0
dtype: int64
```

In [19]:
```
df.describe()
```

Out[19]:

	USAGE	COST($)
count	70368.000000	70368.000000
mean	0.121941	0.024684
std	0.210507	0.042646

min	0.000000	0.000000
25%	0.030000	0.010000
50%	0.050000	0.010000
75%	0.120000	0.020000
max	2.360000	0.650000

changing the date datatype from object to datetime for further data manipulation

In [20]:

```
df['DATE'] = pd.to_datetime(df['DATE'])
df['DATE']
```

Out[20]:

```
0       2016-10-22
1       2016-10-22
2       2016-10-22
3       2016-10-22
4       2016-10-22
           ...
70363   2018-10-24
70364   2018-10-24
70365   2018-10-24
70366   2018-10-24
70367   2018-10-24
Name: DATE, Length: 70368, dtype: datetime64[ns]
```

seperating the DATE into DAY, WEEKDAY, MONTH AND YEAR

In [21]:

```
#seperating the date column into day, month and year
df['Month'] = df['DATE'].dt.month
df['Day'] = df['DATE'].dt.day
df['Year'] = df['DATE'].dt.year
df['Week_Day'] = df['DATE'].dt.weekday
```

In [22]:

```
df.head(3)
```

Out[22]:

	TYPE	DATE	START TIME	END TIME	USAGE	UNITS	COST($)	Month	Day	Year	Week_Day
0	Electric usage	2016-10-22	0:00	0:14	0.01	kWh	0.0	10	22	2016	5
1	Electric usage	2016-10-22	0:15	0:29	0.01	kWh	0.0	10	22	2016	5
2	Electric usage	2016-10-22	0:30	0:44	0.01	kWh	0.0	10	22	2016	5

0=Monday, 1=Tuesday, 2=Wednesday, 3=Thursday, 4=Friday, 5=Saturday, 6=Sunday

In [24]:

```
#checking data type again
df.dtypes
```

Out[24]:

```
TYPE                  object
DATE          datetime64[ns]
START TIME            object
END TIME              object
```

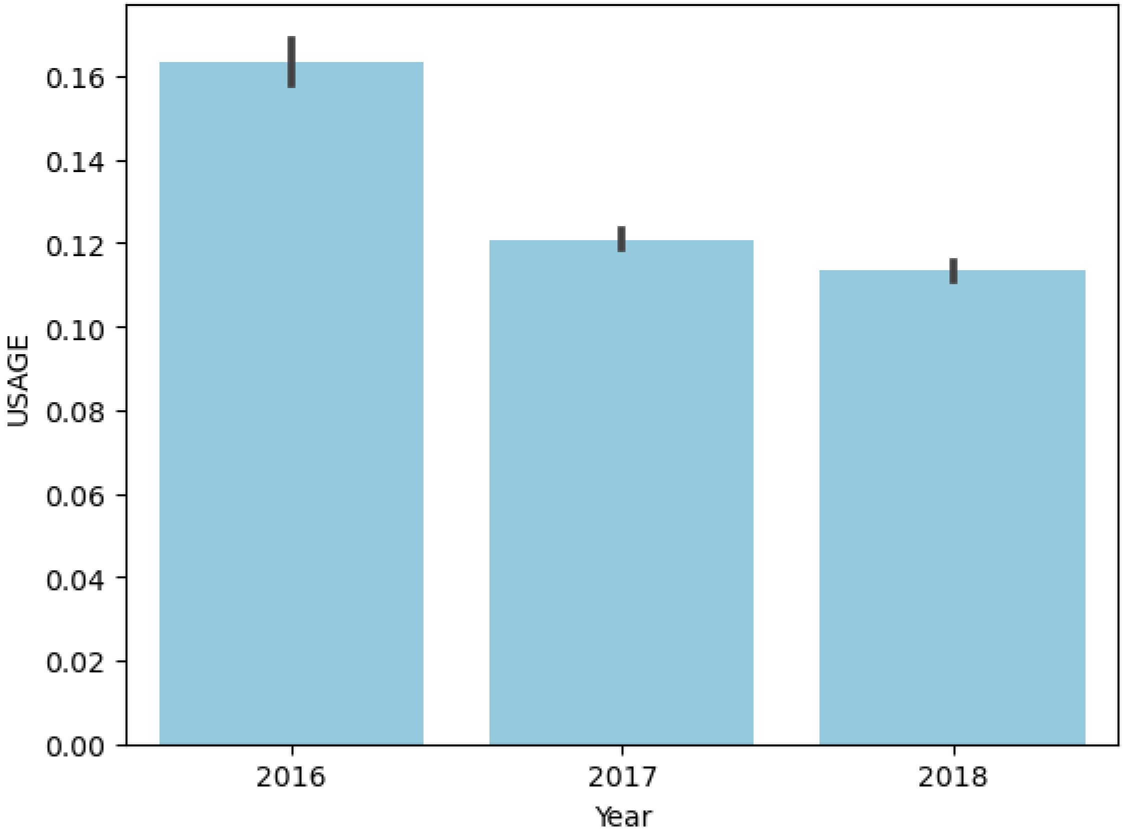
```
USAGE                     float64
UNITS                      object
COST($)                   float64
Month                       int64
Day                         int64
Year                        int64
Week_Day                    int64
dtype: object
```

In [25]:
```
df['Week_Day'].nunique()
```

Out[25]:
```
7
```

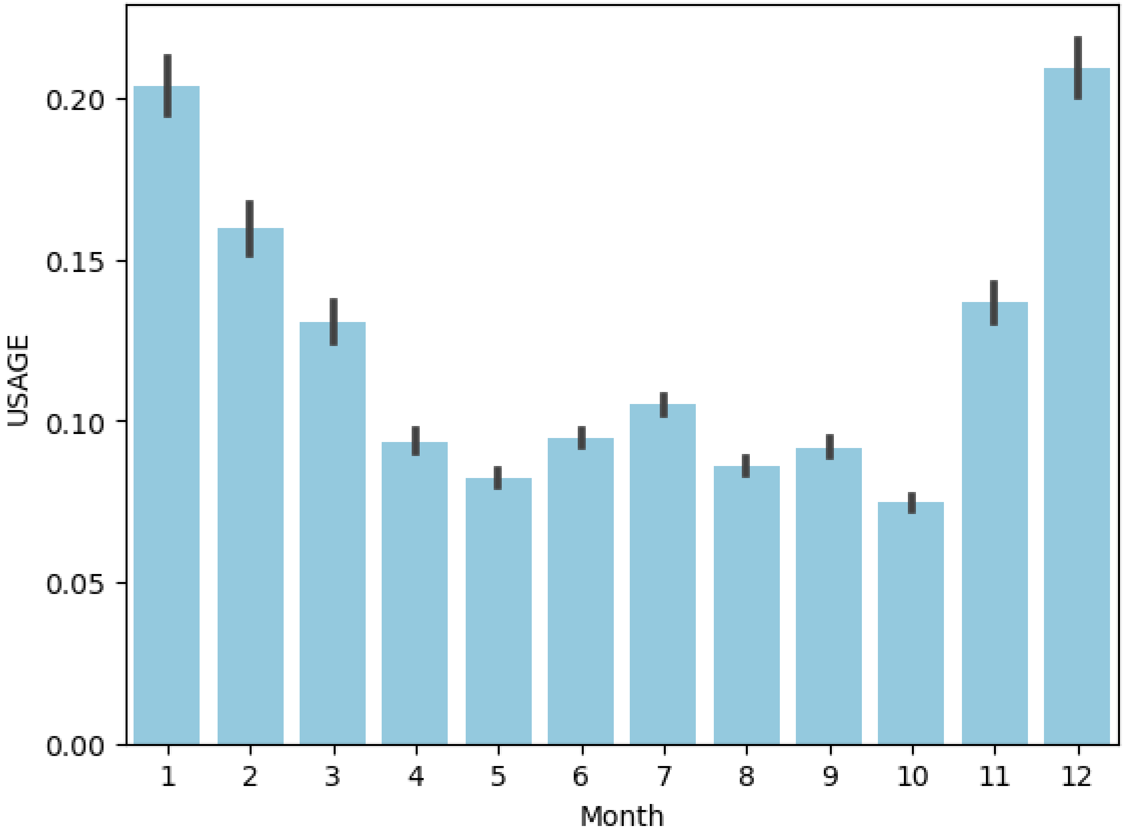
In [26]:
```
sns.barplot(x='Year', y='USAGE', data=df, color='skyblue')
```

Out[26]:
```
<AxesSubplot:xlabel='Year', ylabel='USAGE'>
```



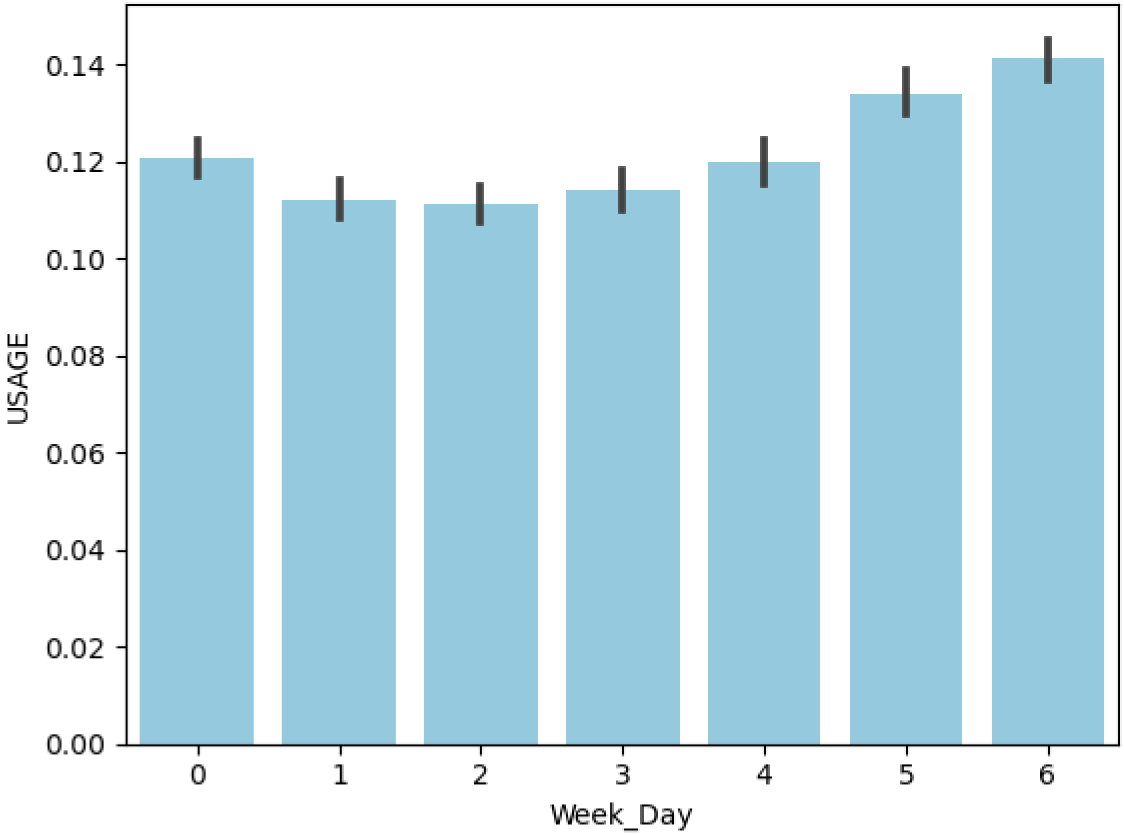
In [27]:
```
sns.barplot(x='Month', y='USAGE', data=df, color='skyblue')
```

Out[27]:
```
<AxesSubplot:xlabel='Month', ylabel='USAGE'>
```



```
In [28]: sns.barplot(x='Week_Day', y='USAGE', data=df, color='skyblue')

Out[28]: <AxesSubplot:xlabel='Week_Day', ylabel='USAGE'>
```

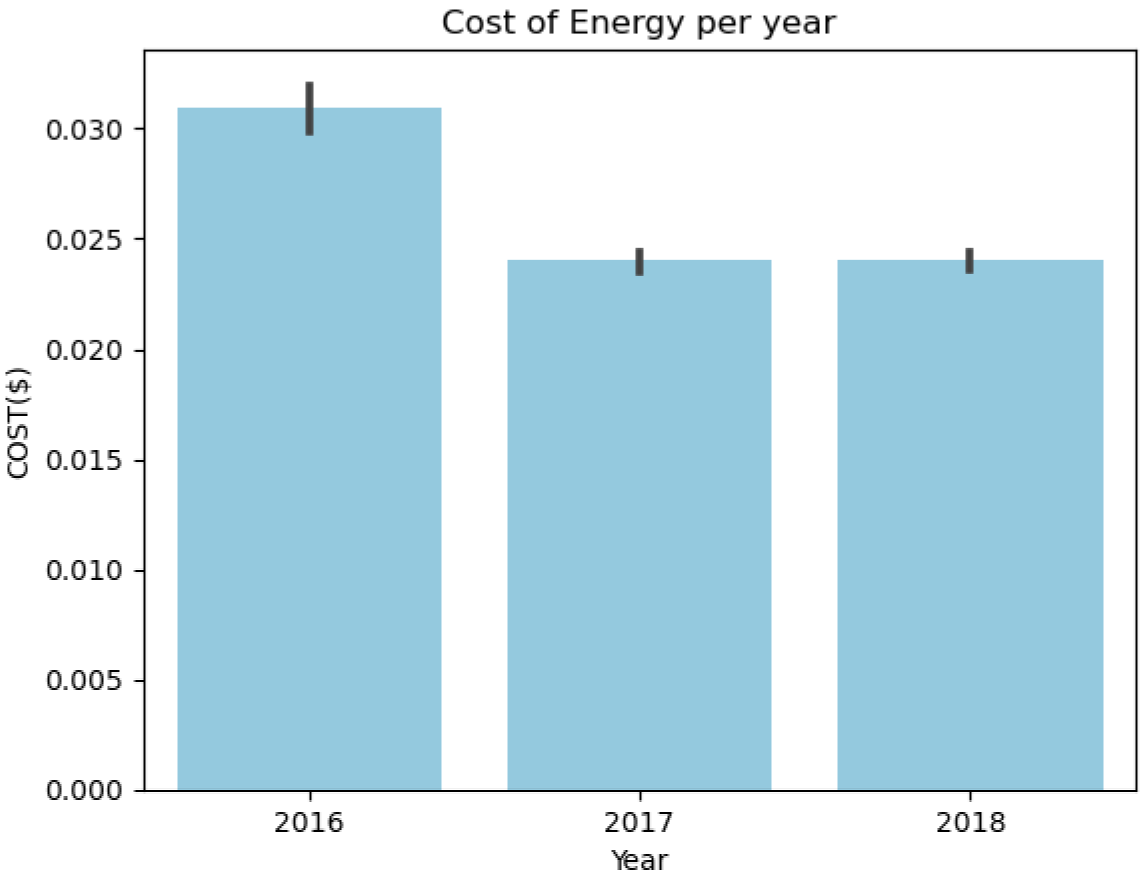


Keys: 0=Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday

Checking the cost of delivery energy w.r.t year, months and week days

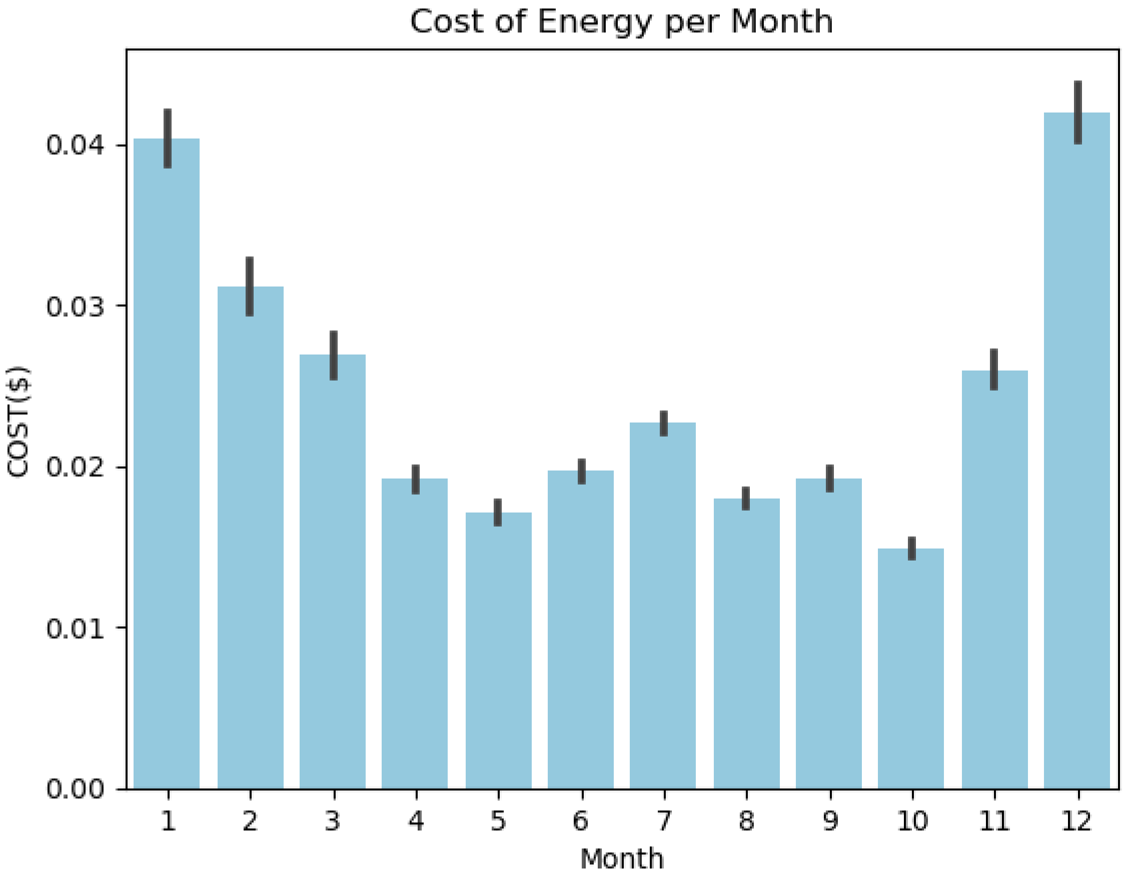
In [37]:
```
sns.barplot(x='Year', y='COST($)', data=df, color='skyblue').set(title='Cost of Energy p
```

Out[37]:
```
[Text(0.5, 1.0, 'Cost of Energy per year')]
```



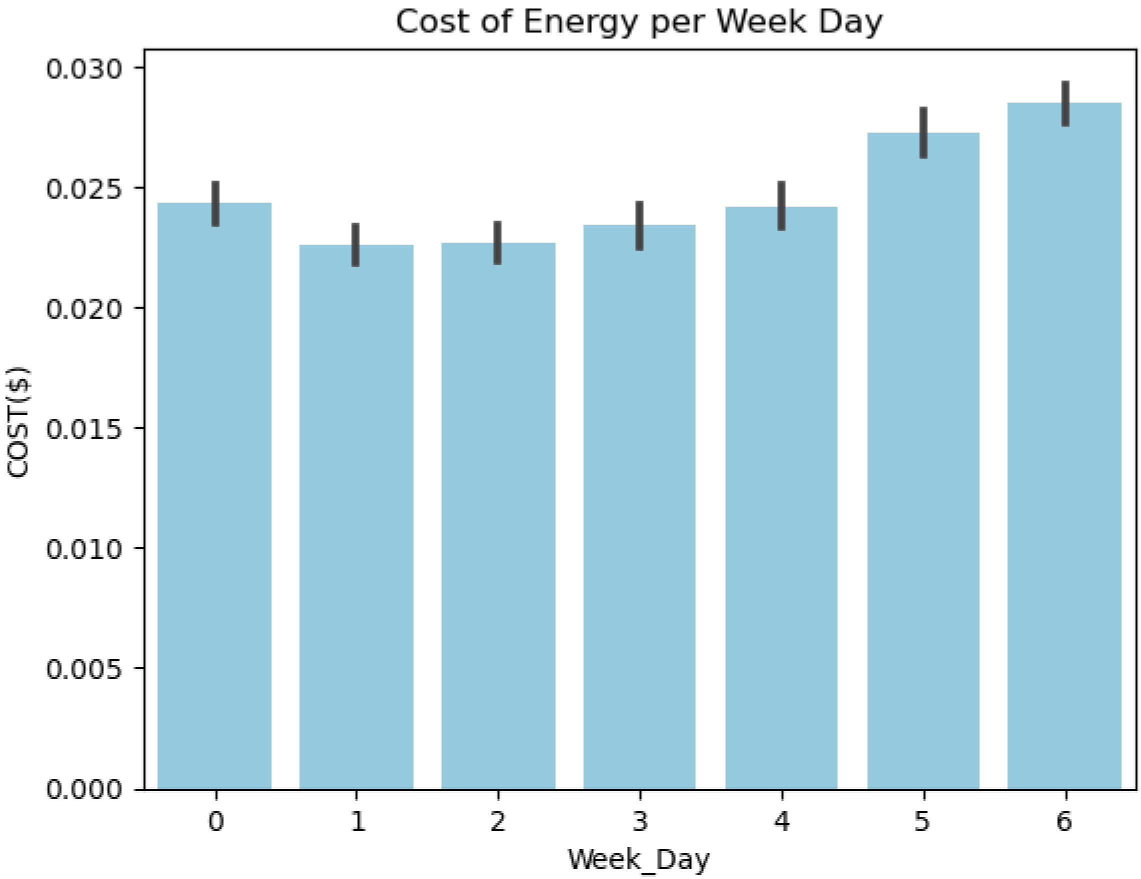
In [38]:
```
sns.barplot(x='Month', y='COST($)', data=df, color='skyblue').set(title='Cost of Energy
```

Out[38]:
```
[Text(0.5, 1.0, 'Cost of Energy per Month')]
```



In [39]:
```
sns.barplot(x='Week_Day', y='COST($)', data=df, color='skyblue').set(title='Cost of Ener
```

Out[39]:
```
[Text(0.5, 1.0, 'Cost of Energy per Week Day')]
```



In [ ]: