Part I: Research Question

A1

Does revenue have any recurring seasonal patterns?

A2

The goal of this analysis is to see if revenue follows a seasonal pattern that could help predict future revenue.

Part II: Method Justification

B

Time series models have assumptions that include stationarity and autocorrelated. The assumption of stationarity is the data follows a consistent behavior over time. The assumption of autocorrelated is that data points are correlated with data points from the past.

Part III: Data Preparation

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from dateutil.parser import parse
import warnings
warnings.filterwarnings('ignore')
import os
from datetime import datetime
from statsmodels.tsa.seasonal import seasonal_decompose
import statsmodels.api
```

In [7]:
```
# Grabbing data from the csv file
ts = pd.read_csv("C:/Users/cklni/Desktop/WGU/D213/Task 1/teleco_time_series .csv", index
```

In [8]:
```
ts.info()
```

```
<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
```

In [9]:
```
ts.describe()
```

Out[9]:

	Revenue

count	731.000000
mean	9.822901
std	3.852645
min	0.000000
25%	6.872836
50%	10.785571
75%	12.566911
max	18.154769

In [5]:
```
ts['Date'] = (pd.date_range(start=datetime(2022, 1, 1),
                            periods=ts.shape[0], freq='24H'))

ts.set_index('Date' ,inplace=True)
ts
```

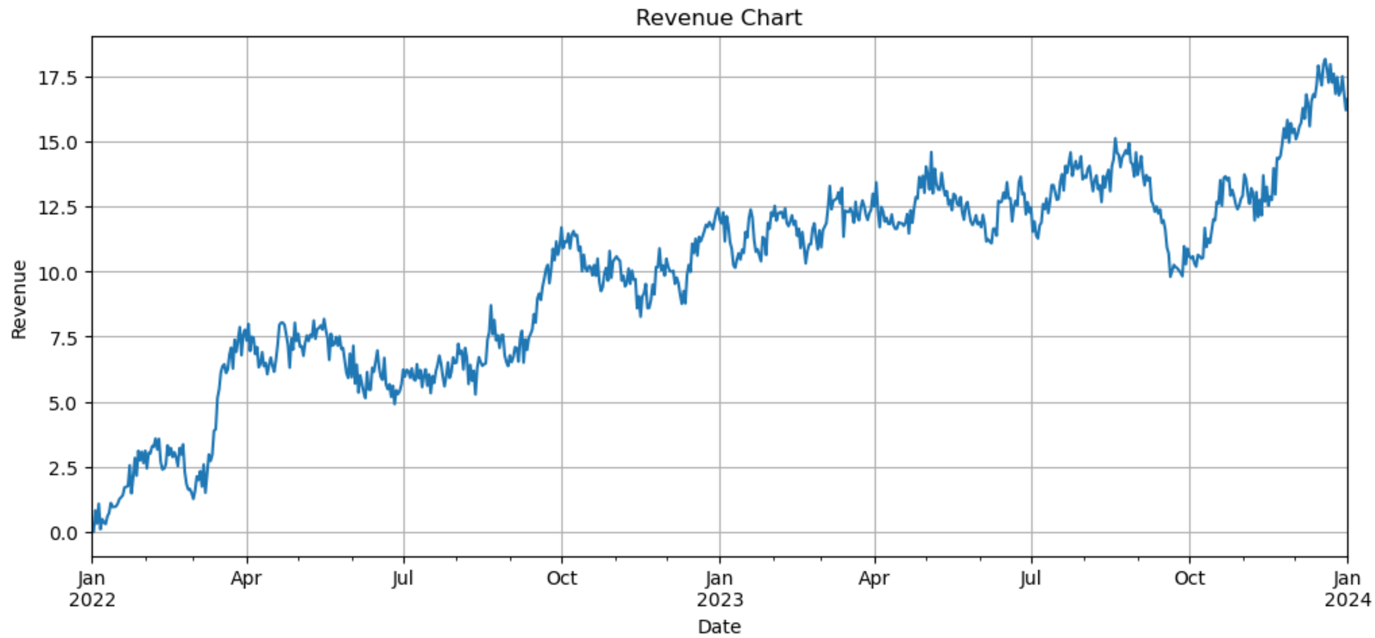
Out[5]:

	Revenue
Date	
2022-01-01	0.000000
2022-01-02	0.000793
2022-01-03	0.825542
2022-01-04	0.320332
2022-01-05	1.082554
...	...
2023-12-28	16.931559
2023-12-29	17.490666
2023-12-30	16.803638
2023-12-31	16.194813
2024-01-01	16.620798

731 rows × 1 columns

C1

In [6]:
```
#Time series graph
ts['Revenue'].plot(figsize=(12,5))
plt.title('Revenue Chart')
plt.xlabel('Date')
plt.ylabel('Revenue')
plt.grid(True)
plt.show()
```



In [7]:
```
ts.isnull().any()
```

Out[7]:
```
Revenue    False
dtype: bool
```

In [8]:
```
ts.isna().sum()
```

Out[8]:
```
Revenue    0
dtype: int64
```

C2

In preparing the data I had to format the "Day" into "Date" as it is necessary to have a format such as year-month-date to be able to see seasonal trends instead of having a count of date that does not give information on the time of year. No gaps are present in the above graph. The length of the sequence is from January 1st 2022 to January 1st 2024 with a total of 731 days.

C3

In [9]:
```
#Testing stationarity mathematically
from statsmodels.tsa.stattools import adfuller
def ad_test(dataset):
    dftest = adfuller(dataset, autolag = 'AIC')
    print("1. ADF: ", dftest[0])
    print("2. P-value: ", dftest[1])
    print("3. Num of Lags:", dftest[2])
    print("4. Num of Obs Used for ADF Regression and Critical Values Calc:", dftest[3])
    print("5. Critical Values: ")
    for key, val in dftest[4].items():
        print("\t",key,": ",val)

ad_test(ts['Revenue'])
```

```
1. ADF:  -1.924612157310183
2. P-value:  0.3205728150793967
3. Num of Lags: 1
4. Num of Obs Used for ADF Regression and Critical Values Calc: 729
5. Critical Values:
	 1% :  -3.4393520240470554
```

```
        5% :  -2.8655128165959236
        10% :  -2.5688855736949163
```

The P-value is not less than 0.05, so the dataset does not have stationarity. I will use differencing to help stabilize the mean.

In [10]:
```
ts_diff = ts.diff().dropna()
ts_diff
```

Out[10]:

Date	Revenue
2022-01-02	0.000793
2022-01-03	0.824749
2022-01-04	-0.505210
2022-01-05	0.762222
2022-01-06	-0.974900
...	...
2023-12-28	0.170280
2023-12-29	0.559108
2023-12-30	-0.687028
2023-12-31	-0.608824
2024-01-01	0.425985

730 rows × 1 columns

In [11]:
```
#Stationarity test after differencing
result = adfuller(ts_diff['Revenue'])
print("Test statistics: ", result[0])
print("p-value: ", result[1])
print("critical values: ", result[4])

if result[1]<= 0.05:
    print("Reject null hypothesis, the time series is stationary")
else:
    print("Fail to reject null hypothesis, the time series is non-stationary")
```

```
Test statistics:  -44.87452719387599
p-value:  0.0
critical values:  {'1%': -3.4393520240470554, '5%': -2.8655128165959236, '10%': -2.56888
55736949163}
Reject null hypothesis, the time series is stationary
```

The dataset now have a p-value less than the 0.05 critical value. The dataset is now stationary.

C4

Using pandas I imported the teleco time series csv file with Day being my index column. I used info and describe to look at the data's characteristics and check for nulls. I changed the day column into Dates. Then I checked if the data has stationarity. It did not, so I used the differencing function. I checked for stationarity again and this time the results came back that it is stationary. I will export this clean data in a csv file as it is

a requirement in C5. I will split the data into training and test sets with the last 30 days for the test and export each of these into a csv file as required.

In [14]:
```
print(ts_diff.shape)
x_train = ts_diff.iloc[:-30]
x_test = ts_diff.iloc[-30:]

print('x_train shape', x_train.shape)
print('x_test shape', x_test.shape)
```

```
(730, 1)
x_train shape (700, 1)
x_test shape (30, 1)
```

C5

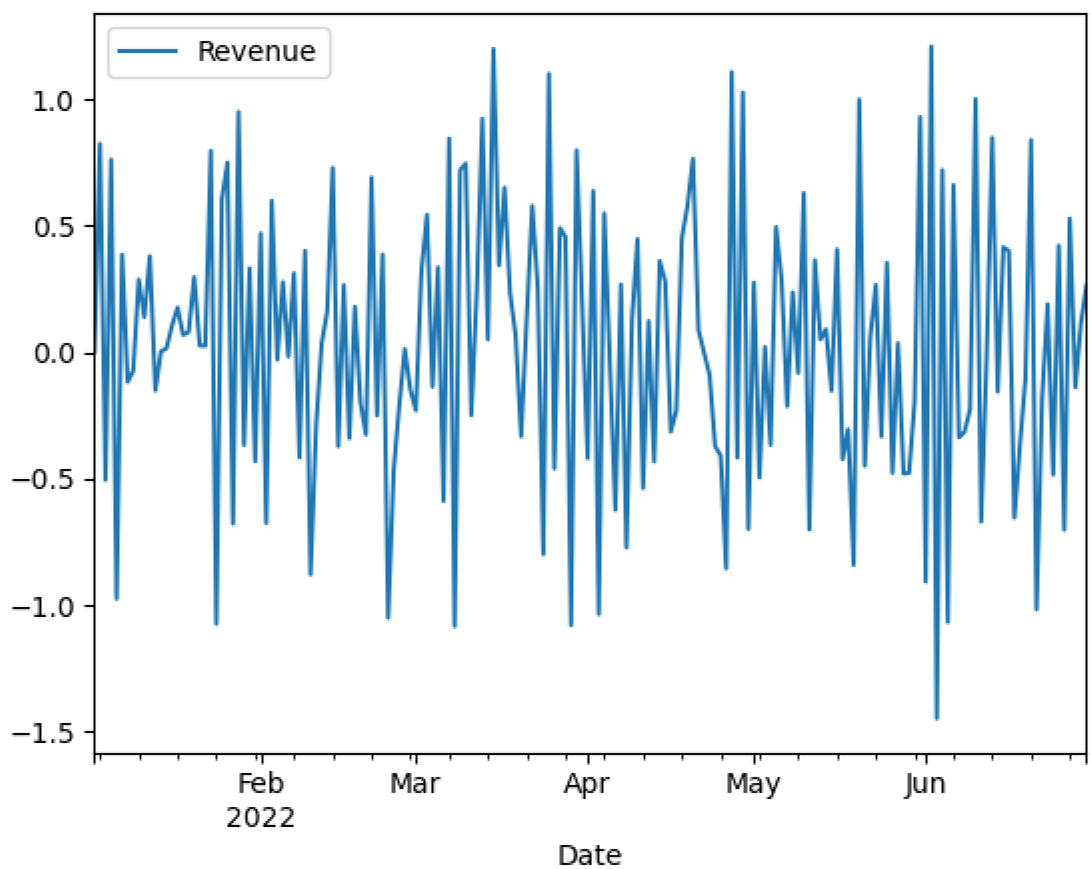
In [15]:
```
#Saving the training and testing data sets
ts_diff.to_csv("C:/Users/cklni/Desktop/WGU/D213/Task 1/D213clean.csv")
x_train.to_csv("C:/Users/cklni/Desktop/WGU/D213/Task 1/Xtrain.csv")
x_test.to_csv("C:/Users/cklni/Desktop/WGU/D213/Task 1/Xtest.csv")
```

Part IV: Model Identification and Analysis

D1

In [16]:
```
#Checking subset of the data for seasonality
ts_diff.loc[:'2022-06-30'].plot()
```

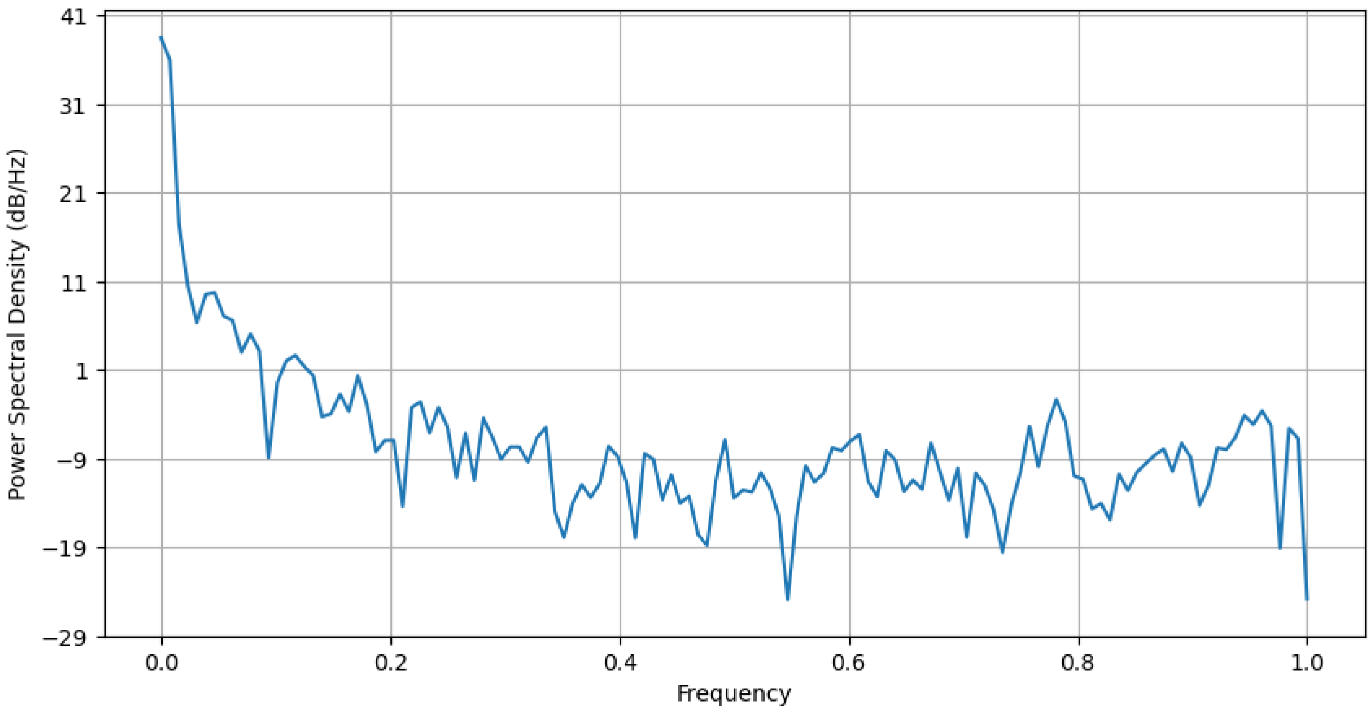
Out[16]: <Axes: xlabel='Date'>



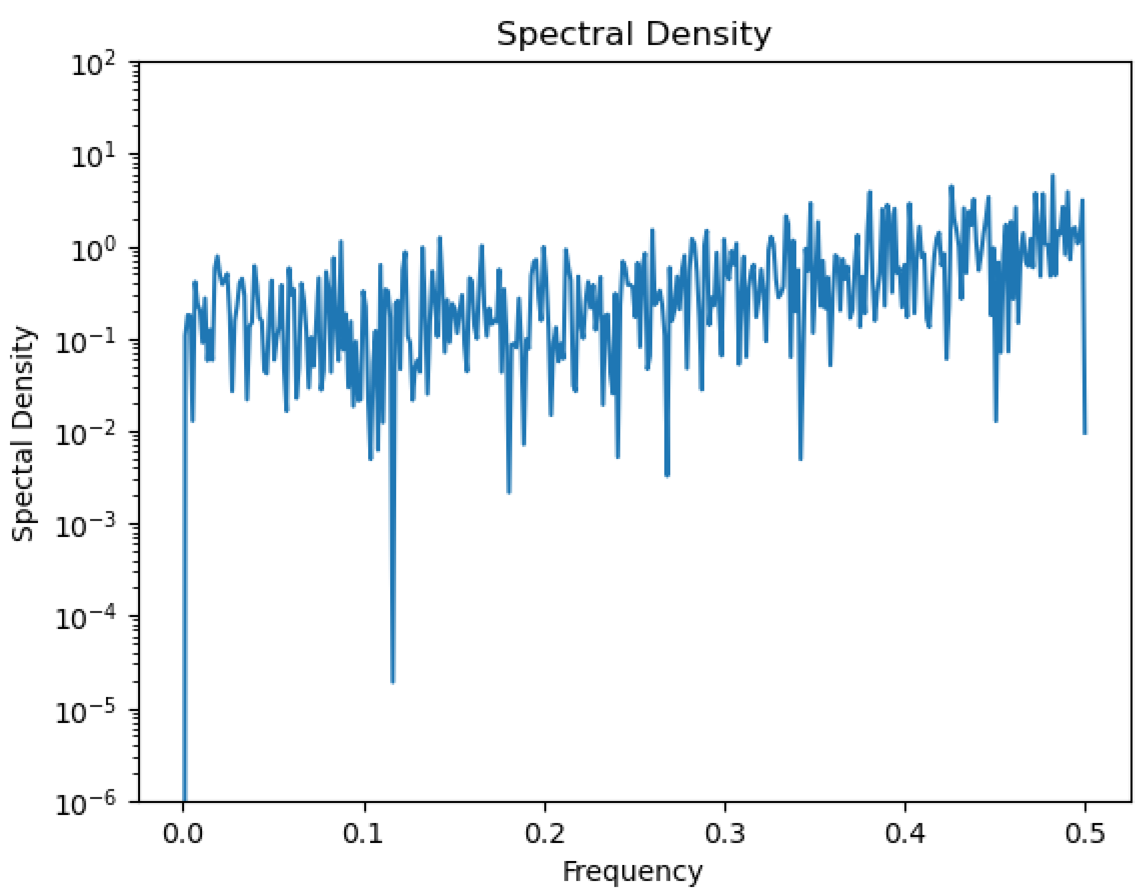
The above visualization of a subset of the data shows no obvious seasonality.

In [17]:
```
#Spectral analysis on the non-stationary revenue data
plt.figure(figsize=(10,5), linewidth=3)
plt.psd(ts['Revenue'])
```

Out[17]:
```
(array([6.97387711e+03, 3.91439441e+03, 5.47611144e+01, 1.10791953e+01,
        4.25439684e+00, 8.90767103e+00, 9.32436541e+00, 5.07699490e+00,
        4.50820186e+00, 1.97142105e+00, 3.17930644e+00, 2.05465010e+00,
        1.27856111e-01, 9.12357670e-01, 1.58928494e+00, 1.82600308e+00,
        1.37345831e+00, 1.08602805e+00, 3.71664573e-01, 4.01447712e-01,
        6.67019362e-01, 4.28134847e-01, 1.08303874e+00, 5.05461139e-01,
        1.49612021e-01, 2.01156666e-01, 2.01929017e-01, 3.62482513e-02,
        4.74386760e-01, 5.46619399e-01, 2.44460915e-01, 4.75225583e-01,
        2.85007556e-01, 7.59631735e-02, 2.43183009e-01, 7.19503505e-02,
        3.60475608e-01, 2.18971480e-01, 1.23421972e-01, 1.69234109e-01,
        1.69371921e-01, 1.14748035e-01, 2.14144408e-01, 2.82736486e-01,
        3.17059554e-02, 1.62458482e-02, 3.98324693e-02, 6.38895068e-02,
        4.57061837e-02, 6.54718394e-02, 1.72598760e-01, 1.32802101e-01,
        6.74716970e-02, 1.61089989e-02, 1.42266347e-01, 1.23640967e-01,
        4.30737122e-02, 8.17619769e-02, 3.95102804e-02, 4.74833201e-02,
        1.73085344e-02, 1.32485092e-02, 7.14595244e-02, 2.04348554e-01,
        4.51051907e-02, 5.55609932e-02, 5.26341931e-02, 8.68327071e-02,
        5.86340275e-02, 2.88124318e-02, 3.23398046e-03, 2.85277556e-02,
        1.03859619e-01, 6.85761289e-02, 8.53467489e-02, 1.65815053e-01,
        1.53444767e-01, 1.97776090e-01, 2.33413022e-01, 6.95161339e-02,
        4.68937467e-02, 1.53353843e-01, 1.21391009e-01, 5.34892233e-02,
        7.17250734e-02, 5.68272472e-02, 1.87708648e-01, 8.92439810e-02,
        4.25453689e-02, 9.79721726e-02, 1.63975102e-02, 8.61687997e-02,
        6.31239429e-02, 3.34886298e-02, 1.09875888e-02, 3.83717041e-02,
        8.81504336e-02, 2.89832055e-01, 1.01621880e-01, 2.88982189e-01,
        5.81784635e-01, 3.29483923e-01, 8.00152896e-02, 7.31104074e-02,
        3.39216170e-02, 3.93543037e-02, 2.55189466e-02, 8.36160222e-02,
        5.48857747e-02, 8.78315482e-02, 1.10183077e-01, 1.37188728e-01,
        1.60562556e-01, 9.01524835e-02, 1.87554502e-01, 1.29971998e-01,
        3.71787543e-02, 6.31650566e-02, 1.65356465e-01, 1.58164919e-01,
        2.17681939e-01, 3.85248684e-01, 3.04221135e-01, 4.32125608e-01,
        2.94750935e-01, 1.22618052e-02, 2.74845469e-01, 2.10555482e-01,
        3.28604031e-03]),
 array([0.       , 0.0078125, 0.015625 , 0.0234375, 0.03125  , 0.0390625,
        0.046875 , 0.0546875, 0.0625   , 0.0703125, 0.078125 , 0.0859375,
        0.09375  , 0.1015625, 0.109375 , 0.1171875, 0.125    , 0.1328125,
        0.140625 , 0.1484375, 0.15625  , 0.1640625, 0.171875 , 0.1796875,
        0.1875   , 0.1953125, 0.203125 , 0.2109375, 0.21875  , 0.2265625,
        0.234375 , 0.2421875, 0.25     , 0.2578125, 0.265625 , 0.2734375,
        0.28125  , 0.2890625, 0.296875 , 0.3046875, 0.3125   , 0.3203125,
        0.328125 , 0.3359375, 0.34375  , 0.3515625, 0.359375 , 0.3671875,
        0.375    , 0.3828125, 0.390625 , 0.3984375, 0.40625  , 0.4140625,
        0.421875 , 0.4296875, 0.4375   , 0.4453125, 0.453125 , 0.4609375,
        0.46875  , 0.4765625, 0.484375 , 0.4921875, 0.5      , 0.5078125,
        0.515625 , 0.5234375, 0.53125  , 0.5390625, 0.546875 , 0.5546875,
        0.5625   , 0.5703125, 0.578125 , 0.5859375, 0.59375  , 0.6015625,
        0.609375 , 0.6171875, 0.625    , 0.6328125, 0.640625 , 0.6484375,
        0.65625  , 0.6640625, 0.671875 , 0.6796875, 0.6875   , 0.6953125,
        0.703125 , 0.7109375, 0.71875  , 0.7265625, 0.734375 , 0.7421875,
        0.75     , 0.7578125, 0.765625 , 0.7734375, 0.78125  , 0.7890625,
        0.796875 , 0.8046875, 0.8125   , 0.8203125, 0.828125 , 0.8359375,
        0.84375  , 0.8515625, 0.859375 , 0.8671875, 0.875    , 0.8828125,
        0.890625 , 0.8984375, 0.90625  , 0.9140625, 0.921875 , 0.9296875,
        0.9375   , 0.9453125, 0.953125 , 0.9609375, 0.96875  , 0.9765625,
        0.984375 , 0.9921875, 1.       ]))
```

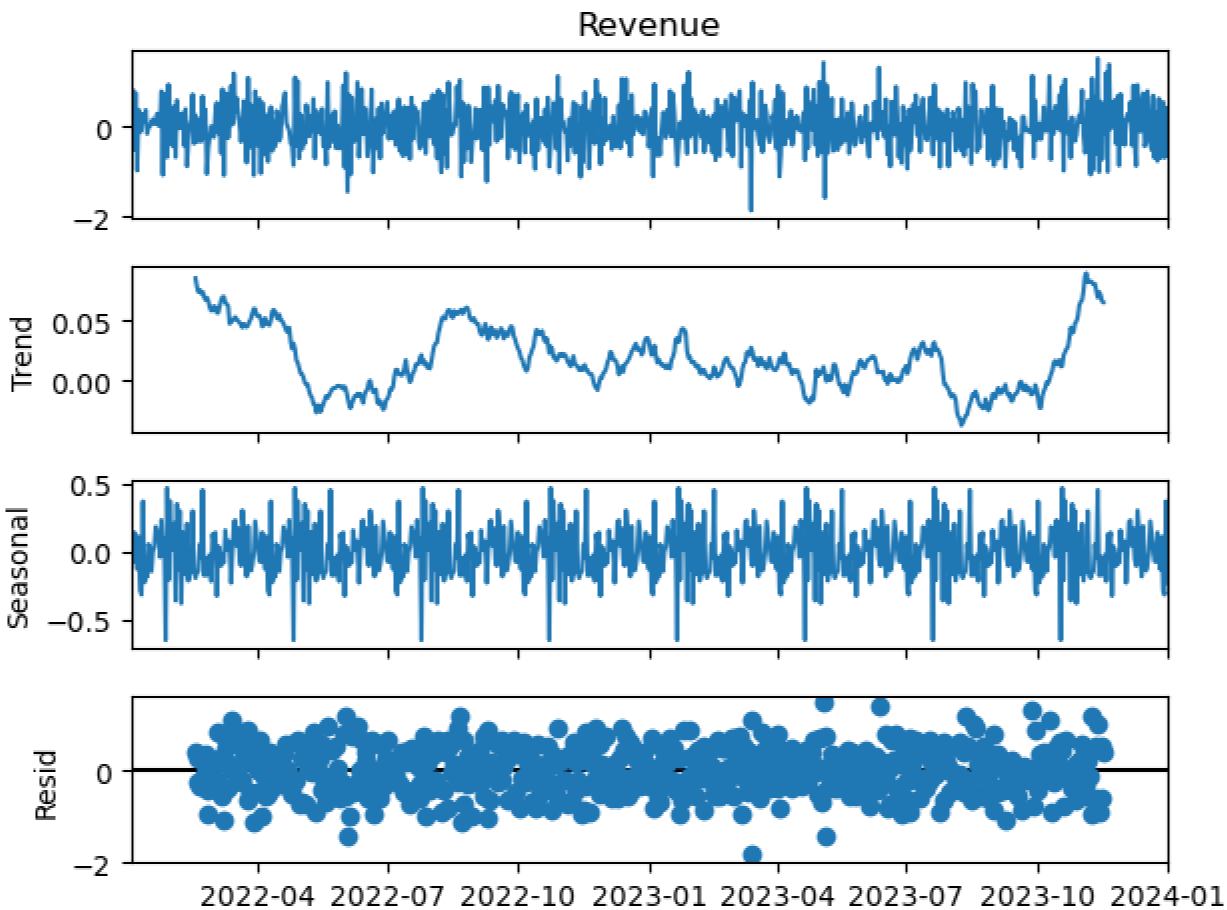


In [18]:
```
#Spectral analysis on the stationary revenue data
from scipy import signal
f, Pxx_den = signal.periodogram(ts_diff['Revenue'])
plt.semilogy(f, Pxx_den)
plt.ylim([1e-6, 1e2])
plt.title('Spectral Density')
plt.xlabel('Frequency')
plt.ylabel('Spectal Density')
plt.show()
```



The spectral density graph above shows a lack of seasonality due to the spikes in it.

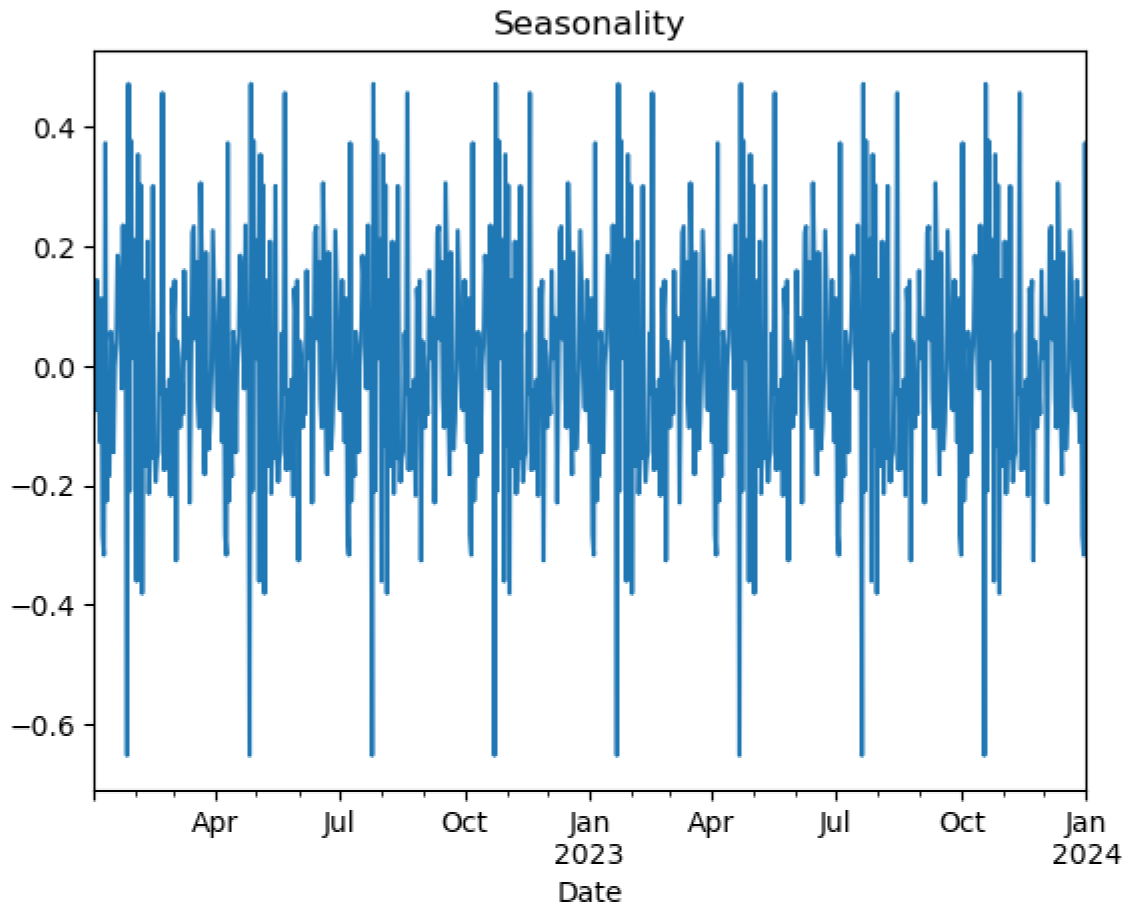
In [19]:
```
#Decomposition.
decomp = seasonal_decompose(ts_diff['Revenue'], period=90)


decomp.plot()
plt.show()
```



The above is the decomposed time series. It shows the revenue, trend, seasonality, and residual graphs.

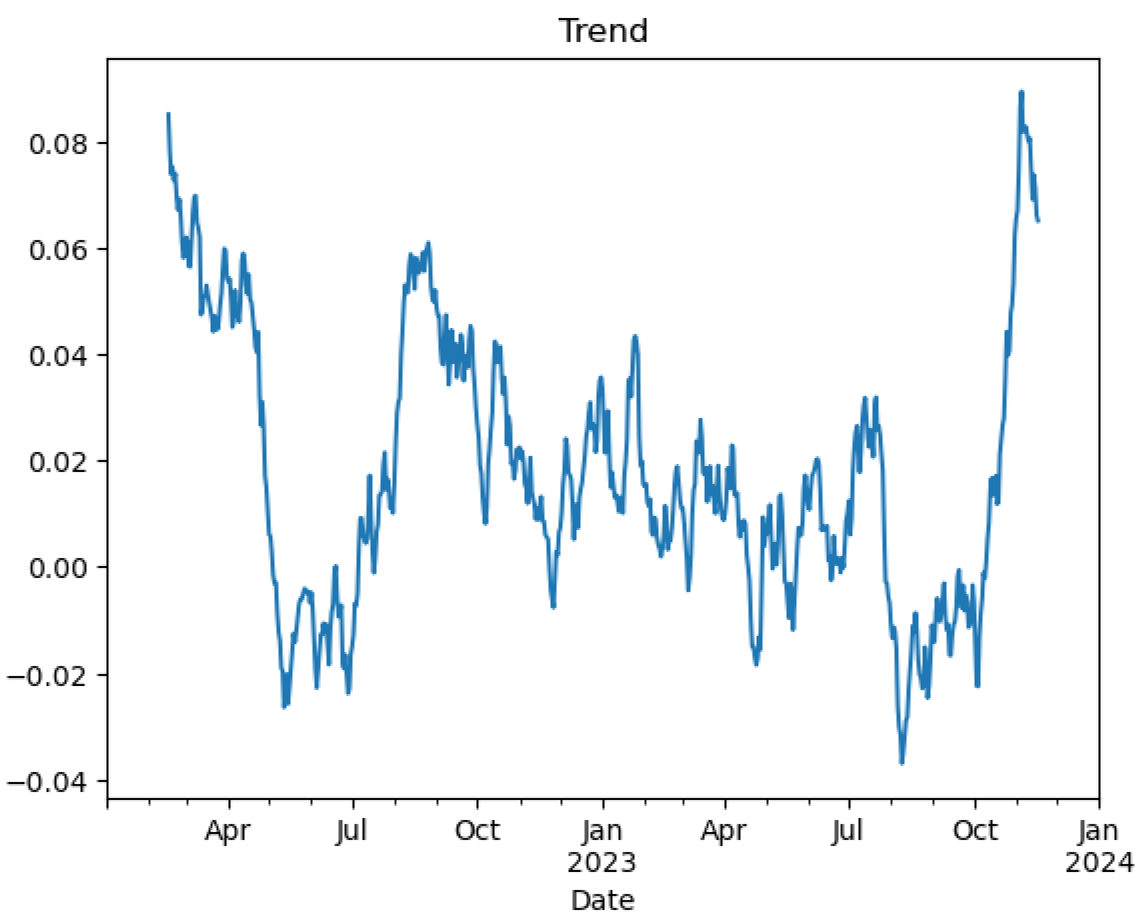
In [20]:
```
#Plot seasonality from the non-stationary data whose value is in the vairable decomp
plt.title('Seasonality')
decomp.seasonal.plot()
```

Out[20]:
```
<Axes: title={'center': 'Seasonality'}, xlabel='Date'>
```



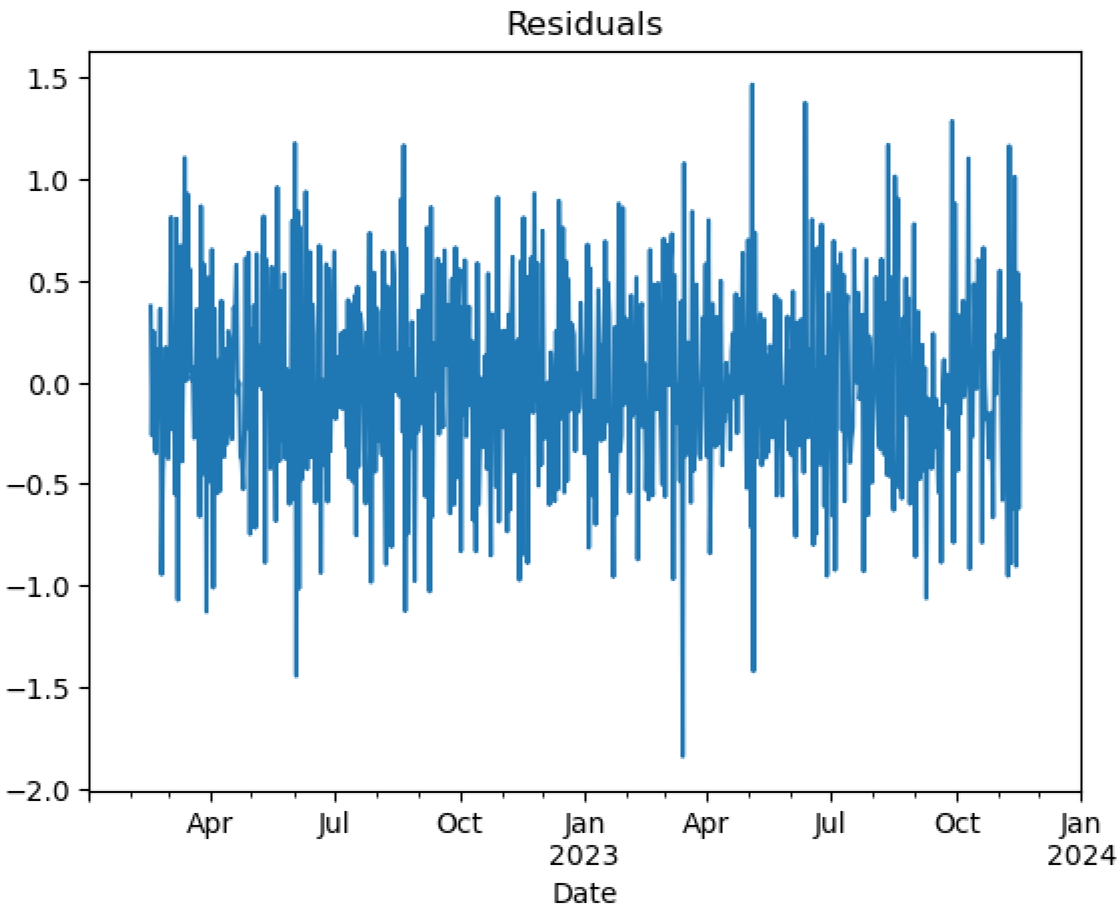
In [21]:
```
plt.title('Trend')
decomp.trend.plot()
```

Out[21]: <Axes: title={'center': 'Trend'}, xlabel='Date'>



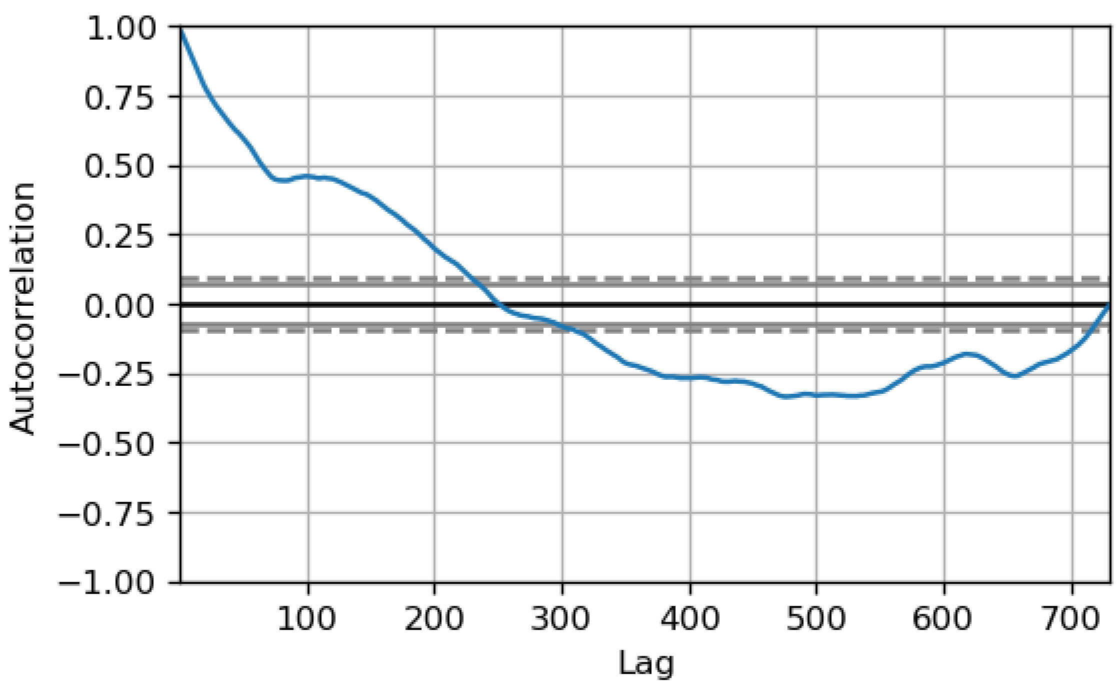
In [22]:
```
plt.title('Residuals')
decomp.resid.plot()
```

Out[22]:
```
<Axes: title={'center': 'Residuals'}, xlabel='Date'>
```



In [23]:
```
#Autocorrelation on non-stationary data
plt.rcParams.update({'figure.figsize':(5,3), 'figure.dpi':120})
pd.plotting.autocorrelation_plot(ts.Revenue.tolist())
```

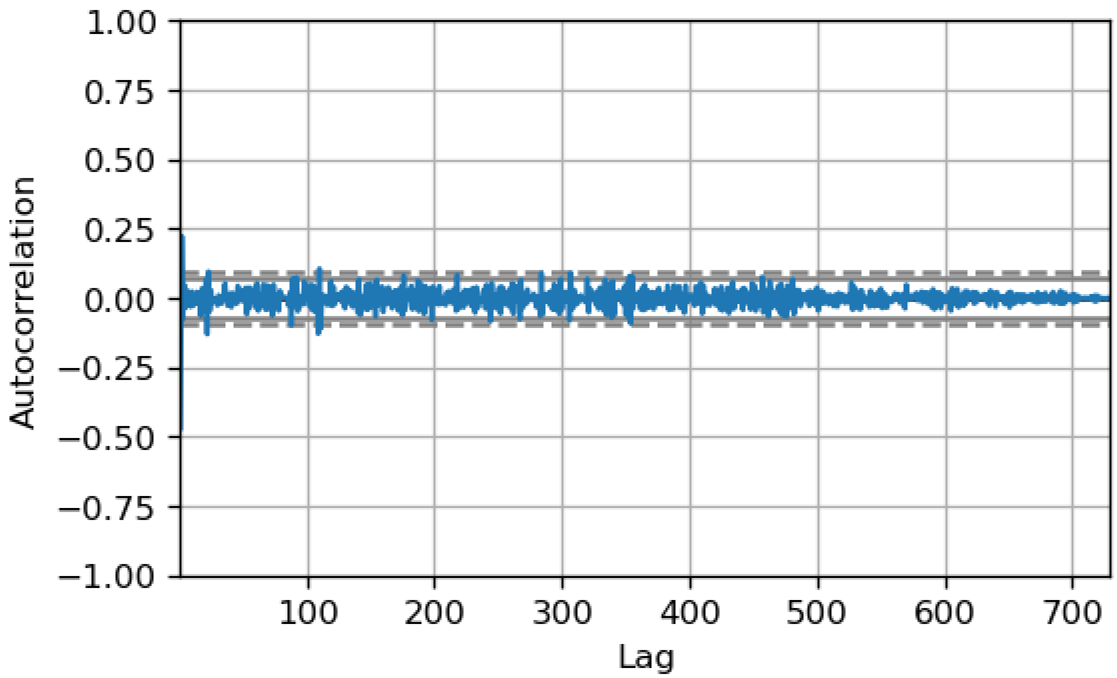
Out[23]:
```
<Axes: xlabel='Lag', ylabel='Autocorrelation'>
```



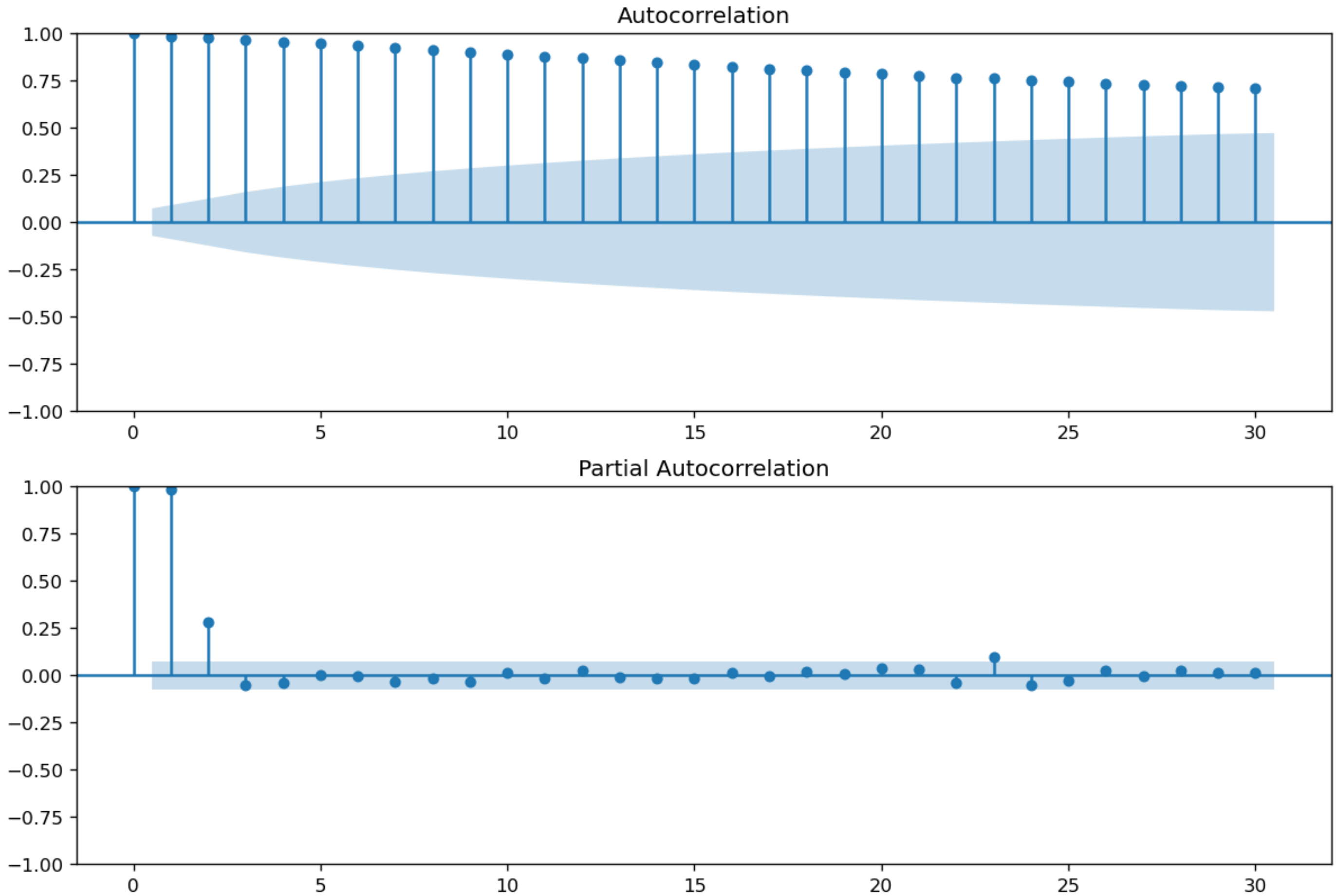
In [24]:
```
#Autocorrelation on stationary data set
```

```
plt.rcParams.update({'figure.figsize':(5,3), 'figure.dpi':120})
pd.plotting.autocorrelation_plot(ts_diff.Revenue.tolist())
```

Out[24]: <Axes: xlabel='Lag', ylabel='Autocorrelation'>

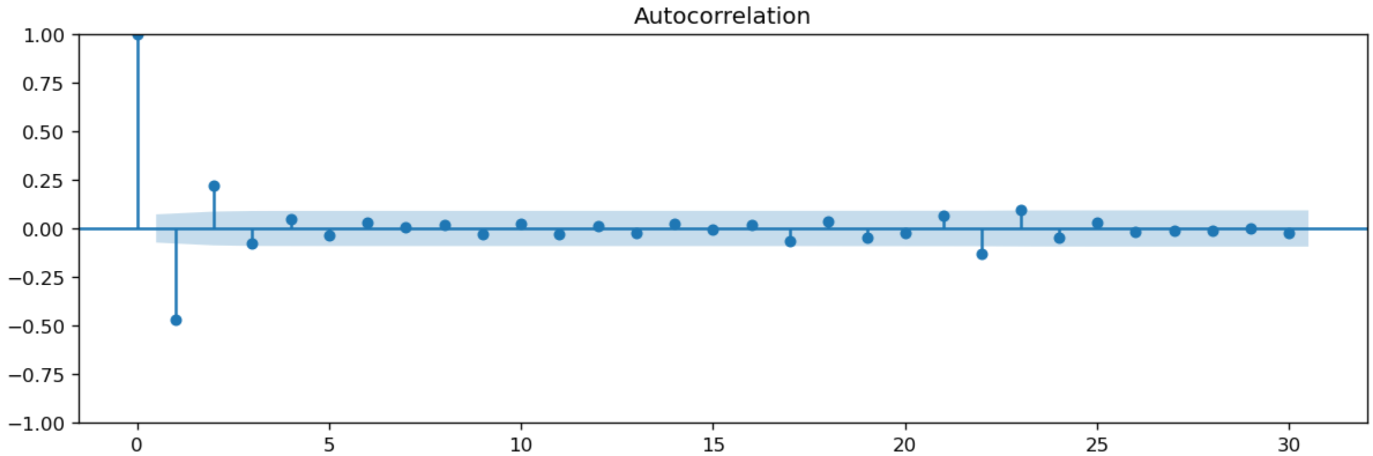


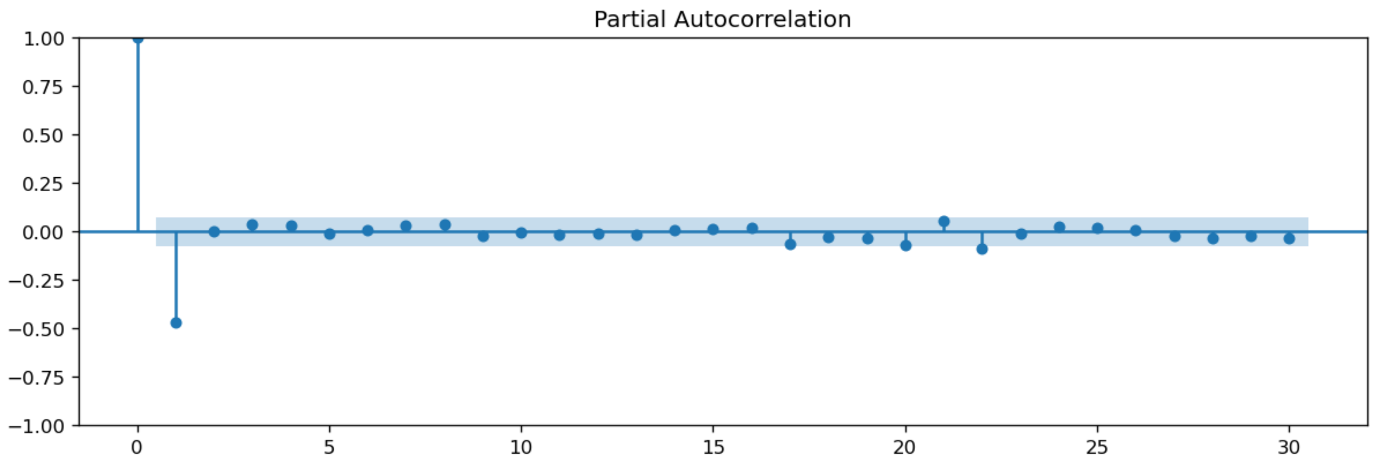
In [25]:
```
#Graphing ACF plot on non-stationary data
import statsmodels.api as sm
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(ts.Revenue, lags=30, ax=ax1)

#Graphing PACF plot on non-stationary data
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(ts.Revenue, lags=30, ax=ax2)
plt.show()
```



The ACF of the data set before differencing shows a slow decrease in lag, which is another indication of the data being non-stationary.

In [26]:
```
#Graphing ACF plot on stationary data
import statsmodels.api as sm
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(ts_diff.Revenue, lags=30, ax=ax1)

#Graphing PACF plot on stationary data
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(ts_diff.Revenue, lags=30, ax=ax2)
plt.show()
```





The data set after differencing display most of the values in the ACF and PACF in the blue shaded area which means they are not statistically significant.

D2

In [28]:
```
from pmdarima import auto_arima
stepwise_fit=auto_arima(ts_diff['Revenue'],trace=True, suppress_warnings=True)
stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=987.305, Time=0.66 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=1162.819, Time=0.07 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=983.122, Time=0.10 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=1019.369, Time=0.13 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=1162.139, Time=0.06 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=985.104, Time=0.13 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=985.106, Time=0.08 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=986.045, Time=0.59 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=984.710, Time=0.05 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0] intercept
Total fit time: 1.877 seconds
```

Out[28]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	730
Model:	SARIMAX(1, 0, 0)	**Log Likelihood**	-488.561
Date:	Tue, 28 May 2024	**AIC**	983.122
Time:	13:55:15	**BIC**	996.901
Sample:	01-02-2022	**HQIC**	988.438
	- 01-01-2024		

Covariance Type:	opg

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0332	0.018	1.895	0.058	-0.001	0.068
ar.L1	-0.4692	0.033	-14.296	0.000	-0.534	-0.405
sigma2	0.2232	0.013	17.801	0.000	0.199	0.248

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2.05
Prob(Q):	0.96	**Prob(JB):**	0.36
Heteroskedasticity (H):	1.02	**Skew:**	-0.02
Prob(H) (two-sided):	0.85	**Kurtosis:**	2.74

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

Using Auto-ARIMA, the best fit model is 1,0,0 with no seasonality 0,0,0, and no periodicity 0.

D3-D5

In [29]:
```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(x_train['Revenue'],order=(1,0,0))
model = model.fit()
model.summary()
```

```
C:\Users\cklni\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\cklni\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\cklni\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
```

Out[29]:

SARIMAX Results

Dep. Variable:	Revenue	**No. Observations:**	700
Model:	ARIMA(1, 0, 0)	**Log Likelihood**	-465.219
Date:	Tue, 28 May 2024	**AIC**	936.439
Time:	13:55:54	**BIC**	950.092
Sample:	01-02-2022	**HQIC**	941.716
	- 12-02-2023		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0217	0.012	1.802	0.072	-0.002	0.045
ar.L1	-0.4737	0.033	-14.216	0.000	-0.539	-0.408

sigma2	0.2211	0.013	17.625	0.000	0.197	0.246

Ljung-Box (L1) (Q):	0.07	**Jarque-Bera (JB):**	1.43
Prob(Q):	0.80	**Prob(JB):**	0.49
Heteroskedasticity (H):	0.97	**Skew:**	-0.00
Prob(H) (two-sided):	0.81	**Kurtosis:**	2.78

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

In [30]:

```
start=len(x_train)
end=len(x_train)+len(x_test)-1
pred=model.predict(start=start, end=end, type='levels')
print(pred)
pred.index=ts.index[start:end+1]
```

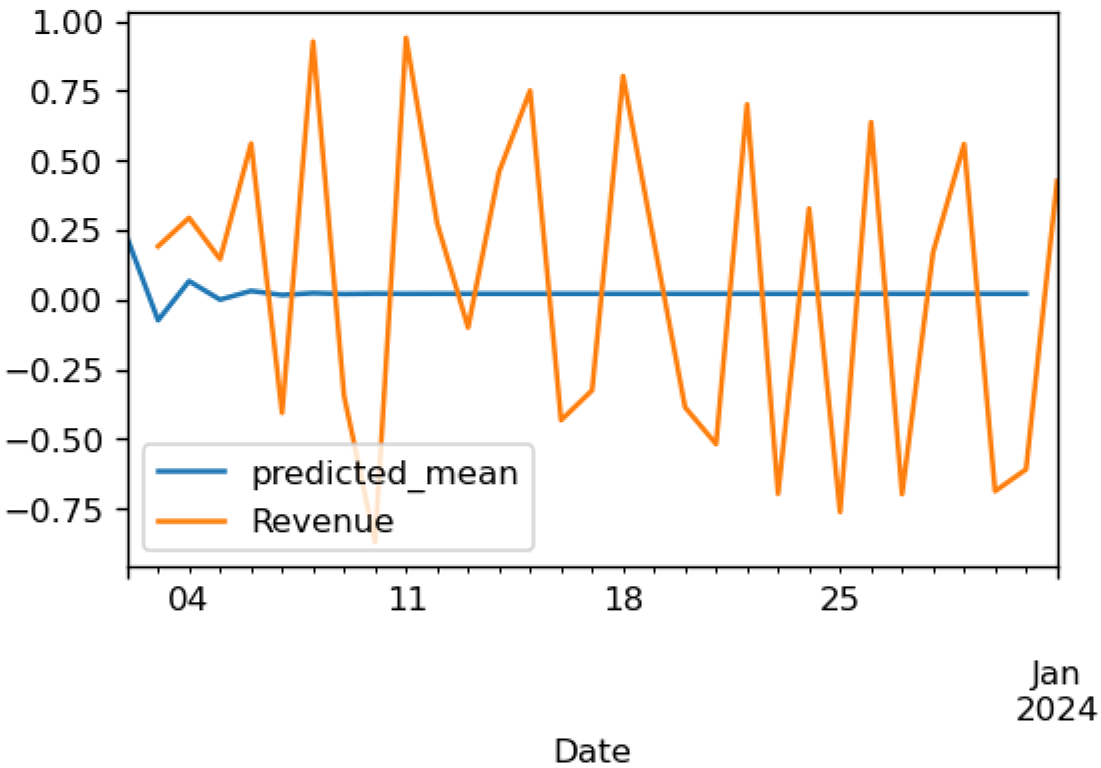
```
2023-12-03    0.223207
2023-12-04   -0.073697
2023-12-05    0.066949
2023-12-06    0.000324
2023-12-07    0.031885
2023-12-08    0.016934
2023-12-09    0.024016
2023-12-10    0.020662
2023-12-11    0.022251
2023-12-12    0.021498
2023-12-13    0.021855
2023-12-14    0.021686
2023-12-15    0.021766
2023-12-16    0.021728
2023-12-17    0.021746
2023-12-18    0.021737
2023-12-19    0.021741
2023-12-20    0.021739
2023-12-21    0.021740
2023-12-22    0.021740
2023-12-23    0.021740
2023-12-24    0.021740
2023-12-25    0.021740
2023-12-26    0.021740
2023-12-27    0.021740
2023-12-28    0.021740
2023-12-29    0.021740
2023-12-30    0.021740
2023-12-31    0.021740
2024-01-01    0.021740
Freq: D, Name: predicted_mean, dtype: float64
```

In [32]:

```
pred.plot(legend=True)
x_test['Revenue'].plot(legend=True)
```

Out[32]:

```
<Axes: xlabel='Date'>
```



In [33]:
```
x_test['Revenue'].mean()
```

Out[33]:
```
0.05142895333333284
```

In [34]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(pred,x_test['Revenue']))
print(rmse)
```

```
0.560851128036282
```

In [36]:
```
model2=ARIMA(ts_diff['Revenue'],order=(1,0,0))
model2 = model2.fit()
ts_diff.tail()
```

```
C:\Users\cklni\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\cklni\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\cklni\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
```

Out[36]:

	Revenue
Date	
2023-12-28	0.170280
2023-12-29	0.559108
2023-12-30	-0.687028
2023-12-31	-0.608824
2024-01-01	0.425985

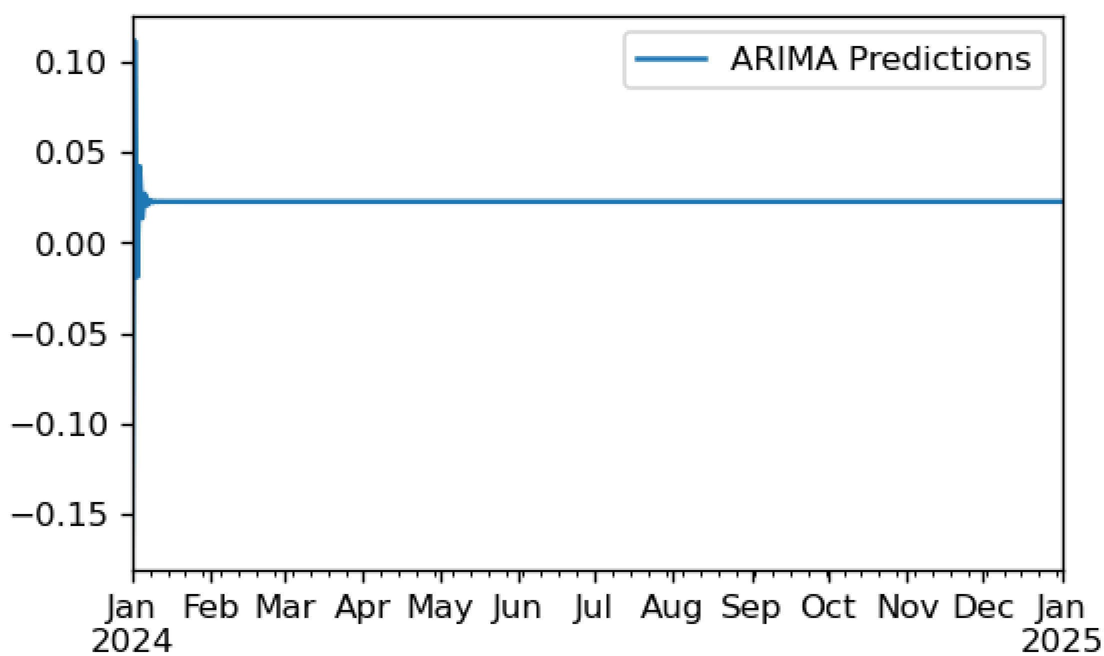
In [37]:
```
index_future_dates = pd.date_range(start='2024-01-01', end='2025-01-01')
```

```
print(index_future_dates)
pred=model2.predict(start=len(ts_diff), end=len(ts_diff)+366, typ='levels').rename('ARIM
pred.index=index_future_dates
print(pred)
```

```
DatetimeIndex(['2024-01-01', '2024-01-02', '2024-01-03', '2024-01-04',
               '2024-01-05', '2024-01-06', '2024-01-07', '2024-01-08',
               '2024-01-09', '2024-01-10',
               ...
               '2024-12-23', '2024-12-24', '2024-12-25', '2024-12-26',
               '2024-12-27', '2024-12-28', '2024-12-29', '2024-12-30',
               '2024-12-31', '2025-01-01'],
              dtype='datetime64[ns]', length=367, freq='D')
2024-01-01   -0.166684
2024-01-02    0.111411
2024-01-03   -0.019078
2024-01-04    0.042151
2024-01-05    0.013421
                ...
2024-12-28    0.022596
2024-12-29    0.022596
2024-12-30    0.022596
2024-12-31    0.022596
2025-01-01    0.022596
Freq: D, Name: ARIMA Predictions, Length: 367, dtype: float64
```

In [38]:
```
pred.plot(legend=True)
```

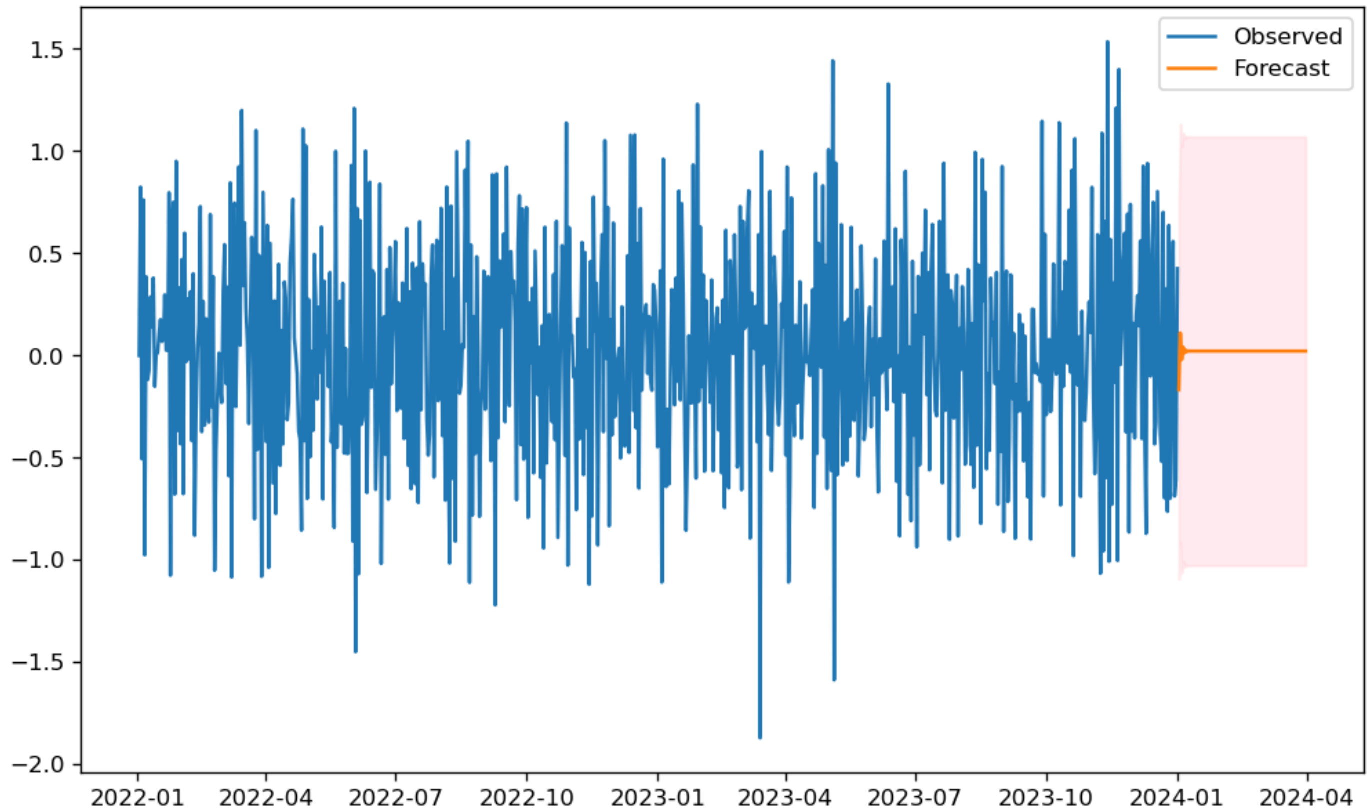
Out[38]: <Axes: >



In [39]:
```
diff_forecast = model2.get_forecast(steps=90)
mean_forecast = diff_forecast.predicted_mean
confidence_intervals = diff_forecast.conf_int()
lower_limits = confidence_intervals.loc[:,'lower Revenue']
upper_limits = confidence_intervals.loc[:,'upper Revenue']
```

In [42]:
```
plt.figure(figsize=(10, 6))
plt.plot(ts_diff, label='Observed')
plt.plot(mean_forecast, label='Forecast')
plt.fill_between(confidence_intervals.index,
                 confidence_intervals.iloc[:, 0],
                 confidence_intervals.iloc[:, 1], color='pink', alpha=0.3)
```

```
plt.legend()
plt.show()
```



Part V: Data Summary and Implications

E1

The ARIMA model was the correct model to select due to no seasonality being present in the time series. Using Auto-ARIMA, the best fit model is 1,0,0 with no seasonality 0,0,0, and no periodicity 0. The forecast prediction interval is 1 day. Our TM data consists of 2 years of daily revenue. Consequently, the ARIMA model identifies correlations and seasonality to predict revenue daily. The forecast is set to 90 days. The two years of revenue data is good enough to forecast up to a year of future revenue. Setting it to 90 days allows the results to be more accurate than the year but long enough to give good insights as many companies do quarterly revenue calls. As pointed out previously I used Auto-Arima to find the best fit model. Using the root mean squared error, RMSE, which resulted with approximately 0.561. A number closer to 0 would be ideal.

E2

The annotated visualization of the forecast of the final model compared to the test set is found above in the last graph.

E3

I would recommend that the company would take these results for the trend of revenue for the next quarter and see how they could expand the company to cause revenue to increase.

In [2]:
```
pip install nbconvert[webpdf]
```

```
Requirement already satisfied: nbconvert[webpdf] in c:\users\cklni\anaconda3\lib\site-pa
ckages (6.5.4)
Requirement already satisfied: lxml in c:\users\cklni\anaconda3\lib\site-packages (from
nbconvert[webpdf]) (4.9.2)
Requirement already satisfied: beautifulsoup4 in c:\users\cklni\anaconda3\lib\site-packa
ges (from nbconvert[webpdf]) (4.12.2)
Requirement already satisfied: bleach in c:\users\cklni\anaconda3\lib\site-packages (fro
m nbconvert[webpdf]) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\cklni\anaconda3\lib\site-packages
(from nbconvert[webpdf]) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\cklni\anaconda3\lib\site-p
ackages (from nbconvert[webpdf]) (0.4)
Requirement already satisfied: jinja2>=3.0 in c:\users\cklni\anaconda3\lib\site-packages
(from nbconvert[webpdf]) (3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\cklni\anaconda3\lib\site-pa
ckages (from nbconvert[webpdf]) (5.3.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\cklni\anaconda3\lib\site-
packages (from nbconvert[webpdf]) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\cklni\anaconda3\lib\site-pack
ages (from nbconvert[webpdf]) (2.1.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\cklni\anaconda3\lib\site-pa
ckages (from nbconvert[webpdf]) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\cklni\anaconda3\lib\site-pack
ages (from nbconvert[webpdf]) (0.5.13)
Requirement already satisfied: nbformat>=5.1 in c:\users\cklni\anaconda3\lib\site-packag
es (from nbconvert[webpdf]) (5.7.0)
Requirement already satisfied: packaging in c:\users\cklni\anaconda3\lib\site-packages
(from nbconvert[webpdf]) (23.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\cklni\anaconda3\lib\site
-packages (from nbconvert[webpdf]) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in c:\users\cklni\anaconda3\lib\site-pack
ages (from nbconvert[webpdf]) (2.15.1)
Requirement already satisfied: tinycss2 in c:\users\cklni\anaconda3\lib\site-packages (f
rom nbconvert[webpdf]) (1.2.1)
Requirement already satisfied: traitlets>=5.0 in c:\users\cklni\anaconda3\lib\site-packa
ges (from nbconvert[webpdf]) (5.7.1)
Requirement already satisfied: pyppeteer<1.1,>=1 in c:\users\cklni\anaconda3\lib\site-pa
ckages (from nbconvert[webpdf]) (1.0.2)
Requirement already satisfied: platformdirs>=2.5 in c:\users\cklni\anaconda3\lib\site-pa
ckages (from jupyter-core>=4.7->nbconvert[webpdf]) (2.5.2)
Requirement already satisfied: pywin32>=300 in c:\users\cklni\anaconda3\lib\site-package
s (from jupyter-core>=4.7->nbconvert[webpdf]) (305.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\cklni\anaconda3\lib\sit
e-packages (from nbclient>=0.5.0->nbconvert[webpdf]) (7.4.9)
Requirement already satisfied: nest-asyncio in c:\users\cklni\anaconda3\lib\site-package
s (from nbclient>=0.5.0->nbconvert[webpdf]) (1.5.6)
Requirement already satisfied: fastjsonschema in c:\users\cklni\anaconda3\lib\site-packa
ges (from nbformat>=5.1->nbconvert[webpdf]) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\cklni\anaconda3\lib\site-pack
ages (from nbformat>=5.1->nbconvert[webpdf]) (4.17.3)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\cklni\anaconda3\lib\sit
e-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (1.4.4)
Requirement already satisfied: certifi>=2021 in c:\users\cklni\anaconda3\lib\site-packag
es (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (2023.7.22)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\cklni\anaconda3\lib\s
ite-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (6.0.0)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in c:\users\cklni\anaconda3\lib\site-p
ackages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (8.2.2)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\cklni\anaconda3\lib\site-
packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (4.65.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\cklni\anaconda3\lib\si
te-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (1.26.16)
Requirement already satisfied: websockets<11.0,>=10.0 in c:\users\cklni\anaconda3\lib\si
te-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf]) (10.4)
Requirement already satisfied: soupsieve>1.2 in c:\users\cklni\anaconda3\lib\site-packag
es (from beautifulsoup4->nbconvert[webpdf]) (2.4)
```

```
Requirement already satisfied: six>=1.9.0 in c:\users\cklni\anaconda3\lib\site-packages
(from bleach->nbconvert[webpdf]) (1.16.0)
Requirement already satisfied: webencodings in c:\users\cklni\anaconda3\lib\site-package
s (from bleach->nbconvert[webpdf]) (0.5.1)
Requirement already satisfied: zipp>=0.5 in c:\users\cklni\anaconda3\lib\site-packages
(from importlib-metadata>=1.4->pyppeteer<1.1,>=1->nbconvert[webpdf]) (3.11.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\cklni\anaconda3\lib\site-packag
es (from jsonschema>=2.6->nbformat>=5.1->nbconvert[webpdf]) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users
\cklni\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert[webpd
f]) (0.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\cklni\anaconda3\lib\si
te-packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert[webpdf]) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\cklni\anaconda3\lib\site-packages
(from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert[webpdf]) (23.2.0)
Requirement already satisfied: tornado>=6.2 in c:\users\cklni\anaconda3\lib\site-package
s (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert[webpdf]) (6.3.2)
Requirement already satisfied: colorama in c:\users\cklni\anaconda3\lib\site-packages (f
rom tqdm<5.0.0,>=4.42.1->pyppeteer<1.1,>=1->nbconvert[webpdf]) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install pyppeteer
```

```
Requirement already satisfied: pyppeteer in c:\users\cklni\anaconda3\lib\site-packages
(1.0.2)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\cklni\anaconda3\lib\sit
e-packages (from pyppeteer) (1.4.4)
Requirement already satisfied: certifi>=2021 in c:\users\cklni\anaconda3\lib\site-packag
es (from pyppeteer) (2023.7.22)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\cklni\anaconda3\lib\s
ite-packages (from pyppeteer) (6.0.0)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in c:\users\cklni\anaconda3\lib\site-p
ackages (from pyppeteer) (8.2.2)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\cklni\anaconda3\lib\site-
packages (from pyppeteer) (4.65.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\cklni\anaconda3\lib\si
te-packages (from pyppeteer) (1.26.16)
Requirement already satisfied: websockets<11.0,>=10.0 in c:\users\cklni\anaconda3\lib\si
te-packages (from pyppeteer) (10.4)
Requirement already satisfied: zipp>=0.5 in c:\users\cklni\anaconda3\lib\site-packages
(from importlib-metadata>=1.4->pyppeteer) (3.11.0)
Requirement already satisfied: colorama in c:\users\cklni\anaconda3\lib\site-packages (f
rom tqdm<5.0.0,>=4.42.1->pyppeteer) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
!jupyter nbconvert --to webpdf "Christian LeBlanc D213 Task 1 V1.ipynb" --allow-chromium
```

```
[NbConvertApp] Converting notebook Christian LeBlanc D213 Task 1 V1.ipynb to webpdf
[NbConvertApp] Building PDF
[INFO] Starting Chromium download.

  0%|          | 0.00/137M [00:00<?, ?b/s]
  0%|          | 399k/137M [00:00<00:34, 3.90Mb/s]
  2%|1         | 2.12M/137M [00:00<00:11, 11.5Mb/s]
  7%|6         | 9.38M/137M [00:00<00:03, 37.1Mb/s]
 10%|9         | 13.7M/137M [00:00<00:03, 39.0Mb/s]
 18%|#8        | 25.2M/137M [00:00<00:01, 63.1Mb/s]
 25%|##4       | 33.6M/137M [00:00<00:02, 47.6Mb/s]
 31%|###       | 42.0M/137M [00:01<00:02, 40.3Mb/s]
 36%|###6      | 49.3M/137M [00:01<00:01, 46.7Mb/s]
 40%|###9      | 54.8M/137M [00:01<00:01, 47.0Mb/s]
 44%|####3     | 60.0M/137M [00:01<00:02, 38.1Mb/s]
 51%|#####1    | 70.1M/137M [00:01<00:01, 51.0Mb/s]
 57%|#####6    | 77.4M/137M [00:01<00:01, 55.8Mb/s]
 61%|######1   | 83.9M/137M [00:01<00:01, 51.5Mb/s]
 65%|######5   | 89.7M/137M [00:01<00:00, 52.7Mb/s]
```

```
 70%|######9   | 95.4M/137M [00:02<00:00, 52.3Mb/s]
 74%|#######3  | 101M/137M [00:02<00:00, 53.3Mb/s]
 80%|#######9  | 109M/137M [00:02<00:00, 53.1Mb/s]
 84%|########3 | 115M/137M [00:02<00:00, 53.2Mb/s]
 93%|#########2| 127M/137M [00:02<00:00, 67.2Mb/s]
100%|##########| 137M/137M [00:02<00:00, 53.2Mb/s]
[INFO] Beginning extraction
[INFO] Chromium extracted to: C:\Users\cklni\AppData\Local\pyppeteer\pyppeteer\local-chr
omium\588429
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 966577 bytes to Christian LeBlanc D213 Task 1 V1.pdf
```

In [ ]: