Time-Series Modeling

Patrick Herlihy

ID: 010440477

Date: 02/24/23

Data Preparation Steps

- Load the data into a Pandas dataframe
- Review the head of all the data, data types, and statistics of all numerical data
- Check for null values and duplicated data
- Visualize linegraph of the data
- Perform the Augmented Dickey Fuller (ADF) Test to check if the data is stationary
- If ADF test returns value > 0.05 data is non-stationary, take the difference of the data and check again
- Data is prepared, Split data so that firts 80% is a training set and remaining 20% is the test set

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import datetime
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import acf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.gridspec as gridspec
from statsmodels.tsa.arima.model import ARIMA
from pmdarima import auto_arima

from statsmodels.tsa.seasonal import STL
```

In [3]:
```
path='/Users/herlihpj/Desktop/Data Analytics/D213 - Advanced Data Analytics/Task 1/Medical/'
#Reads CSV to data frame, sets case order to index
```

```
med_rev= pd.read_csv(path+'medical_time_series.csv', index_col=0)
print(med_rev.head())
print()
print('Shape: ', med_rev.shape)
print()
print(med_rev.info())
print()
print(med_rev.describe())
```

```
      Revenue
Day
1    0.000000
2   -0.292356
3   -0.327772
4   -0.339987
5   -0.124888

Shape:  (731, 1)

<class 'pandas.core.frame.DataFrame'>
Int64Index: 731 entries, 1 to 731
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Revenue  731 non-null    float64
dtypes: float64(1)
memory usage: 11.4 KB
None

          Revenue
count  731.000000
mean    14.179608
std      6.959905
min     -4.423299
25%     11.121742
50%     15.951830
75%     19.293506
max     24.792249
```

In [4]:
```
#Check for Null
print('Summary of Null: ')
print(med_rev.isna().sum())
print()
#Check for duplicated data
duplicates=med_rev.duplicated()
```

```
print('Duplicates: ', duplicates.sum())
#No Null values or duplicates found
```

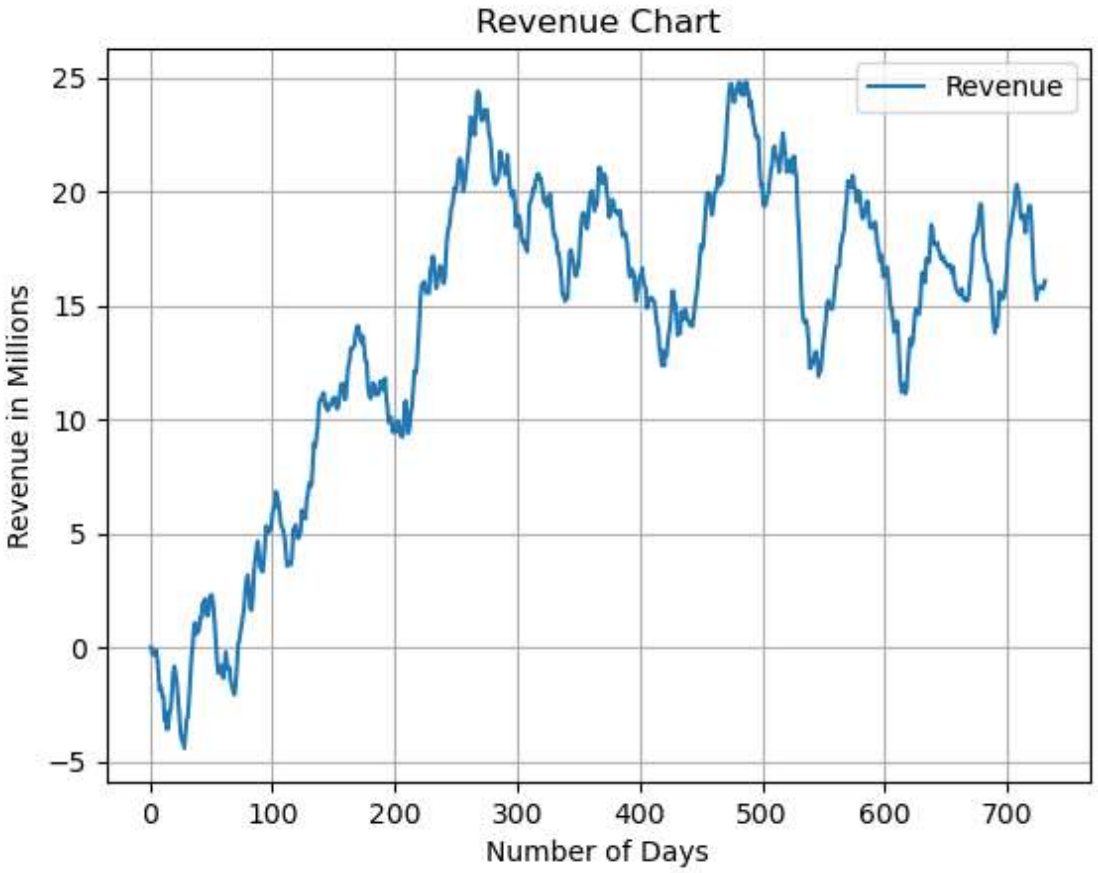
```
Summary of Null:
Revenue    0
dtype: int64

Duplicates:  0
```

In [5]:

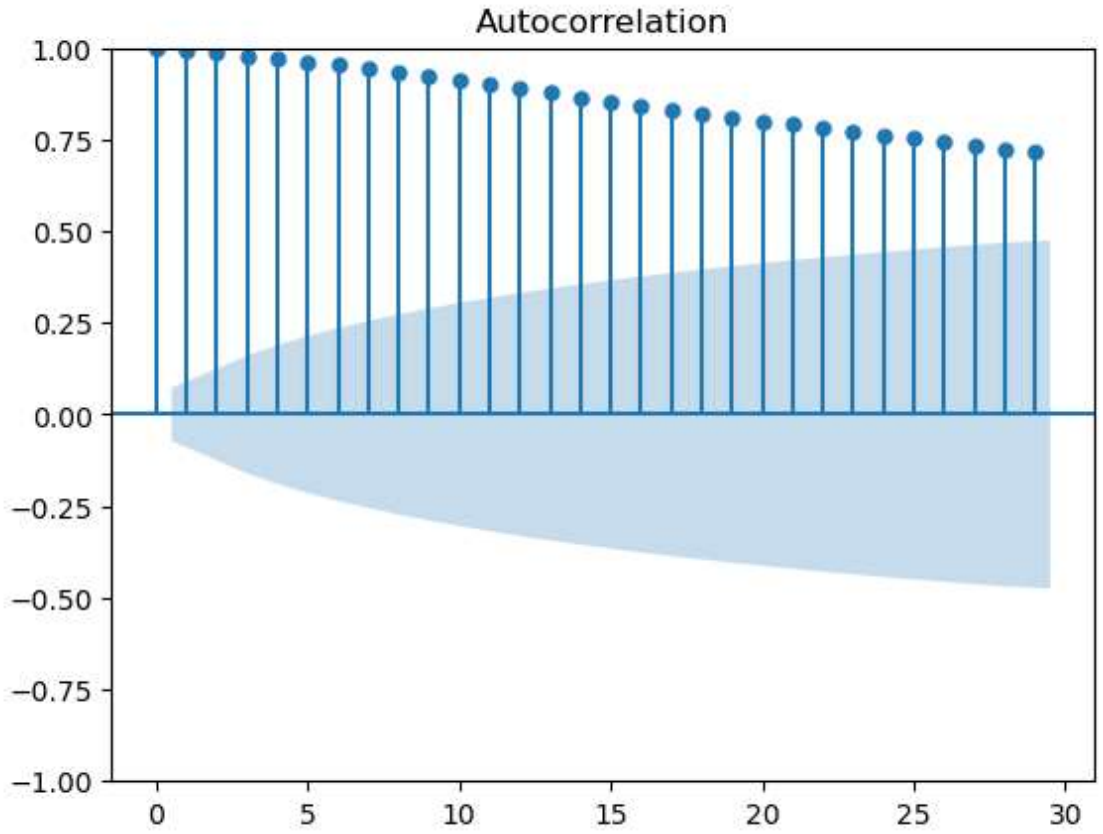
```
#Visualize the data
plt.figure(figsize=(10,5))
med_rev.plot()
#plt.plot(med_rev.Revenue)
plt.title('Revenue Chart')
plt.xlabel('Number of Days')
plt.ylabel('Revenue in Millions')
plt.grid(True)
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```



In [6]:
```
#ADF Dickey Fuller test: Original Dataset
result=adfuller(med_rev['Revenue'])
print('Test Statistics: ',result[0])
print('P-Value: ', result[1])
print('Lags: ', result[2])
print('Observations: ', result[3])
print('Critical Values: ', result[4])
plot_acf(med_rev['Revenue'])
plt.show()
```

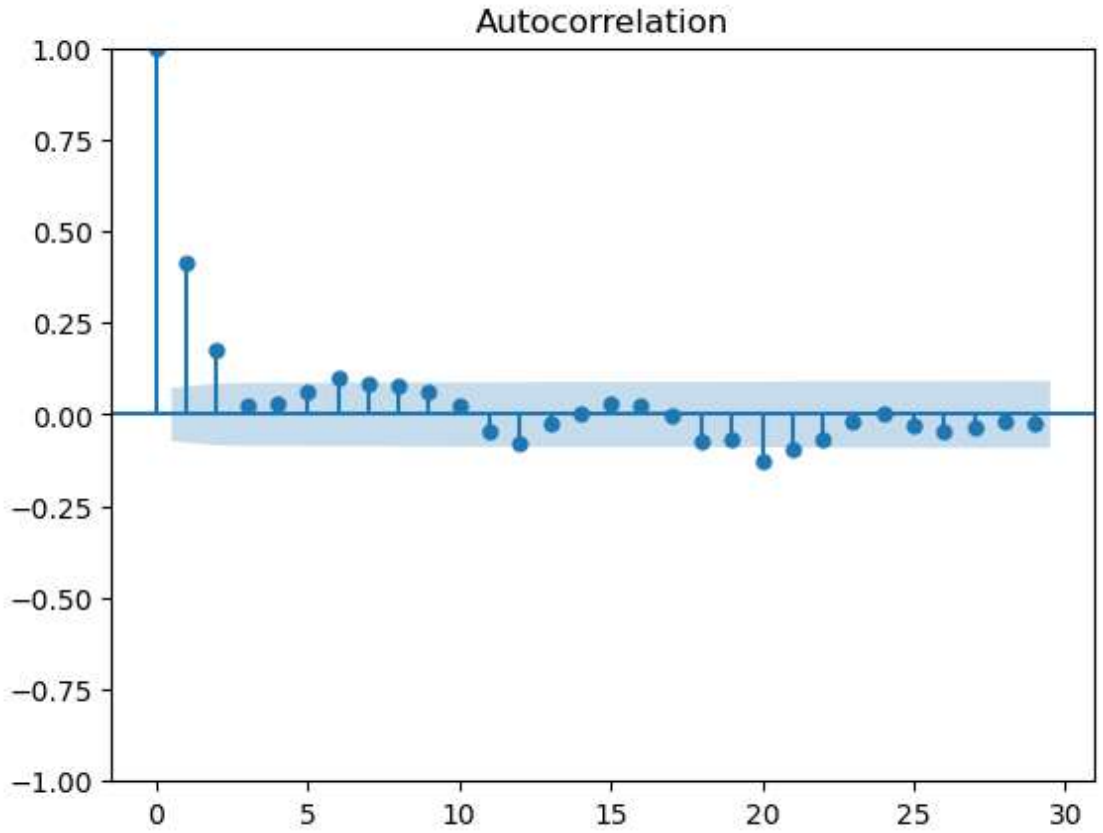
```
Test Statistics:  -2.218319047608946
P-Value:  0.19966400615064328
Lags:  1
Observations:  729
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
```



In [7]:
```
med_rev['differences'] =med_rev['Revenue'].diff()
#drop first value because it will be nan
med_rev = med_rev.dropna()
result2=adfuller(med_rev['differences'])
print('After Taking Differences')
print('Test Statistics: ',result2[0])
print('P-Value: ', result2[1])
print('Lags: ', result2[2])
print('Observations: ', result2[3])
print('Critical Values: ', result2[4])
```

```
After Taking Differences
Test Statistics:  -17.374772303557062
P-Value:  5.113206978840171e-30
Lags:  0
Observations:  729
Critical Values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.5688855736949163}
```

In [8]:
```
#check the ACF plot of the differences
plot_acf(med_rev['differences'])
plt.show()
```



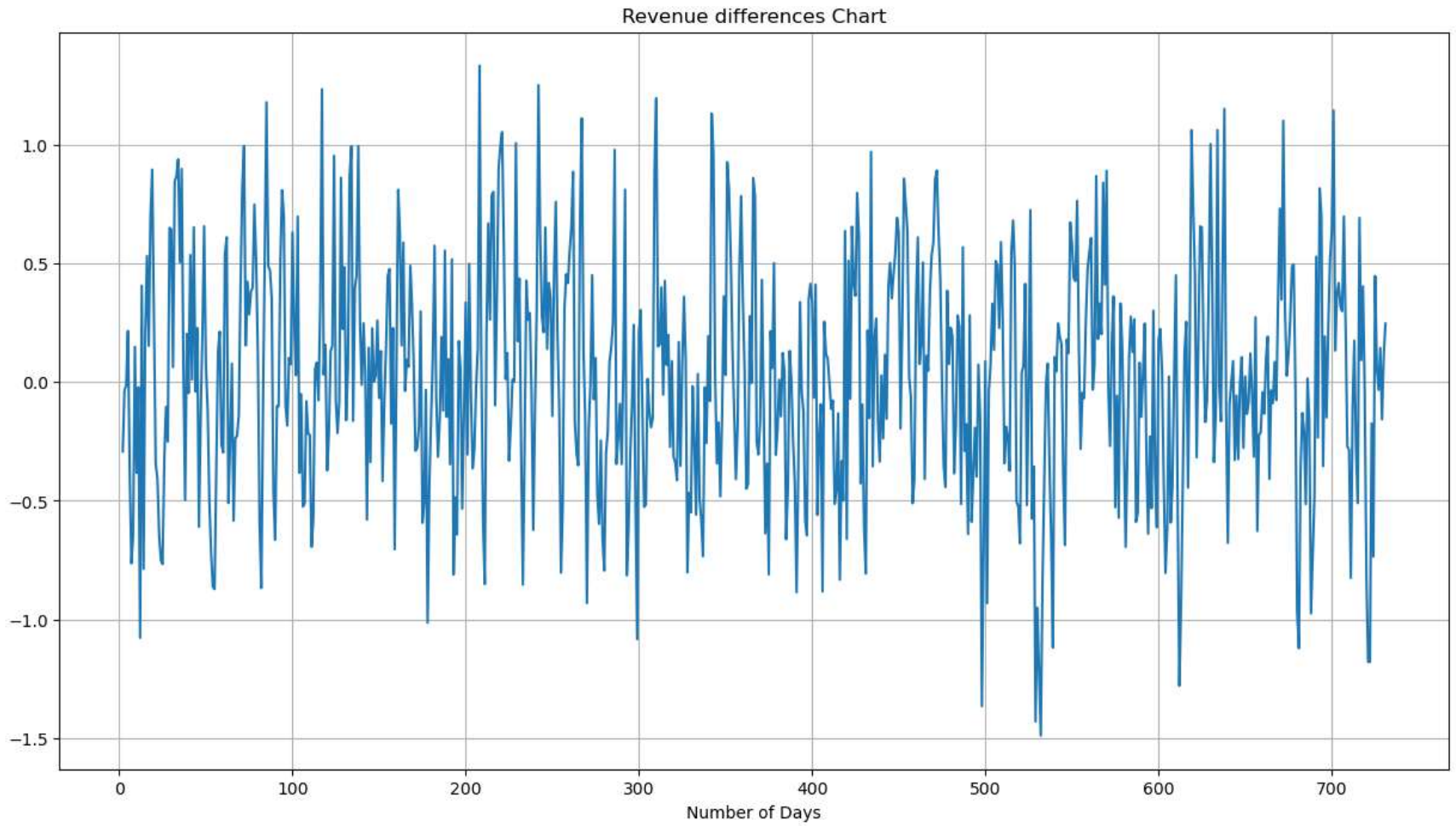
After taking the difference the data is now stationary

- P-Value < 0.05
- Autocorrelation Rapidly drops within 0.05

Visualize Stationarity of the Data:

In [25]:
```
#Visualize the differences data
plt.figure(figsize=(15,8))
plt.plot(med_rev.differences)
```

```
plt.title('Revenue differences Chart')
plt.xlabel('Number of Days')
plt.grid(True)
plt.show()
```



In [9]:
```
#train test split the data 70% train, 30% Test
train_percent=0.8
train_len=int(med_rev.shape[0]*train_percent)
train_length_str=str(train_len)
x_train=med_rev.loc[:train_length_str]
x_test=med_rev.loc[train_length_str:]
print('Length of training set', x_train.shape)
print('Length of test set', x_test.shape)
```

```
Length of training set (583, 2)
Length of test set (148, 2)
```

In [10]:
```
x_train.to_csv(path+'Training Set.csv')
x_test.to_csv(path+'Testing Set.csv')
med_rev.to_csv(path+'Full Prepared Set.csv')
```

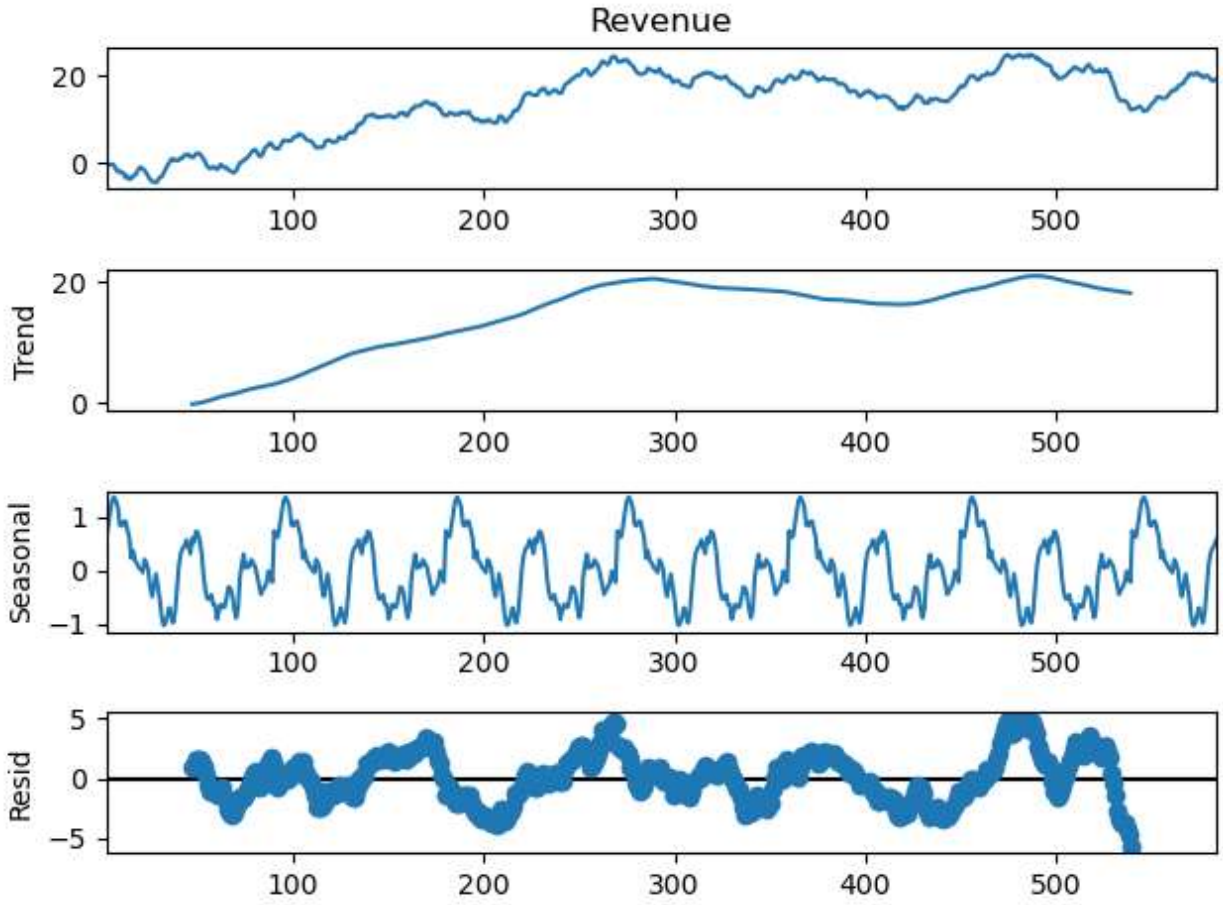
Data Analysis

Decompose the dataset to visualize

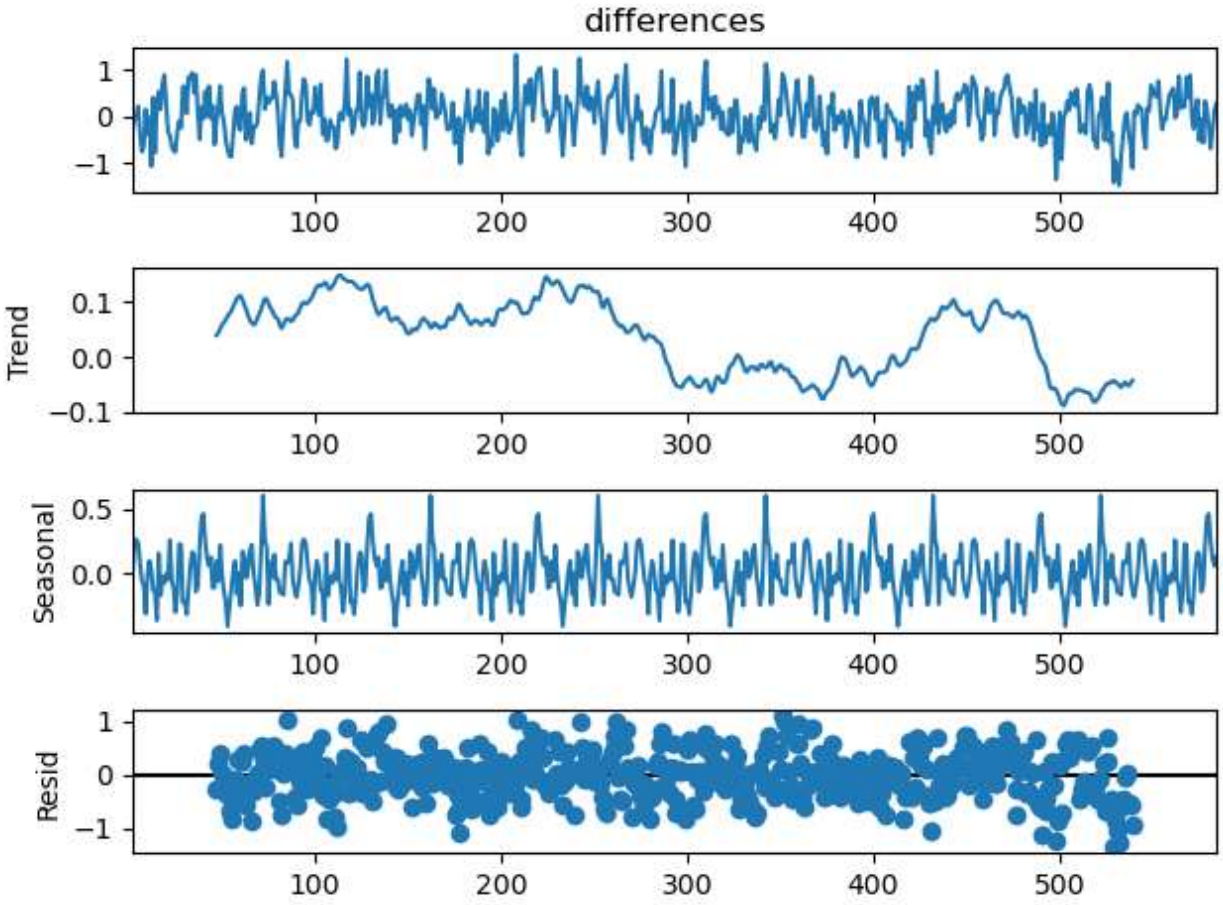
- Trends
- Seasonality
- Lack of trends in Residuals

In [13]:
```
decomp=seasonal_decompose(x_train['Revenue'], model='additive', period=90)
#decomp.seasonal.plot()
print(decomp.plot())
plt.show()
print('After Taking the Difference')
decomp2=seasonal_decompose(x_train['differences'], model='additive', period=90)
print(decomp2.plot())
plt.show()
```

```
Figure(640x480)
```

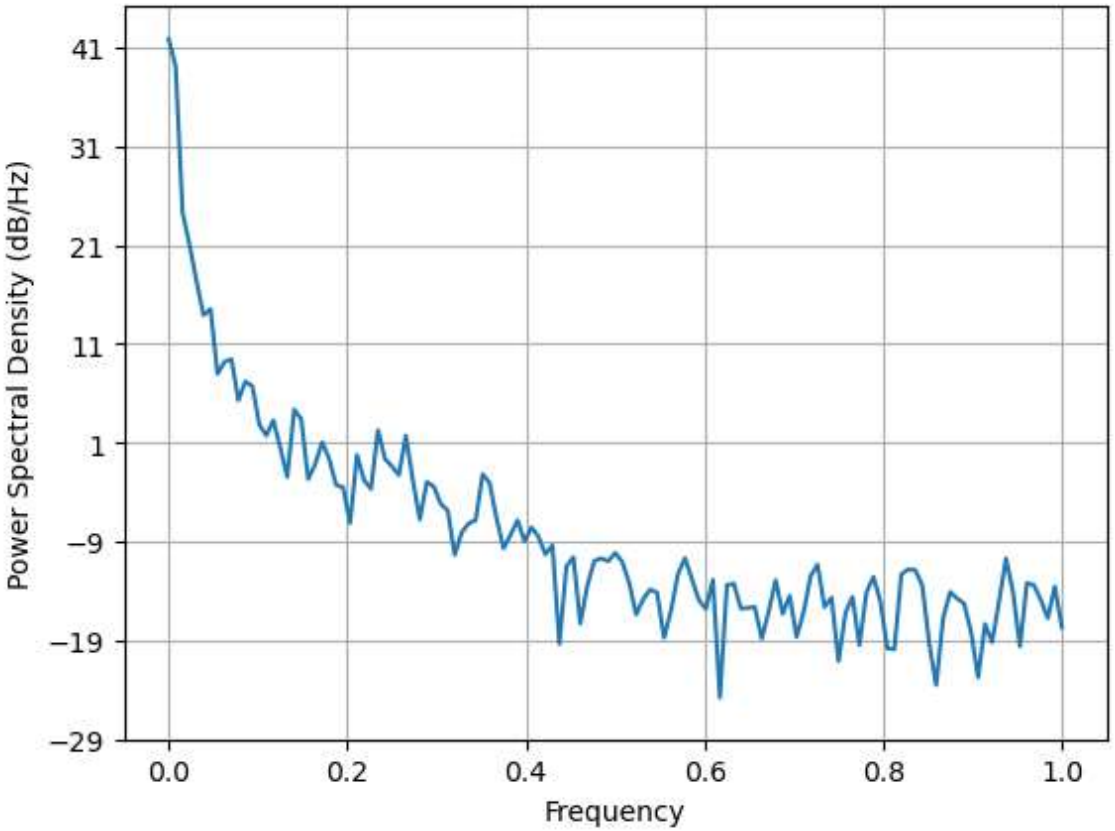



```
After Taking the Difference
Figure(640x480)
```



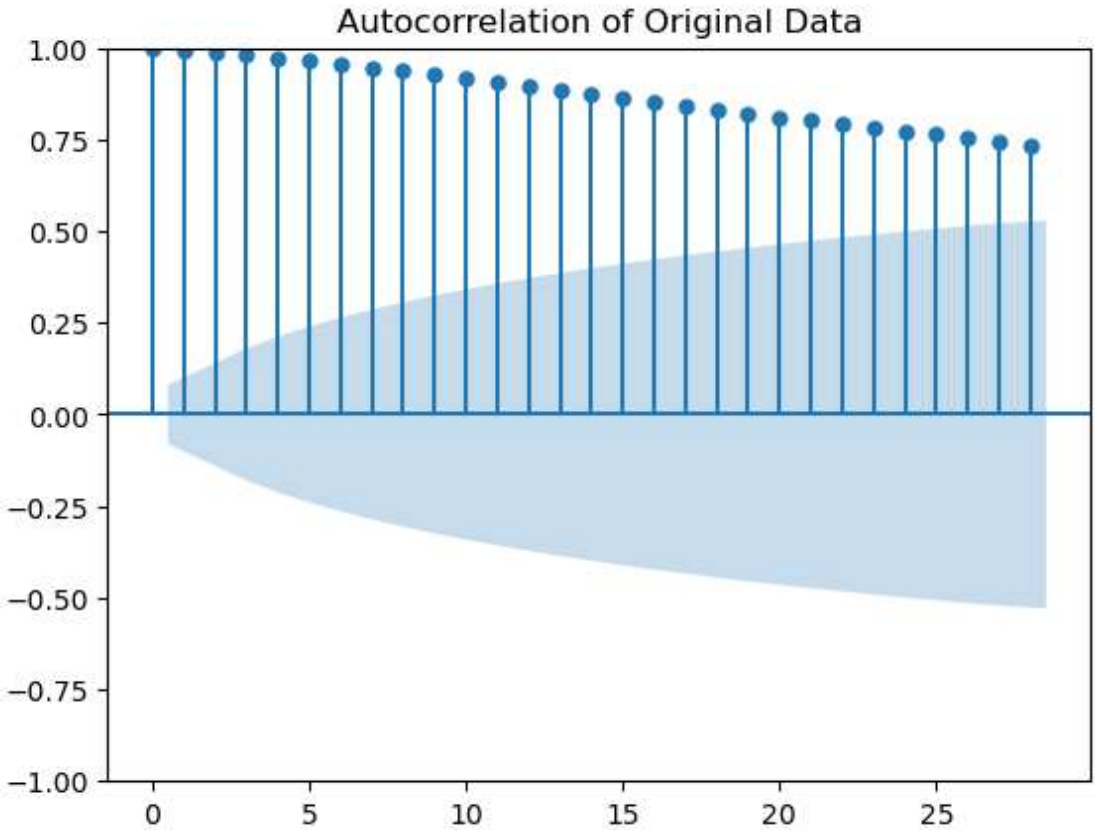
Spectral Density

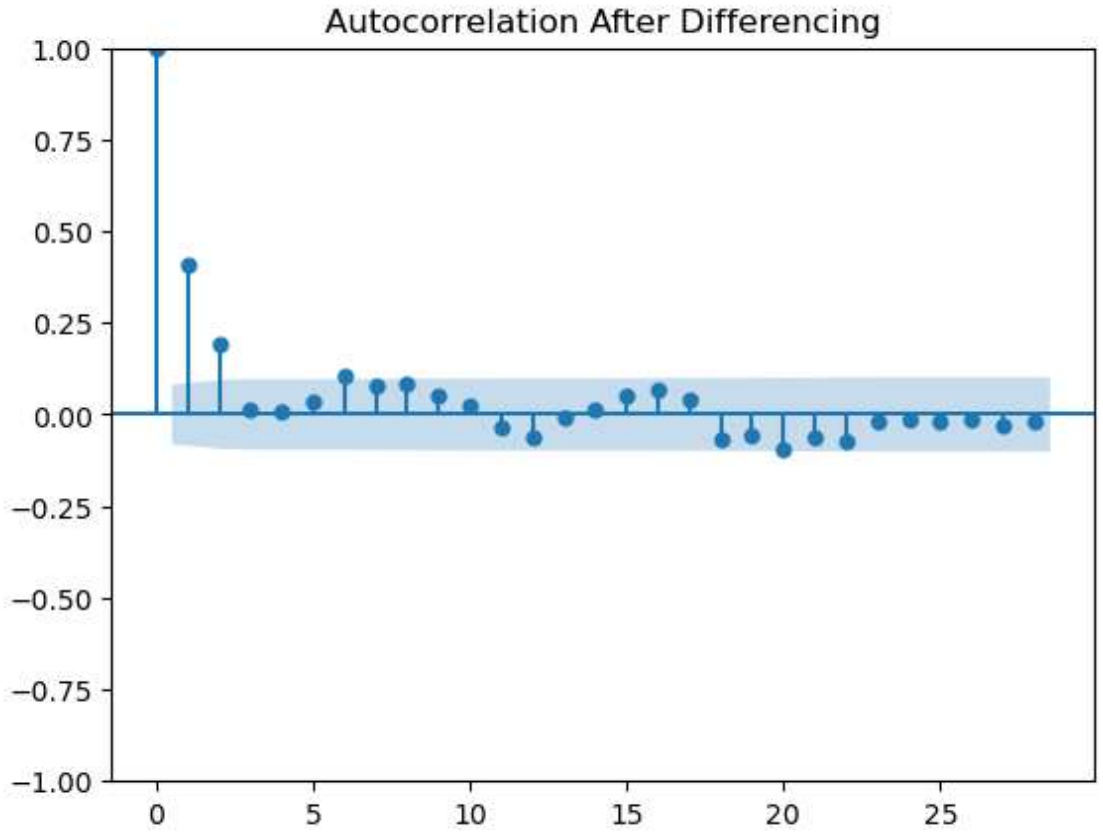
In [29]:
```
plt.psd(x_train['Revenue'])
plt.show()
```



Autocorrelation for both stationary and the non-stationary data:

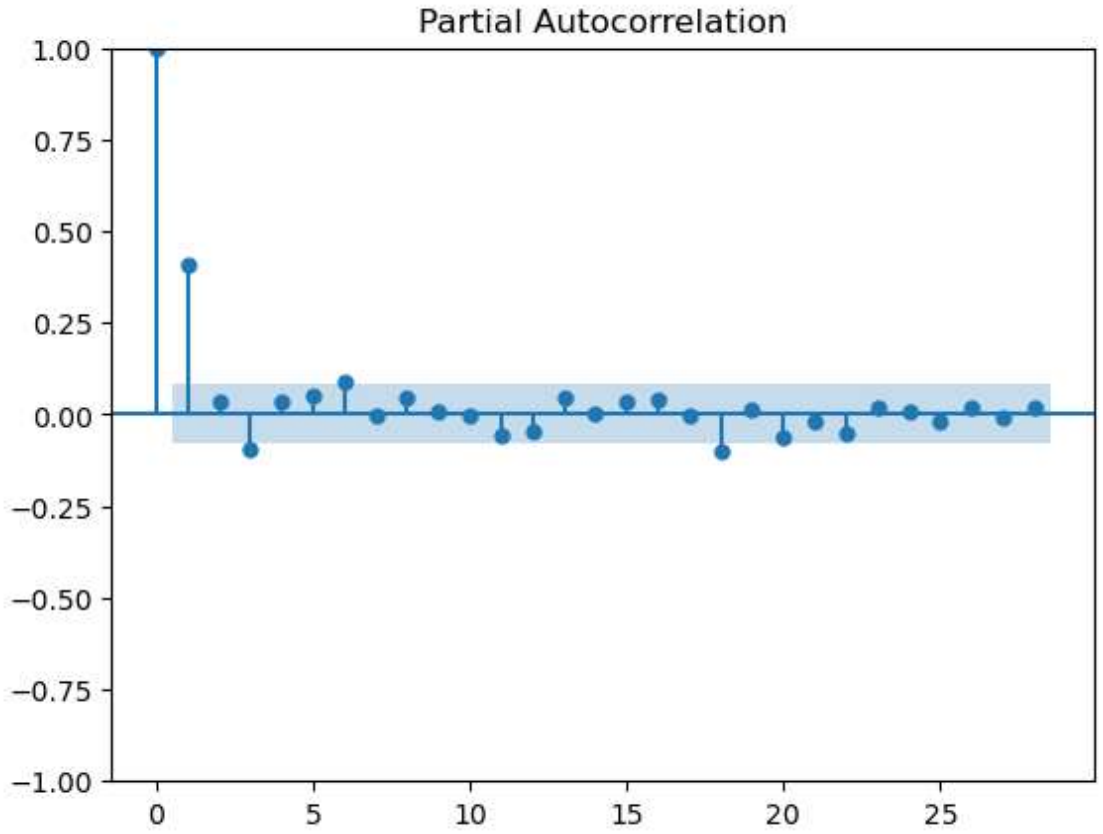
In [30]:
```
#Plot ACF of the original set
plot_acf(x_train['Revenue'])
plt.title('Autocorrelation of Original Data')
plt.show()
#check the ACF plot of the differences
plot_acf(x_train['differences'])
plt.title('Autocorrelation After Differencing')
plt.show()
```


Autocorrelation of Original Data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25



In [15]:
```
plot_pacf(x_train['differences'])
plt.show()
```

```
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default method 'y
w' can produce PACF values outside of the [-1,1] interval. After 0.13, the default will change tounadjusted Yule-Walker
('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```



Begin Model Construction

In [42]:
```
#AR Model
#model=ARIMA(x_train['Revenue'], order=(3,1,2))

#print(results_AR.summary()),error_action='ignore',suppress_warnings=True
from pmdarima import auto_arima
best_arima_model = auto_arima(x_train['Revenue'], start_p=1,d=1, start_q=1,max_p=3, max_q=3, m=30,
                              start_P=0, seasonal=True, D=1, trace=True,stepwise=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[30]             : AIC=inf, Time=6.92 sec
 ARIMA(0,1,0)(0,1,0)[30]             : AIC=1163.401, Time=0.22 sec
 ARIMA(1,1,0)(1,1,0)[30]             : AIC=888.723, Time=1.43 sec
 ARIMA(0,1,1)(0,1,1)[30]             : AIC=inf, Time=4.43 sec
 ARIMA(1,1,0)(0,1,0)[30]             : AIC=1046.641, Time=0.29 sec
 ARIMA(1,1,0)(2,1,0)[30]             : AIC=836.470, Time=4.60 sec
 ARIMA(1,1,0)(2,1,1)[30]             : AIC=inf, Time=21.72 sec
 ARIMA(1,1,0)(1,1,1)[30]             : AIC=inf, Time=6.23 sec
 ARIMA(0,1,0)(2,1,0)[30]             : AIC=934.347, Time=2.22 sec
 ARIMA(2,1,0)(2,1,0)[30]             : AIC=837.828, Time=4.82 sec
 ARIMA(1,1,1)(2,1,0)[30]             : AIC=837.954, Time=7.30 sec
 ARIMA(0,1,1)(2,1,0)[30]             : AIC=856.556, Time=4.08 sec
 ARIMA(2,1,1)(2,1,0)[30]             : AIC=838.081, Time=13.01 sec
 ARIMA(1,1,0)(2,1,0)[30] intercept   : AIC=838.469, Time=10.14 sec

Best model:  ARIMA(1,1,0)(2,1,0)[30]
Total fit time: 87.433 seconds
```

Auto ARIMA determined the optimal Model ARIMA(1, 1, 0) (2, 1, 0) [30]

Construct the training model

In [84]:
```
#arima forcast
# SARIMAX(1, 1, 0)x(2, 1, 0)[30]
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = SARIMAX(x_train['Revenue'], order=(1,1,0), seasonal_order = (2,1,0,30))
test_model = model.fit()
test_model.summary()
```

```
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: An unsupported index
was provided and will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: An unsupported index
was provided and will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

Out[84]:

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	583
Model:	SARIMAX(1, 1, 0)x(2, 1, 0, 30)	**Log Likelihood**	-414.235
Date:	Thu, 23 Feb 2023	**AIC**	836.470
Time:	22:15:00	**BIC**	853.724
Sample:	0	**HQIC**	843.212
	- 583		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.4096	0.039	10.633	0.000	0.334	0.485
ar.S.L30	-0.6848	0.043	-16.064	0.000	-0.768	-0.601
ar.S.L60	-0.3312	0.047	-7.088	0.000	-0.423	-0.240
sigma2	0.2549	0.016	16.014	0.000	0.224	0.286

Ljung-Box (L1) (Q):	0.11	**Jarque-Bera (JB):**	0.70
Prob(Q):	0.74	**Prob(JB):**	0.71
Heteroskedasticity (H):	1.05	**Skew:**	0.06
Prob(H) (two-sided):	0.72	**Kurtosis:**	2.87

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [100…

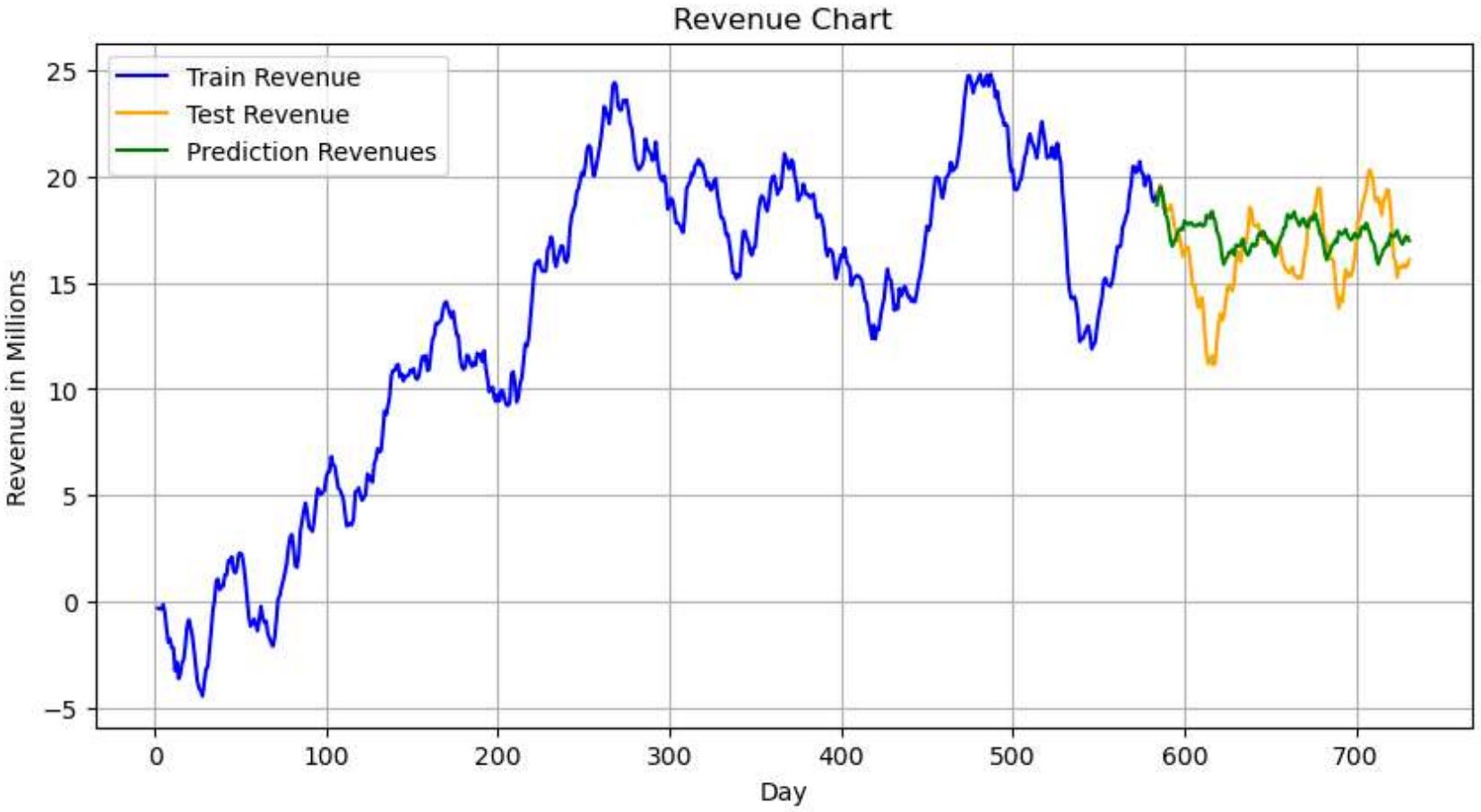
```
#Generate predictions
start=len(x_train)-1
end=start+len(x_test)-1
predictions=test_model.predict(start=start,end=end, typ='levels').rename('Prediction Revenues')
predictions.index=med_rev.index[start:end+1]
print(predictions)
```

```
Day
584    18.648989
585    19.149971
586    19.496403
587    19.253494
588    18.768538
         ...
727    16.799083
728    16.932229
729    17.169005
730    17.154513
731    16.969389
Name: Prediction Revenues, Length: 148, dtype: float64
```

```
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:834: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
```

In [104…

```
#Final visualizaion of the data
plt.figure(figsize=(10,5))
plt.plot(x_train.Revenue,c='blue',label='Train Revenue')
plt.plot(x_test.Revenue,c='orange', label='Test Revenue')
predictions.plot(c='green')
plt.title('Revenue Chart')
plt.xlabel('Day')
plt.ylabel('Revenue in Millions')
plt.legend(loc="upper left")
plt.grid(True)
plt.show()
```



In [61]:
```
x_train['Revenue'].mean()
```

Out[61]:
```
13.63537734359863
```

Evaluate model using RMSE

In [64]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
print(x_test.tail())
print(predictions.tail())
rmse=sqrt(mean_squared_error(predictions,x_test['Revenue']))
print('Root Mean Squared Error:', rmse)
```

```
       Revenue  differences
Day
727  15.722056    -0.032693
728  15.865822     0.143766
729  15.708988    -0.156834
730  15.822867     0.113880
731  16.069429     0.246562
Day
727    16.799083
728    16.932229
729    17.169005
730    17.154513
731    16.969389
Name: Future Revenues, dtype: float64
Root Mean Squared Error: 2.2984803062592616
```

An RMSE of 2.3 with a Mean of 13.64 is fairly accurate

Construct final Model and Forecast the next 60 days.

In [65]:
```
#error is good, fit entire model, predict 30 days into future
#arima forcast
# SARIMAX(1, 1, 0)x(2, 1, 0)[30]
model = SARIMAX(med_rev['Revenue'], order=(1,1,0), seasonal_order = (2,1,0,30))
final_model = model.fit()
final_model.summary()
```

```
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: An unsupported index
was provided and will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWarning: An unsupported index
was provided and will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

Out[65]:

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	730
Model:	SARIMAX(1, 1, 0)x(2, 1, 0, 30)	**Log Likelihood**	-523.754
Date:	Thu, 23 Feb 2023	**AIC**	1055.508
Time:	09:03:08	**BIC**	1073.707
Sample:	0	**HQIC**	1062.543
	- 730		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.4256	0.034	12.552	0.000	0.359	0.492
ar.S.L30	-0.6880	0.037	-18.595	0.000	-0.761	-0.616
ar.S.L60	-0.3011	0.039	-7.693	0.000	-0.378	-0.224
sigma2	0.2562	0.014	17.823	0.000	0.228	0.284

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	1.41
Prob(Q):	0.92	**Prob(JB):**	0.49
Heteroskedasticity (H):	1.12	**Skew:**	0.07
Prob(H) (two-sided):	0.39	**Kurtosis:**	2.82

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [108…

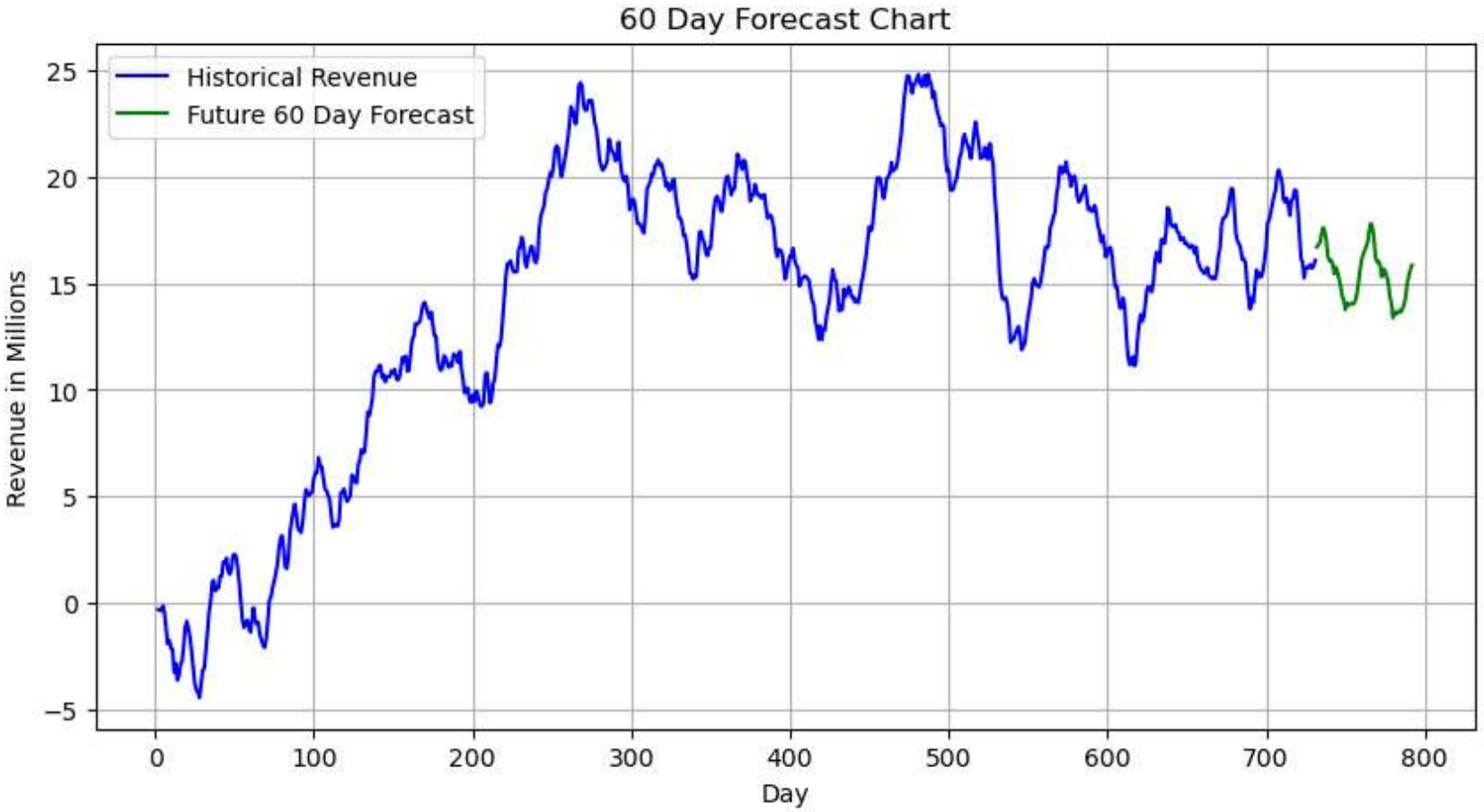
```
#Generate predictions
days_to_predict=60
start=len(med_rev)+2
end=start+days_to_predict
forecasts=final_model.predict(start=start,end=end, typ='levels').rename('Future 60 Day Forecast')
print(forecasts)
```

```
732    16.693084
733    16.812677
734    16.907043
735    17.308930
736    17.603860
         ...
788    14.383012
789    15.017770
790    15.358732
791    15.653509
792    15.844145
Name: Future 60 Day Forecast, Length: 61, dtype: float64
```

```
C:\Users\herlihpj\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:834: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
```

In [109…

```
#Final visualizaion of the data
plt.figure(figsize=(10,5))
plt.plot(med_rev.Revenue,c='blue',label='Historical Revenue')
forecasts.plot(c='green')
plt.title('60 Day Forecast Chart')
plt.xlabel('Day')
plt.ylabel('Revenue in Millions')
plt.legend(loc="upper left")
plt.grid(True)
plt.show()
```



Recomendations

As we can see from the above graph revenue is projected slowly downward with some seasonality over the next 60 days. We should look into what features are causing a decrease in revenue and mitigate the issues. We should also look into what new additions we can make to the hospital in order to get the revenues into an upward trajectory.