In [388]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xlrd
import pmdarima
import arch
from matplotlib.dates import MONDAY
from matplotlib.dates import WeekdayLocator
import matplotlib.ticker as ticker
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller, kpss, coint
import statsmodels.tsa.stattools as smtools
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.vector_ar.vecm import coint_johansen
import statsmodels.api as sm
import seaborn as sns
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.plotting import autocorrelation_plot
from pmdarima.arima import auto_arima
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from pmdarima import auto_arima
import yfinance as yf
import arch
from pandas.plotting import autocorrelation_plot
from arch.univariate import GARCH
from arch.univariate import EGARCH
from arch.unitroot import engle_granger
from scipy.stats import pearsonr
import warnings
warnings.filterwarnings('ignore')
```

Step 1. Data Importing

In [389]:
```
#creating the gold ETF data frame
gold_etf = yf.Ticker('GLD')
df_goldetf = gold_etf.history(start="2020-03-01",  end="2020-12-31")
df_goldetf.index = pd.to_datetime(df_goldetf.index)
df_goldetf = df_goldetf.sort_index(ascending = True)
df_goldetf = df_goldetf.astype(float)
#print(df_goldetf.index)
df_goldetf.head()
```

Out[389]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-03-02	150.000000	150.729996	149.039993	149.199997	16295400.0	0.0	0.0
2020-03-03	150.839996	155.240005	150.740005	153.889999	28687700.0	0.0	0.0
2020-03-04	154.399994	154.960007	153.699997	154.160004	12315500.0	0.0	0.0
2020-03-05	156.059998	157.619995	155.720001	157.490005	17973500.0	0.0	0.0
2020-03-06	158.330002	159.250000	154.539993	157.550003	26973400.0	0.0	0.0

In [390]:
```
#creating the equity ETF dataframe
equity_etf = yf.Ticker('CSUK.L')
df_equityetf = equity_etf.history(start="2020-03-01",  end="2020-12-31")
df_equityetf.index = pd.to_datetime(df_equityetf.index)
df_equityetf = df_equityetf.sort_index(ascending = True)
df_equityetf = df_equityetf.astype(float)
df_equityetf.head()
```

Out[390]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-03-02	10090.0	10230.0	9890.0	10100.0	1629.0	0.0	0.0
2020-03-03	10236.0	10338.0	10236.0	10176.0	4437.0	0.0	0.0
2020-03-04	10368.0	10408.0	10306.0	10332.0	4675.0	0.0	0.0
2020-03-05	10384.0	10384.0	10164.0	10231.0	14929.0	0.0	0.0
2020-03-06	10040.0	10062.0	9834.0	9854.0	2880.0	0.0	0.0

In [391]:
```
#creating the bitcoin dataframe
bitcoin = yf.Ticker('BTC-USD')
df_bitcoin = bitcoin.history(start="2020-03-01",  end="2020-12-31")
df_bitcoin.index = pd.to_datetime(df_bitcoin.index)
df_bitcoin = df_bitcoin.sort_index(ascending = True)
df_bitcoin = df_bitcoin.astype(float)
df_bitcoin.head()
```

Out[391]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-02-29	8671.212891	8775.631836	8599.508789	8599.508789	3.579239e+10	0.0	0.0
2020-03-01	8599.758789	8726.796875	8471.212891	8562.454102	3.534916e+10	0.0	0.0
2020-03-02	8563.264648	8921.308594	8532.630859	8869.669922	4.285767e+10	0.0	0.0
2020-03-03	8865.387695	8901.598633	8704.990234	8787.786133	4.238672e+10	0.0	0.0
2020-03-04	8788.541992	8843.366211	8712.431641	8755.246094	3.474671e+10	0.0	0.0

Step 2. Data Processing

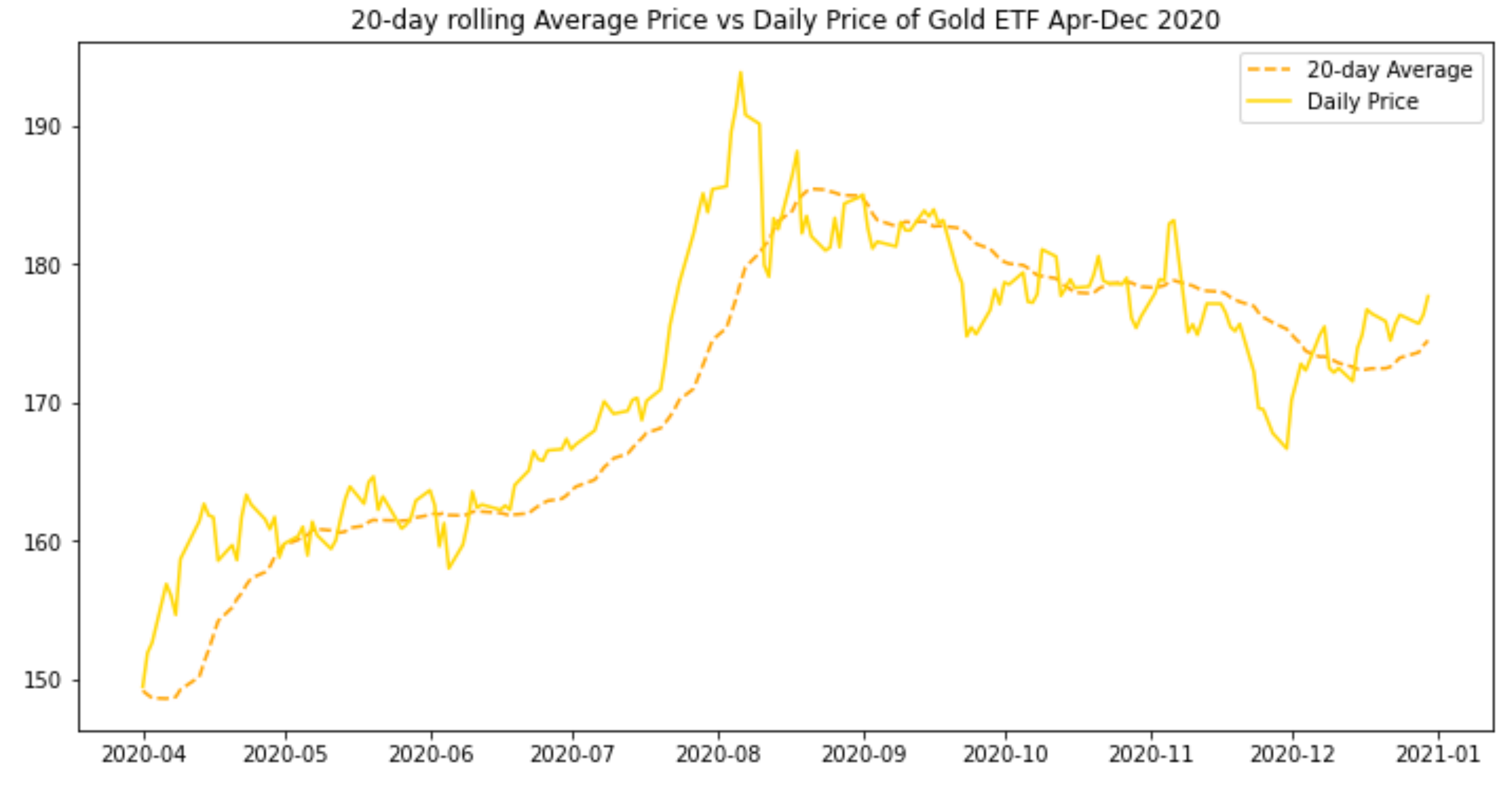
In [392]:
```
#Daily return on gold etf, equity etf and bitcoin
df_goldetf_daily_return = pd.DataFrame(np.log(df_goldetf['Close']/df_goldetf['Close'].shift(1)).iloc[1:]
                                       .rename('Gold ETF Daily Returns'))
df_equityetf_daily_return = pd.DataFrame(np.log(df_equityetf['Close']/df_equityetf['Close'].shift(1)).iloc[1:]
                                         .rename('Equity ETF Daily Returns'))
df_bitcoin_daily_return = pd.DataFrame(np.log(df_bitcoin['Close']/df_bitcoin['Close'].shift(1)).iloc[1:]
                                       .rename('Bitcoin-USD daily Returns'))
```

Steps 3 and 4. Data Summaries and Graphing

In [393]:
```
df_gold_avg = df_goldetf['Close'].rolling(20).mean().loc['Apr-2020':'Dec-2020']
df_equity_avg = df_equityetf['Close'].rolling(20).mean().loc['Apr-2020':'Dec-2020']
df_bitcoin_avg = df_bitcoin['Close'].rolling(20).mean().loc['Apr-2020':'Dec-2020']
```

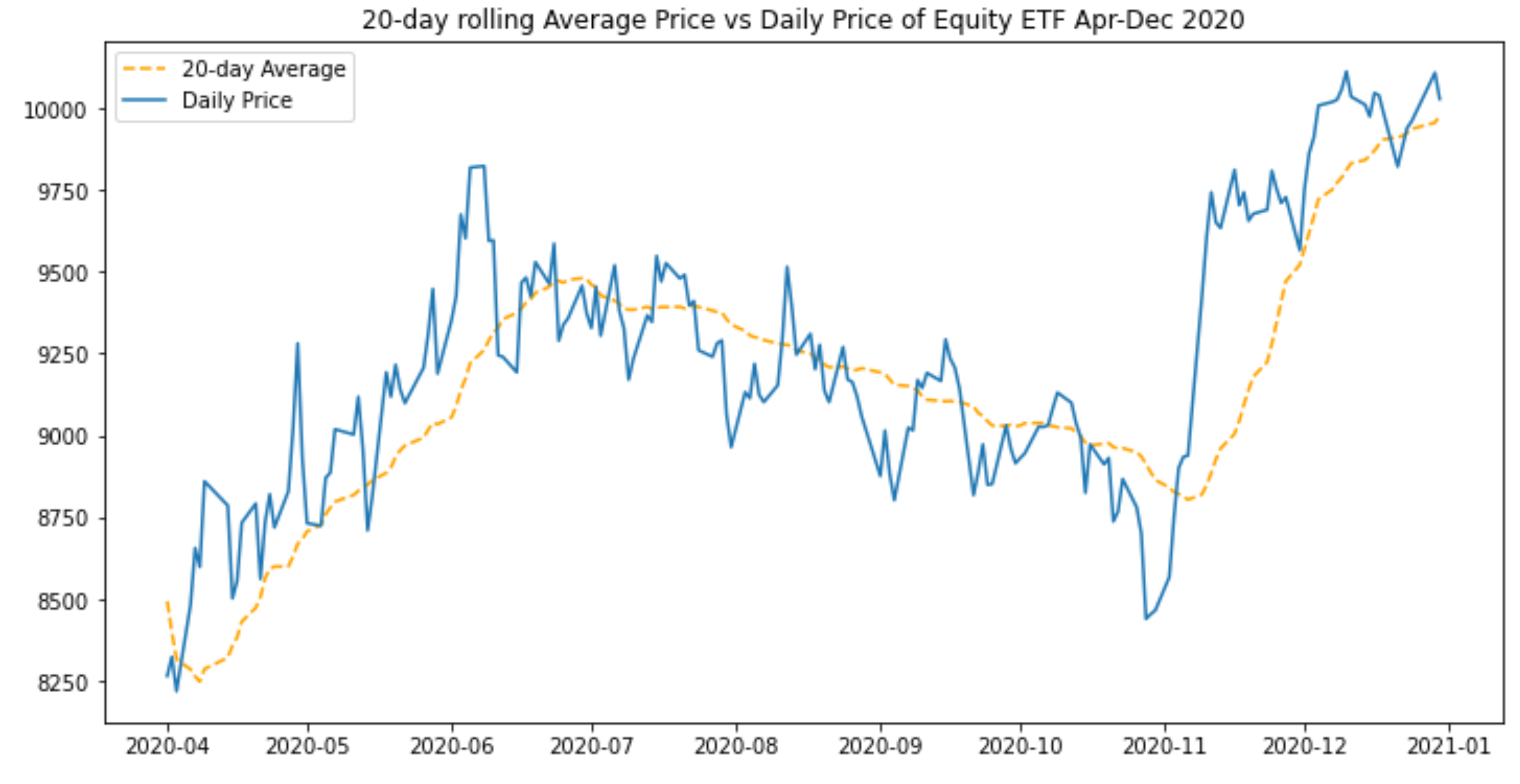
In [394]:
```
#20-day moving average price of your GOLD ETF.

fig, ax = plt.subplots(figsize=(12,6))
ax.set_title('20-day rolling Average Price vs Daily Price of Gold ETF Apr-Dec 2020')
ax.plot(df_gold_avg, label = '20-day Average', linestyle = '--', color = 'orange')
ax.plot(df_goldetf['Close'].loc['Apr-2020':'Dec-2020'], label = 'Daily Price', color = 'gold')
ax.legend()
plt.show()
```

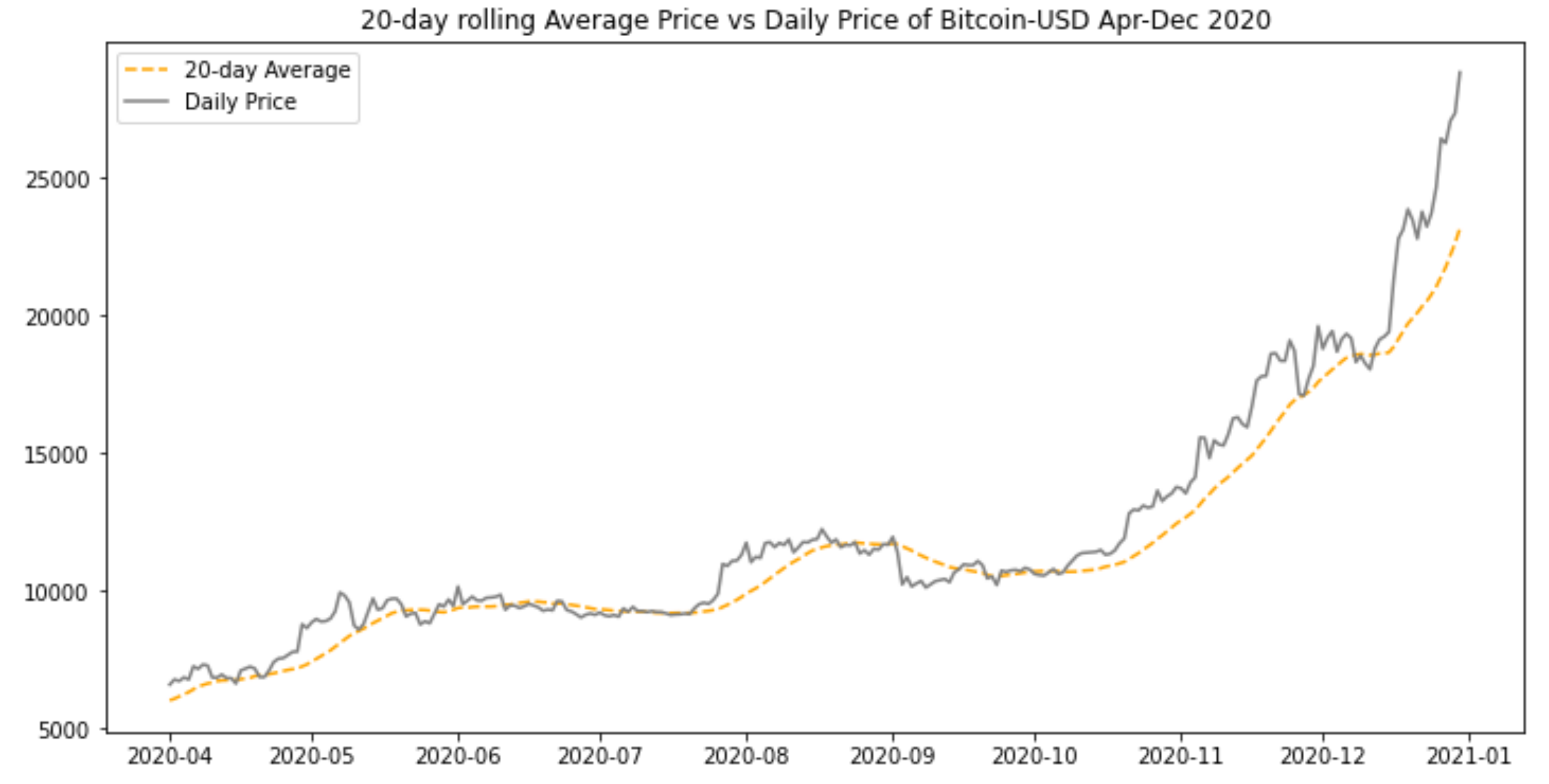


In [395]:
```
#20-day moving average price of your Equity ETF.
#df_equity_avg = df_equityetf['Close'].rolling(20).mean()
#df_equity_avg = df_equity_avg.loc['Apr-2020':'Dec-2020']
#df_equity_avg.plot(title = '20-day rolling Average Price of Equity ETF Apr-Dec 2020');

fig, ax = plt.subplots(figsize=(12,6))
ax.set_title('20-day rolling Average Price vs Daily Price of Equity ETF Apr-Dec 2020')
ax.plot(df_equity_avg, label = '20-day Average', linestyle = '--', color = 'orange')
ax.plot(df_equityetf['Close'].loc['Apr-2020':'Dec-2020'], label = 'Daily Price')
ax.legend()
plt.show()
```

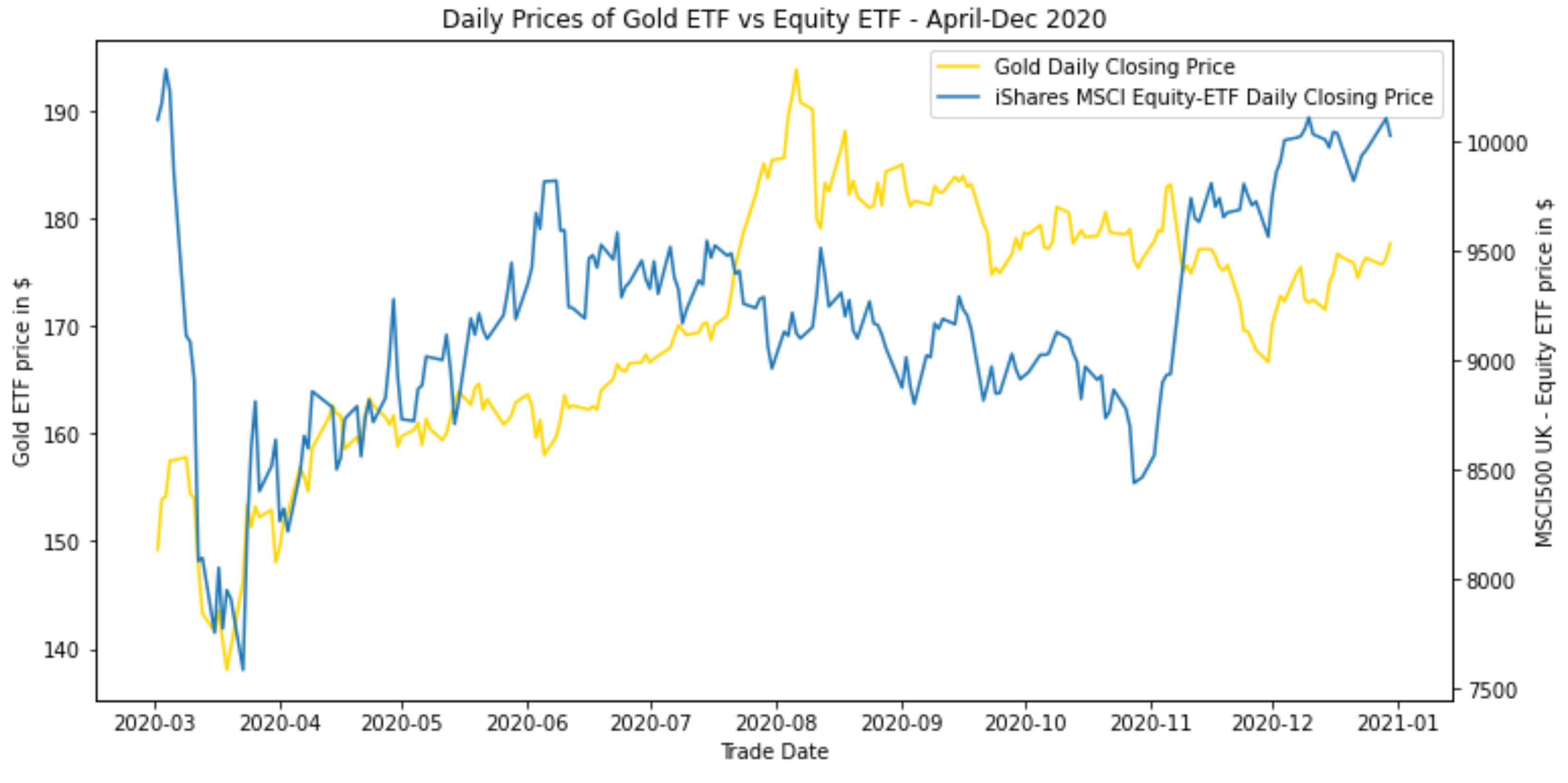


In [396]:
```
#20-day moving average price of your Equity ETF.
df_bitcoin_avg = df_bitcoin['Close'].rolling(20).mean()
df_bitcoin_avg = df_bitcoin_avg.loc['Apr-2020':'Dec-2020']
fig, ax = plt.subplots(figsize=(12,6))
ax.set_title('20-day rolling Average Price vs Daily Price of Bitcoin-USD Apr-Dec 2020')
ax.plot(df_bitcoin_avg, label = '20-day Average', linestyle = '--', color = 'orange')
ax.plot(df_bitcoin['Close'].loc['Apr-2020':'Dec-2020'], label = 'Daily Price', color = 'grey')
ax.legend()
plt.show()
#df_bitcoin_avg.plot(title = '20-day rolling Average Price of Bitcoin-USD Apr-Dec 2020');
```

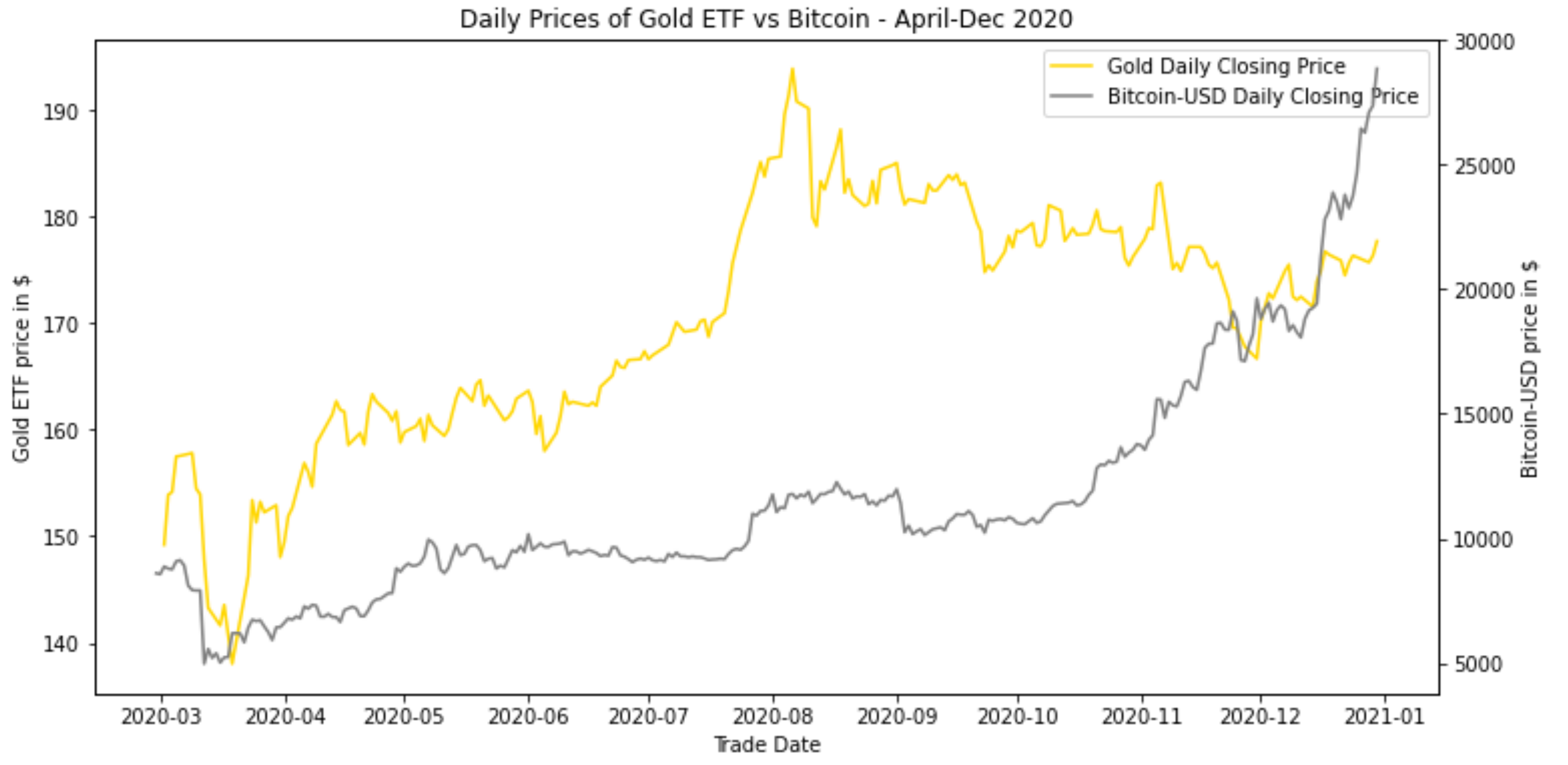


We notice that the Gold ETF and the UK MSCI Equity ETF are highly correlated and depict the same behavior, however at a different scale. We notice that the Bitcoin-USD is correlated to the remaining two pair, however not to a strong extent

In [397]:
```
#'Graph gold and equity prices on the same plot.  Use a separate scale for each series, and be sure to add a label and legend
fig, ax1 = plt.subplots(figsize=(12,6))
ax1.set_title("Daily Prices of Gold ETF vs Equity ETF - April-Dec 2020")
lns1 = ax1.plot(df_goldetf['Close'], color = 'gold', label = "Gold Daily Closing Price")
ax1.set_xlabel("Trade Date")
ax1.set_ylabel("Gold ETF price in $")
ax2 = ax1.twinx()
ax2.set_ylabel("MSCI500 UK - Equity ETF price in $")
lns2 = ax2.plot(df_equityetf['Close'], label = "iShares MSCI Equity-ETF Daily Closing Price")
lns = lns1+lns2
labs = [l.get_label() for l in lns]
ax1.legend(lns, labs)
plt.show()
```

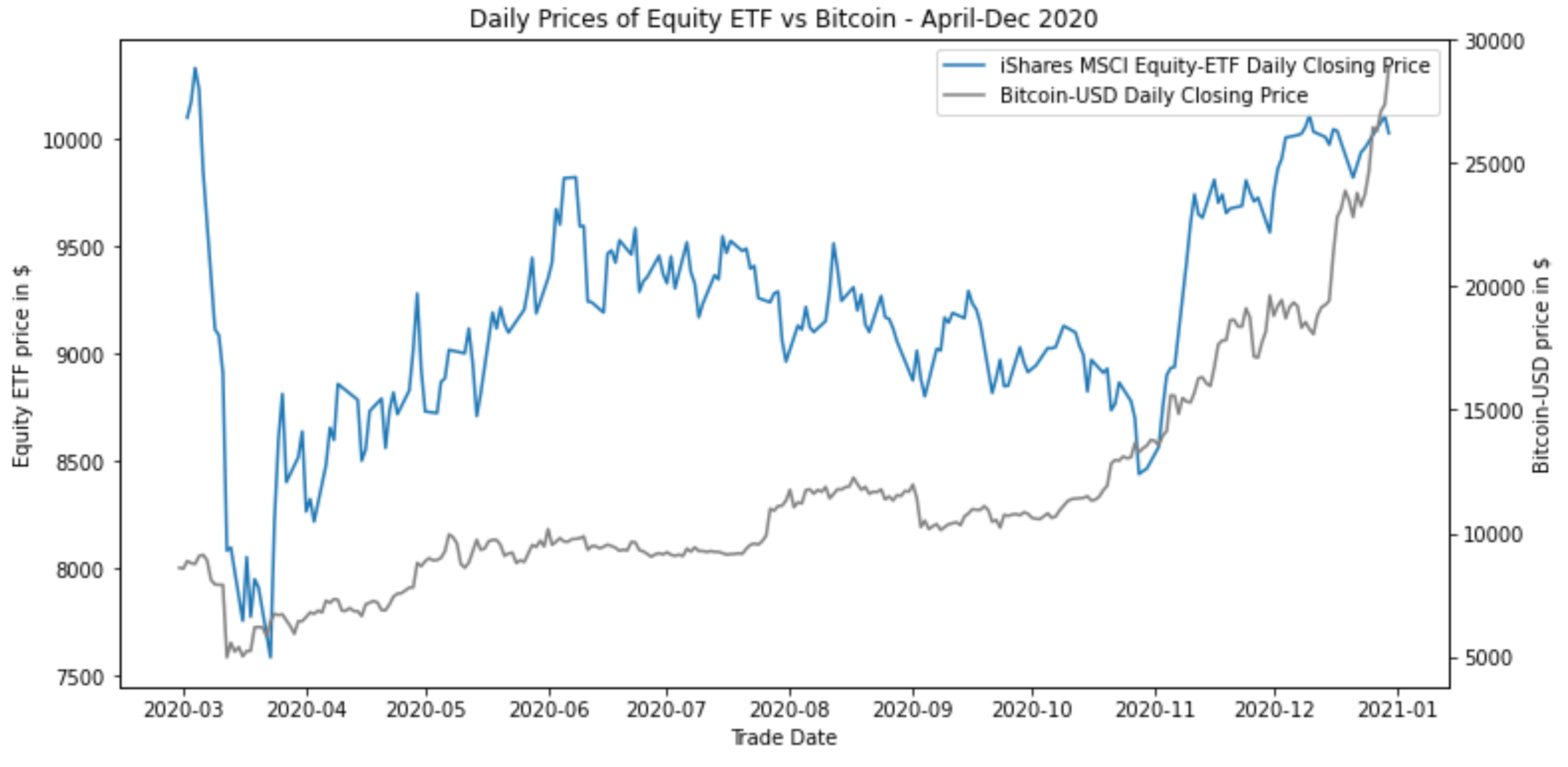


In [398]:
```
#Graph gold and bitcoin prices on the same plot.  Use a separate scale for each series, and be sure to add
#a label and legend
fig, ax1 = plt.subplots(figsize=(12,6))
ax1.set_title("Daily Prices of Gold ETF vs Bitcoin - April-Dec 2020")
lns1 = ax1.plot(df_goldetf['Close'], color = 'gold', label = "Gold Daily Closing Price")
ax1.set_xlabel("Trade Date")
ax1.set_ylabel("Gold ETF price in $")
ax2 = ax1.twinx()
ax2.set_ylabel("Bitcoin-USD price in $")
lns2 = ax2.plot(df_bitcoin['Close'], color = 'grey', label = "Bitcoin-USD Daily Closing Price")
lns = lns1+lns2
labs = [l.get_label() for l in lns]
ax1.legend(lns, labs)
plt.show()
```



In [399]:

```
#'Graph equity and bitcoin prices on the same plot.  Use a separate scale for each series,
#and be sure to add a label and legend
fig, ax1 = plt.subplots(figsize=(12,6))
ax1.set_title("Daily Prices of Equity ETF vs Bitcoin - April-Dec 2020")
lns1 = ax1.plot(df_equityetf['Close'], label = "iShares MSCI Equity-ETF Daily Closing Price")
ax1.set_xlabel("Trade Date")
ax1.set_ylabel("Equity ETF price in $")
ax2 = ax1.twinx()
ax2.set_ylabel("Bitcoin-USD price in $")
lns2 = ax2.plot(df_bitcoin['Close'], color = 'grey', label = "Bitcoin-USD Daily Closing Price")
lns = lns1+lns2
labs = [l.get_label() for l in lns]
ax1.legend(lns, labs)
plt.show()
```



Step 5. Fitting a GARCH Model

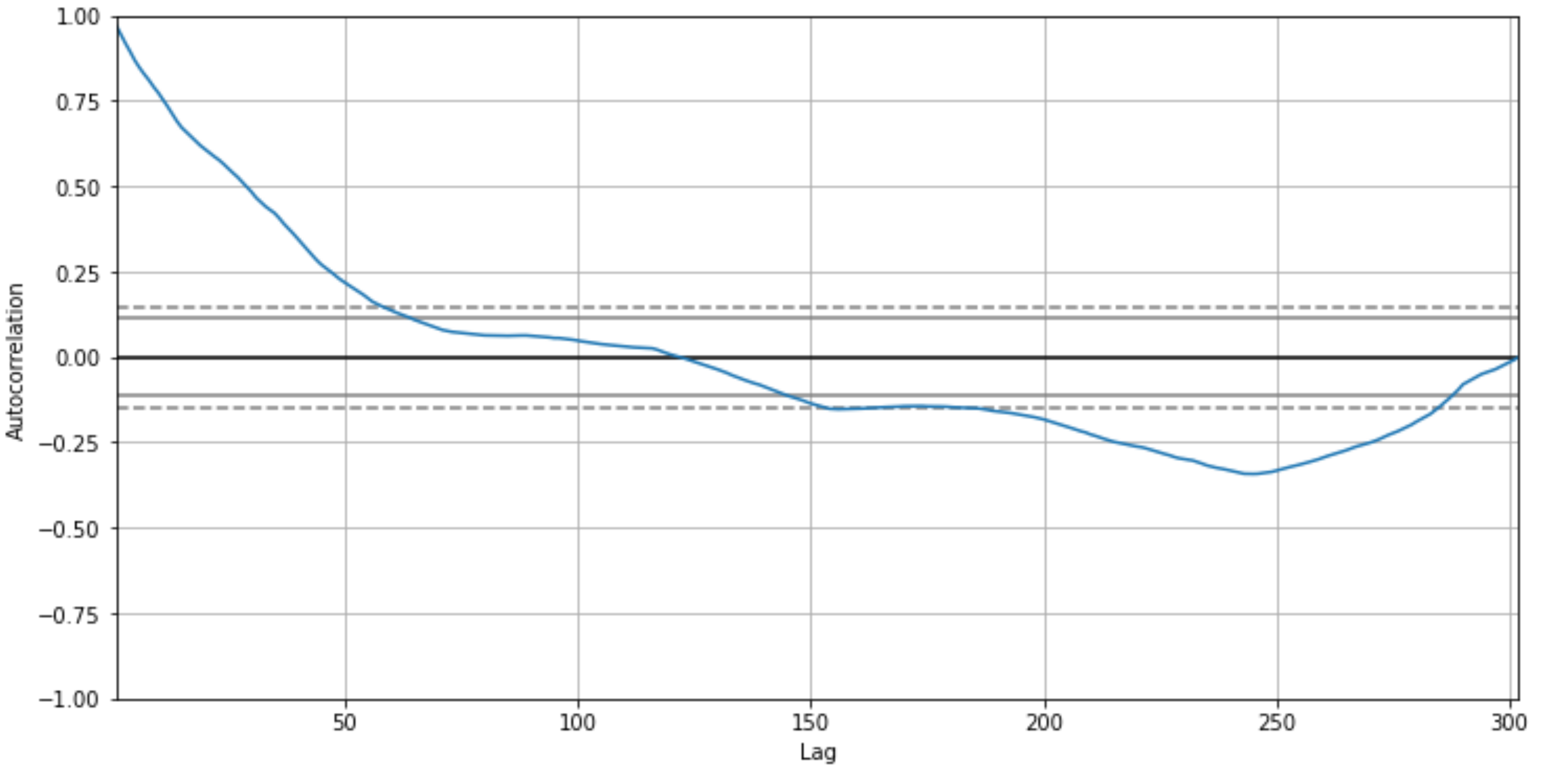
We will be looking into the bitcoin-USD for the purpose of fitting the GARCH model

The autocorrelation plot of bitcoin-USD daily closing price indicates that the time series for bicoin-USD daily prices is non-stationary

In [400]:

```
autocorrelation_plot(df_bitcoin['Close'])
adfuller(df_bitcoin['Close'])
```

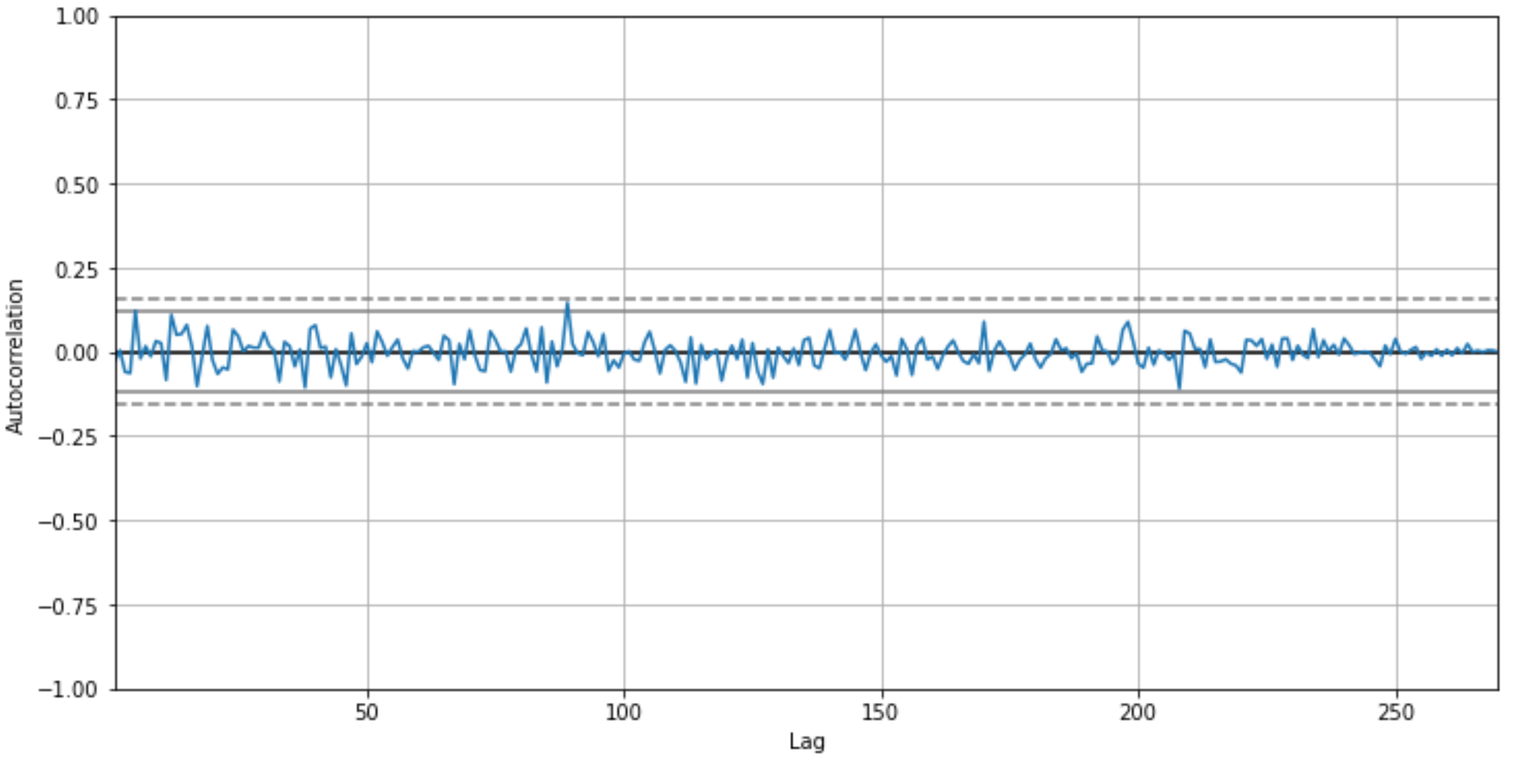
Out[400]:
```
(3.6495078180993246,
 1.0,
 0,
 301,
 {'1%': -3.452263435801039,
  '5%': -2.871190526189069,
  '10%': -2.571911967527952},
 4219.062168810124)
```



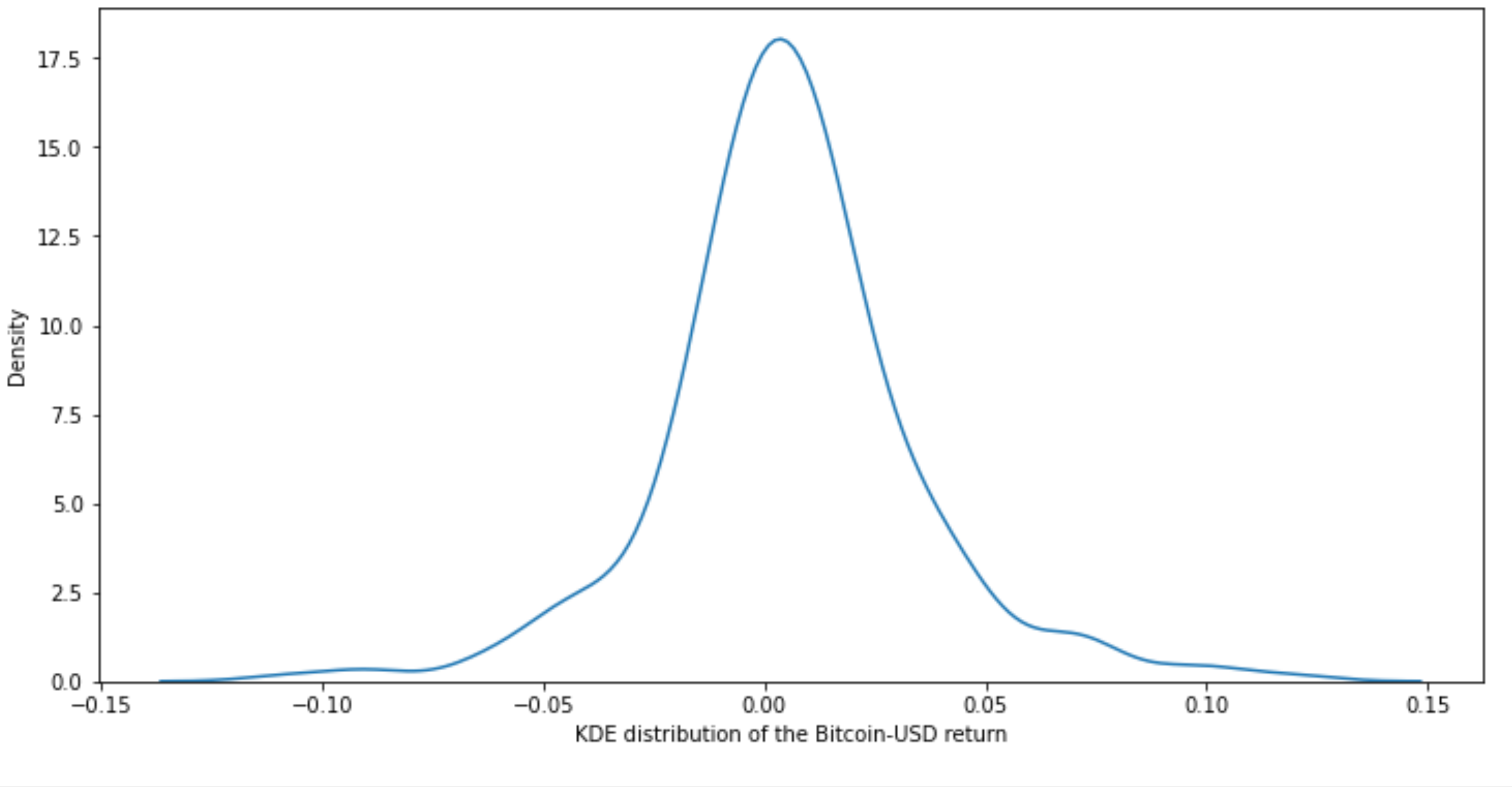
A review of the autocorrelation plot and the ADF test for the bitcoin daily returns reveals that the daily returns are stationary. Thus the daily returns are eligible for time series modelling

In [401]:
```
df_bitcoin_daily_return = df_bitcoin_daily_return.loc['Apr-2020':'Dec-2020']
autocorrelation_plot(df_bitcoin_daily_return)
adfuller(df_bitcoin_daily_return)
```

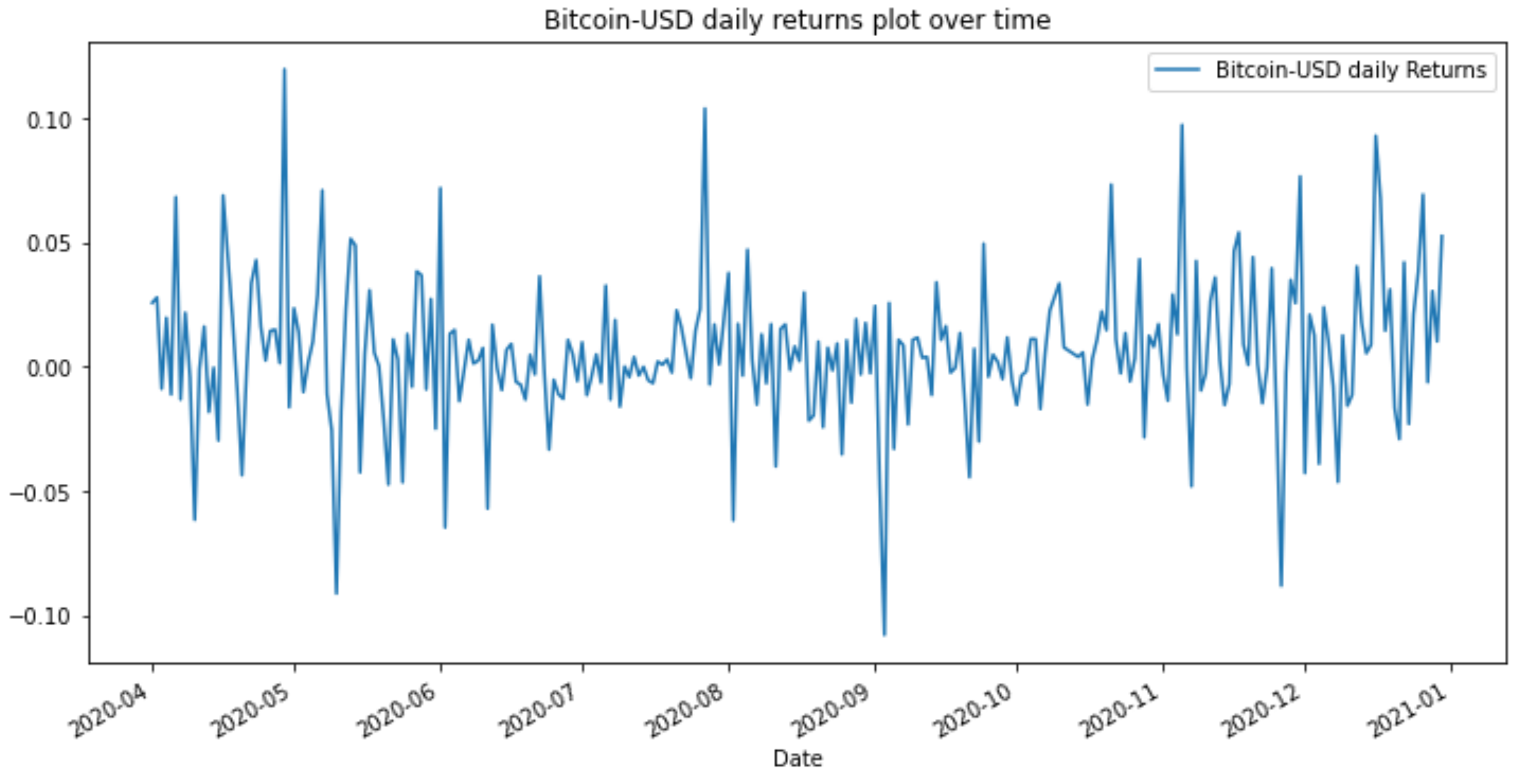
Out[401]:
```
(-16.725547127961725,
 1.3969539876176353e-29,
 0,
 269,
 {'1%': -3.4548957220044336,
  '5%': -2.8723451788613157,
  '10%': -2.572527778361272},
 -1069.567863101716)
```



In [402]:
```
sns.distplot(df_bitcoin_daily_return, hist=False, axlabel = 'KDE distribution of the Bitcoin-USD return');
```

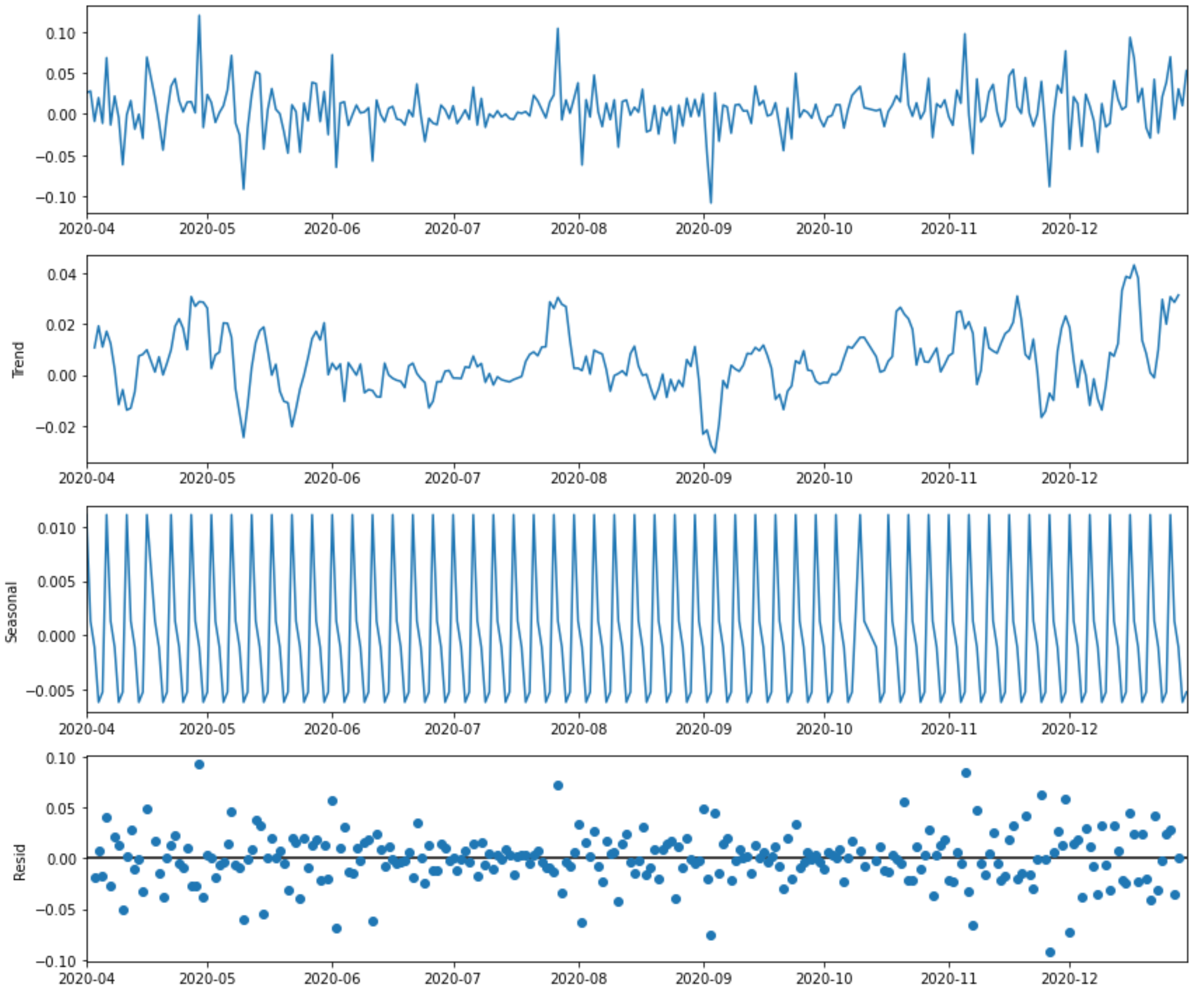


In [403]:
```
df_bitcoin_daily_return.plot(title = 'Bitcoin-USD daily returns plot over time', figsize=(12,6));
```

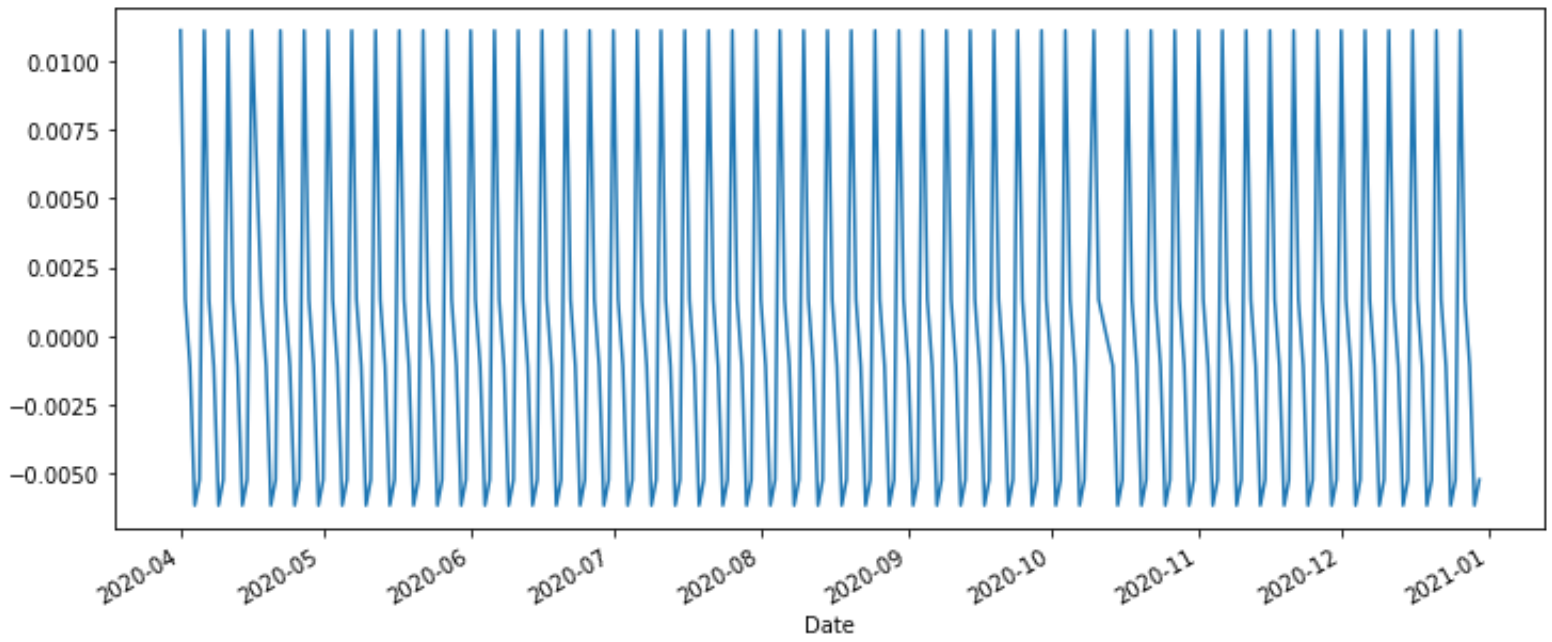


Identifying seasonality in the bitcoin daily return

In [404]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
from pylab import rcParams
rcParams['figure.figsize'] = 12,10
seasonal_decompose(df_bitcoin_daily_return, period = 5).plot();
```



In [405]:
```
#Zooming onto the seasonal component
rcParams['figure.figsize'] = 12,6
sd = seasonal_decompose(df_bitcoin_daily_return, period = 5)
sd.seasonal.plot(figsize = (12,5));
```



We notice that the seasonal component repeats itself about 4 - 4.5 times a month. Thus we use m = 5 as it captures the seasonal component

In [406]:
```
auto_arima(df_bitcoin_daily_return,trace=True, seasonal=True, m=5).summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(1,0,1)[5] intercept   : AIC=-1131.267, Time=0.85 sec
 ARIMA(0,0,0)(0,0,0)[5] intercept   : AIC=-1138.449, Time=0.07 sec
 ARIMA(1,0,0)(1,0,0)[5] intercept   : AIC=-1138.713, Time=0.35 sec
 ARIMA(0,0,1)(0,0,1)[5] intercept   : AIC=-1138.655, Time=0.23 sec
 ARIMA(0,0,0)(0,0,0)[5]             : AIC=-1130.836, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[5] intercept   : AIC=-1136.647, Time=0.08 sec
 ARIMA(1,0,0)(2,0,0)[5] intercept   : AIC=-1136.745, Time=0.61 sec
 ARIMA(1,0,0)(1,0,1)[5] intercept   : AIC=-1137.259, Time=0.49 sec
 ARIMA(1,0,0)(0,0,1)[5] intercept   : AIC=-1138.656, Time=0.16 sec
 ARIMA(1,0,0)(2,0,1)[5] intercept   : AIC=-1134.722, Time=0.60 sec
 ARIMA(0,0,0)(1,0,0)[5] intercept   : AIC=-1140.644, Time=0.15 sec
 ARIMA(0,0,0)(2,0,0)[5] intercept   : AIC=-1138.678, Time=0.23 sec
 ARIMA(0,0,0)(1,0,1)[5] intercept   : AIC=-1139.140, Time=0.19 sec
 ARIMA(0,0,0)(0,0,1)[5] intercept   : AIC=-1140.582, Time=0.14 sec
 ARIMA(0,0,0)(2,0,1)[5] intercept   : AIC=-1136.653, Time=0.34 sec
 ARIMA(0,0,1)(1,0,0)[5] intercept   : AIC=-1138.712, Time=0.54 sec
 ARIMA(1,0,1)(1,0,0)[5] intercept   : AIC=-1136.713, Time=0.43 sec
 ARIMA(0,0,0)(1,0,0)[5]             : AIC=-1135.097, Time=0.07 sec

Best model:  ARIMA(0,0,0)(1,0,0)[5] intercept
Total fit time: 5.607 seconds
```

Out[406]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	270
Model:	SARIMAX(1, 0, 0, 5)	**Log Likelihood**	573.322
Date:	Thu, 08 Jul 2021	**AIC**	-1140.644
Time:	23:23:17	**BIC**	-1129.849
Sample:	0	**HQIC**	-1136.309
	- 270		
Covariance Type:	opg		

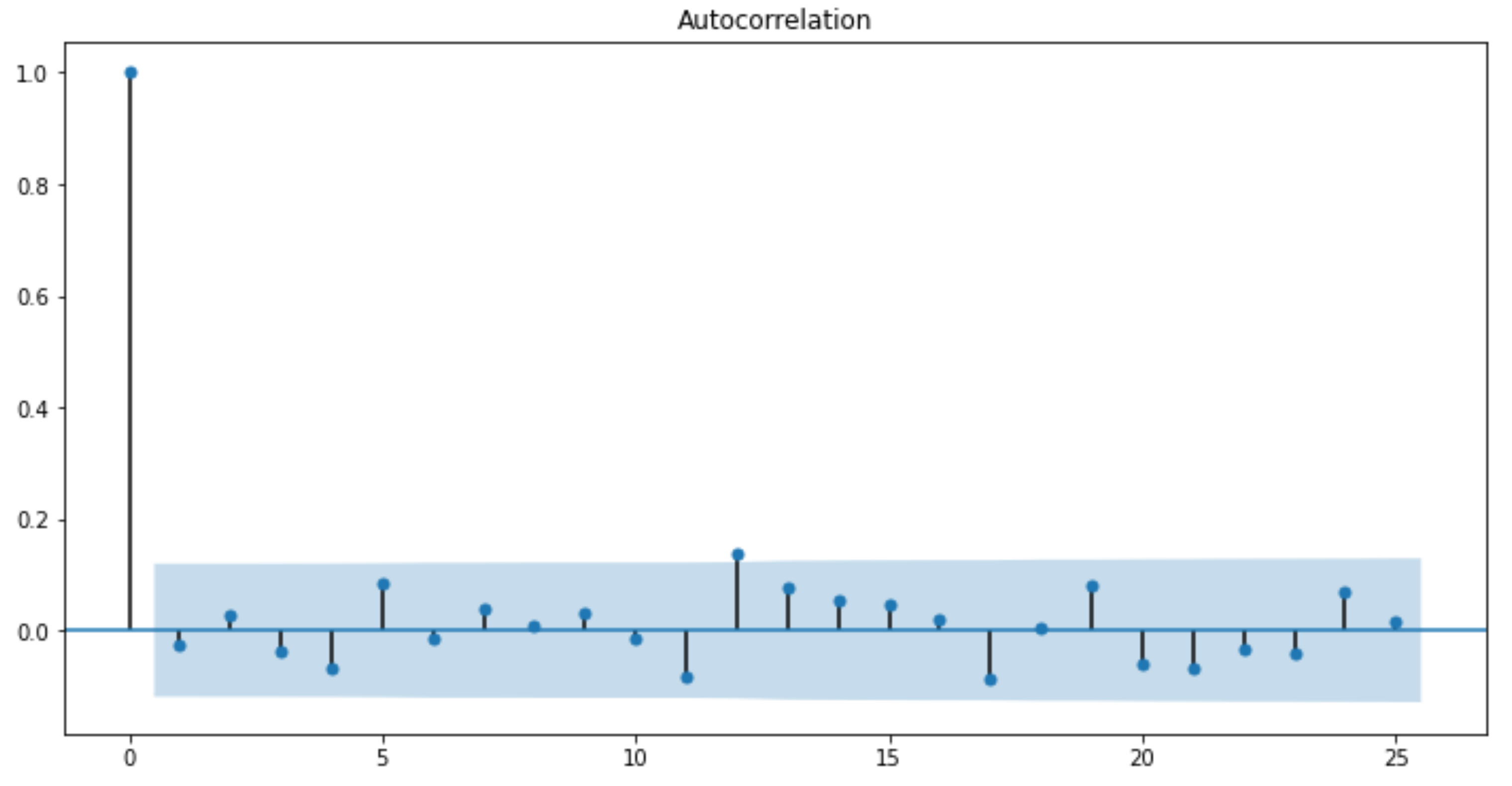
	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0049	0.002	2.714	0.007	0.001	0.008
ar.S.L5	0.1254	0.072	1.747	0.081	-0.015	0.266
sigma2	0.0008	4.71e-05	17.771	0.000	0.001	0.001

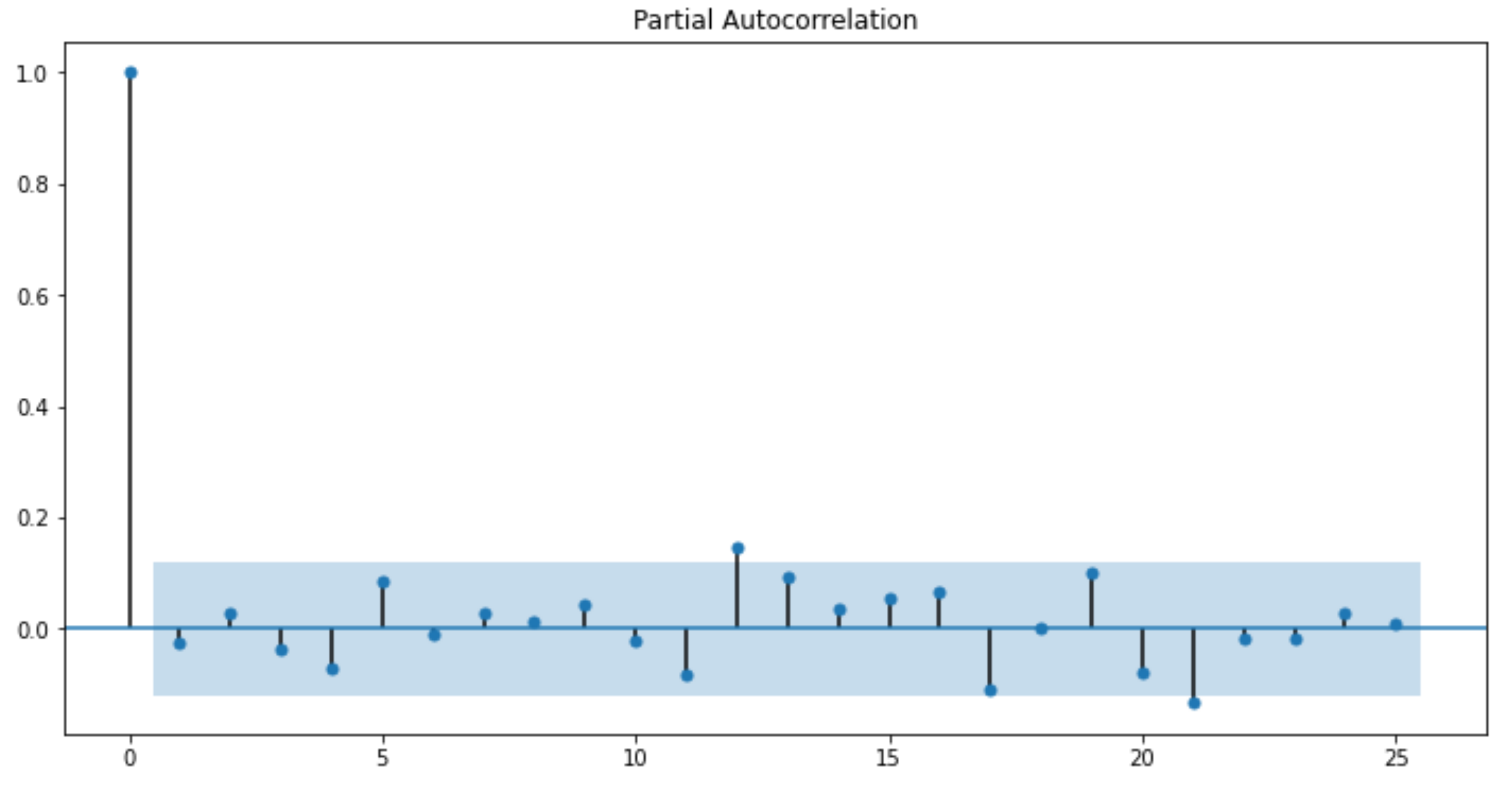
Ljung-Box (L1) (Q):	0.07	**Jarque-Bera (JB):**	83.72
Prob(Q):	0.79	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.00	**Skew:**	0.15
Prob(H) (two-sided):	1.00	**Kurtosis:**	5.71

Warnings:

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

In [407]:
```
#Plotting the ACF and PACF from the deseasonalized data
plot_acf(pd.Series(df_bitcoin_daily_return['Bitcoin-USD daily Returns'])-sd.seasonal);
plot_pacf(pd.Series(df_bitcoin_daily_return['Bitcoin-USD daily Returns'])-sd.seasonal);
```





In [408]:
```
sarimabitcoinreturnsfit = SARIMAX(df_bitcoin_daily_return,order=(0,0,0), seasonal_order=(1,0,0,5)).fit()
sarimabitcoinreturnsfit.summary()
#sns.distplot(sarimabitcoinreturnsfit.resid, hist=False);
```

Out[408]:

SARIMAX Results

Dep. Variable:	Bitcoin-USD daily Returns	**No. Observations:**	270
Model:	SARIMAX(1, 0, 0, 5)	**Log Likelihood**	569.548
Date:	Thu, 08 Jul 2021	**AIC**	-1135.097
Time:	23:23:18	**BIC**	-1127.900
Sample:	0	**HQIC**	-1132.207
	- 270		
Covariance Type:	opg		

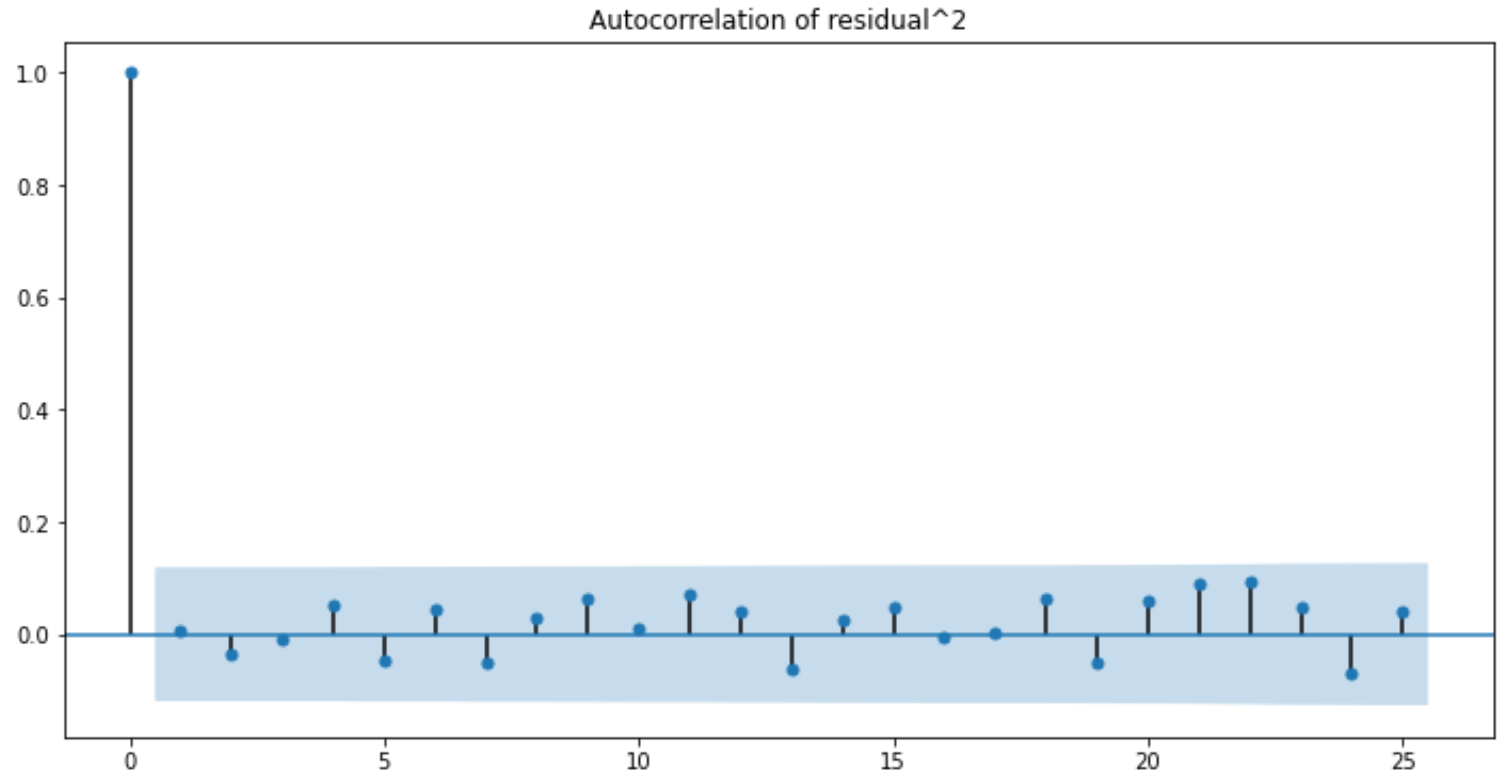
	coef	std err	z	P>|z|	[0.025	0.975]
ar.S.L5	0.1534	0.073	2.098	0.036	0.010	0.297
sigma2	0.0009	4.88e-05	17.633	0.000	0.001	0.001

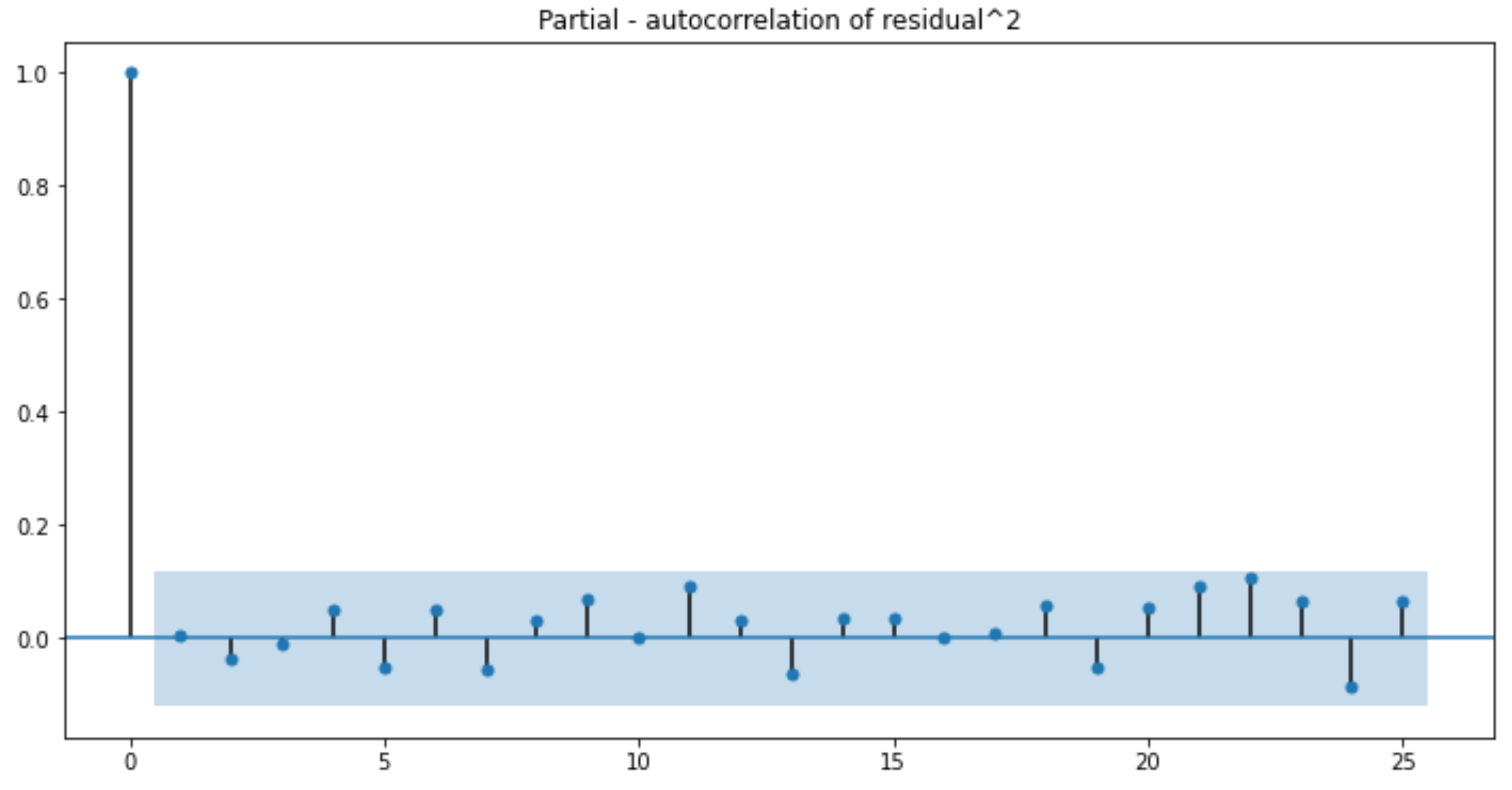
Ljung-Box (L1) (Q):	0.05	**Jarque-Bera (JB):**	82.61
Prob(Q):	0.83	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.07	**Skew:**	0.15
Prob(H) (two-sided):	0.75	**Kurtosis:**	5.69

Warnings:

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

In [409]:
```
#Plotting the square of the residuals
plot_acf(sarimabitcoinreturnsfit.resid**2, title='Autocorrelation of residual^2');
plot_pacf(sarimabitcoinreturnsfit.resid**2, title='Partial - autocorrelation of residual^2');
```





In [410]:
```
sm.stats.acorr_ljungbox(sarimabitcoinreturnsfit.resid**2, lags=15, return_df=True)
```

Out[410]:

	lb_stat	lb_pvalue
1	0.005213	0.942440
2	0.349377	0.839718
3	0.369237	0.946522
4	1.058737	0.900763
5	1.719450	0.886440
6	2.295748	0.890590
7	2.966036	0.888126
8	3.208030	0.920632
9	4.343602	0.887378
10	4.365854	0.929337
11	5.768803	0.888336
12	6.248059	0.903065
13	7.416230	0.879448
14	7.588309	0.909654
15	8.248763	0.913422

GARCH model

We notice that the GARCH(1,1) model works best with a decent AIC score and omega, alpha, beta values having significant p-values

In [411]:
```
garchbitcoinreturnfit = arch.arch_model(sarimabitcoinreturnsfit.resid, p=1, q=1, vol='Garch', mean='constant').fit()
garchbitcoinreturnfit.summary()
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 2130158748803158.5
Iteration:      2,   Func. Count:     17,   Neg. LLF: 2317713091.2711616
Iteration:      3,   Func. Count:     25,   Neg. LLF: 1624.059636418288
Iteration:      4,   Func. Count:     34,   Neg. LLF: 1128.7561496648525
Iteration:      5,   Func. Count:     43,   Neg. LLF: -562.9820412865863
Iteration:      6,   Func. Count:     51,   Neg. LLF: -573.8427846663164
Optimization terminated successfully    (Exit mode 0)
            Current function value: -573.8427863111627
            Iterations: 10
            Function evaluations: 51
            Gradient evaluations: 6
```

Out[411]:

Constant Mean - GARCH Model Results

Dep. Variable:	None	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	573.843
Distribution:	Normal	**AIC:**	-1139.69
Method:	Maximum Likelihood	**BIC:**	-1125.29
		No. Observations:	270
Date:	Thu, Jul 08 2021	**Df Residuals:**	266
Time:	23:23:18	**Df Model:**	4

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	4.4675e-03	2.008e-03	2.225	2.609e-02	[5.319e-04,8.403e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.9013e-05	2.356e-11	8.070e+05	0.000	[1.901e-05,1.901e-05]
alpha[1]	9.9833e-03	2.769e-02	0.360	0.718	[-4.429e-02,6.426e-02]
beta[1]	0.9676	2.584e-02	37.450	5.932e-307	[0.917, 1.018]

Covariance estimator: robust

GARCH unconditional variance

In [412]:

```python
garchbitcoinreturnfit.params[1]/(1-(garchbitcoinreturnfit.params[2]+garchbitcoinreturnfit.params[3]))
```

Out[412]: `0.0008472274986073406`

GARCH(1,1)-M Model for April-December 2020 BTC Log Returns

Dataframe of Mean Estimate from ARIMA and Variance Estimate from GARCH Model

In [413]:

```python
#BTC-USD Returns
btc_ticker = yf.Ticker('BTC-USD')
btc = btc_ticker.history(start = '2020-04-01', end = '2020-12-31')
btc_extended = btc_ticker.history(start = '2020-03-03', end = '2020-12-31')

btc_aprdec_logreturns = np.log(btc['Close'])[1:] - np.log(btc['Close'].shift(1))[1:]

#DataFrame of Log Returns and Variance
df_sarima_garch11 = pd.DataFrame(btc_aprdec_logreturns)
df_sarima_garch11 = df_sarima_garch11.rename(columns = {'Close':'Log Return'})

btc_extended_logreturns = np.log(btc_extended['Close'])[1:] - np.log(btc_extended['Close'].shift(1))[1:]
df_temp = pd.DataFrame(btc_extended_logreturns)
df_temp['std21'] = df_temp['Close'].rolling(21).std()
df_sarima_garch11['St Dev 21'] = df_temp['std21']
df_sarima_garch11['Ann Vol 21'] = df_sarima_garch11['St Dev 21']*(252**0.5)
df_sarima_garch11['Variance'] = df_sarima_garch11['Ann Vol 21']**2

#Adding the Mean Estimate from the SARIMA Model to the Data Frame
df_sarima_garch11['AR'] = ''
df_sarima_garch11['SARIMA Resid'] = sarimabitcoinreturnsfit.resid

for i in range(df_sarima_garch11.index.size):
    df_sarima_garch11['AR'].iloc[i] = sarimabitcoinreturnsfit.params['ar.S.L5']*df_sarima_garch11['Log Return'].iloc[i-1]

df_sarima_garch11['Mean Est'] = df_sarima_garch11['AR'] + df_sarima_garch11['SARIMA Resid']

#Adding the Volatility Estimate from the GARCH(1,1) Model to the Data Frame
df_sarima_garch11['SARIMA Sqrd Resid'] = df_sarima_garch11['SARIMA Resid']**2
df_sarima_garch11['Var Est'] = ''

for i in range(df_sarima_garch11.index.size):
    df_sarima_garch11['Var Est'].iloc[i] = garchbitcoinreturnfit.params['omega'] + (garchbitcoinreturnfit.params['alpha[1]']*df_sarima_garch11['SARIMA Sq

df_sarima_garch11['Vol Est'] = (df_sarima_garch11['Var Est']**0.5)*0.01
df_sarima_garch11
```

Out[413]:

	Log Return	St Dev 21	Ann Vol 21	Variance	AR	SARIMA Resid	Mean Est	SARIMA Sqrd Resid	Var Est	Vol Est
Date										
2020-04-01	0.025778	0.121636	1.930916	3.728437	0.00807173	0.025778	0.0338495	0.000664	0.263355	0.00513181
2020-04-02	0.027889	0.062294	0.988884	0.977891	0.00395384	0.027889	0.0318426	0.000778	3.60757	0.0189936
2020-04-03	-0.008906	0.058269	0.924990	0.855606	0.00427762	-0.008906	-0.00462866	0.000079	0.94621	0.00972733
2020-04-04	0.019726	0.055581	0.882321	0.778490	-0.00136606	0.019726	0.0183597	0.000389	0.827883	0.00909881
2020-04-05	-0.011187	0.055563	0.882038	0.777991	0.00302557	-0.011187	-0.00816129	0.000125	0.75327	0.00867911
...	...	...	...	...	...	...	...	...	...	...
2020-12-26	0.069389	0.035039	0.556231	0.309393	0.00588771	0.073858	0.0797461	0.005455	0.263509	0.00513331
2020-12-27	-0.006251	0.035341	0.561018	0.314742	0.010643	-0.012705	-0.00206174	0.000161	0.299435	0.00547207
2020-12-28	0.030458	0.035109	0.557344	0.310633	-0.000958786	0.033992	0.0330331	0.001155	0.304557	0.00551867
2020-12-29	0.010198	0.032088	0.509378	0.259466	0.0046717	0.006968	0.01164	0.000049	0.300591	0.00548262
2020-12-30	0.052625	0.032862	0.521669	0.272138	0.00156421	0.046737	0.0483017	0.002184	0.251072	0.00501071

270 rows × 10 columns

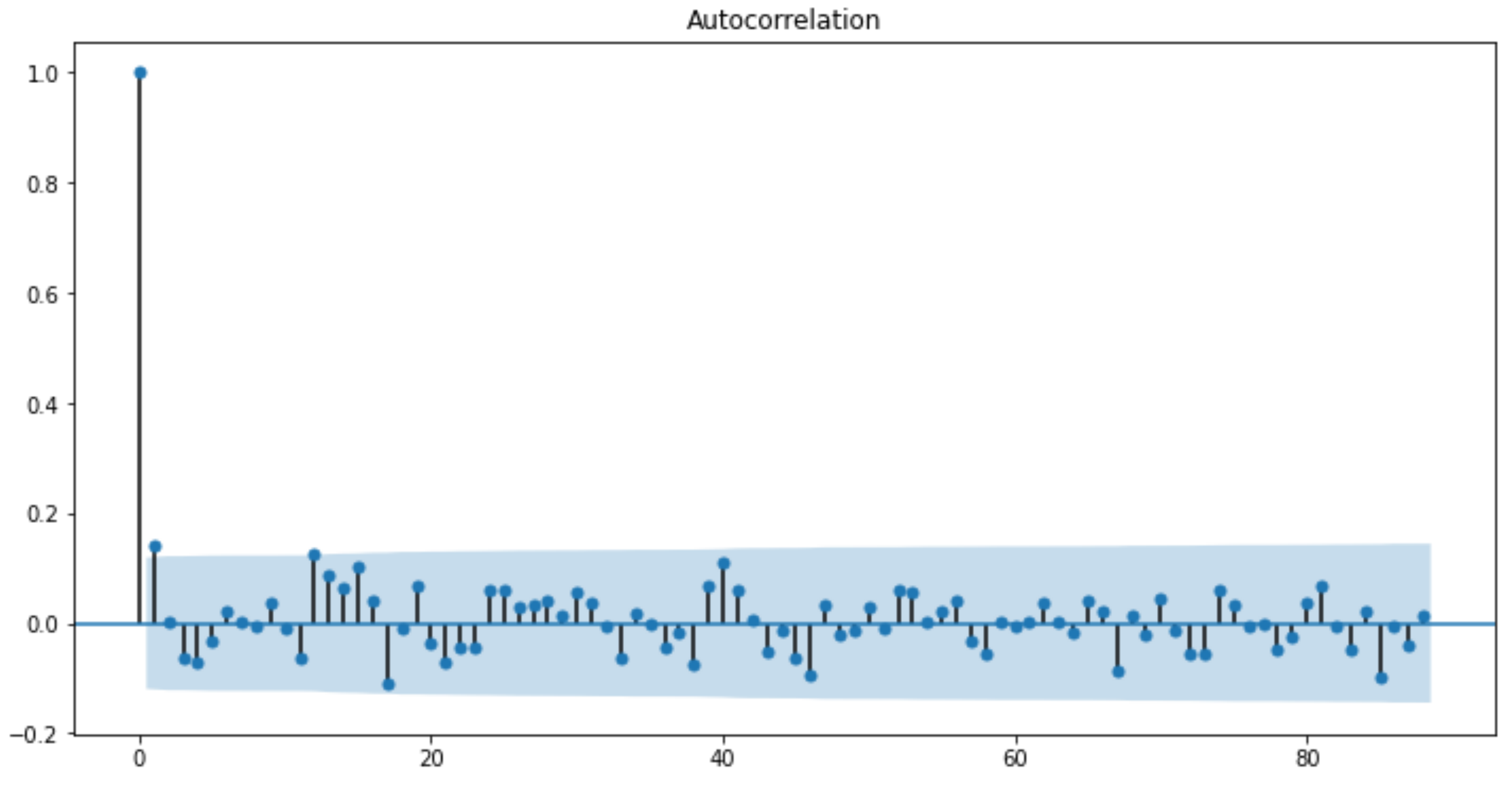
Process Obtained by Incorporating GARCH in the Mean Process

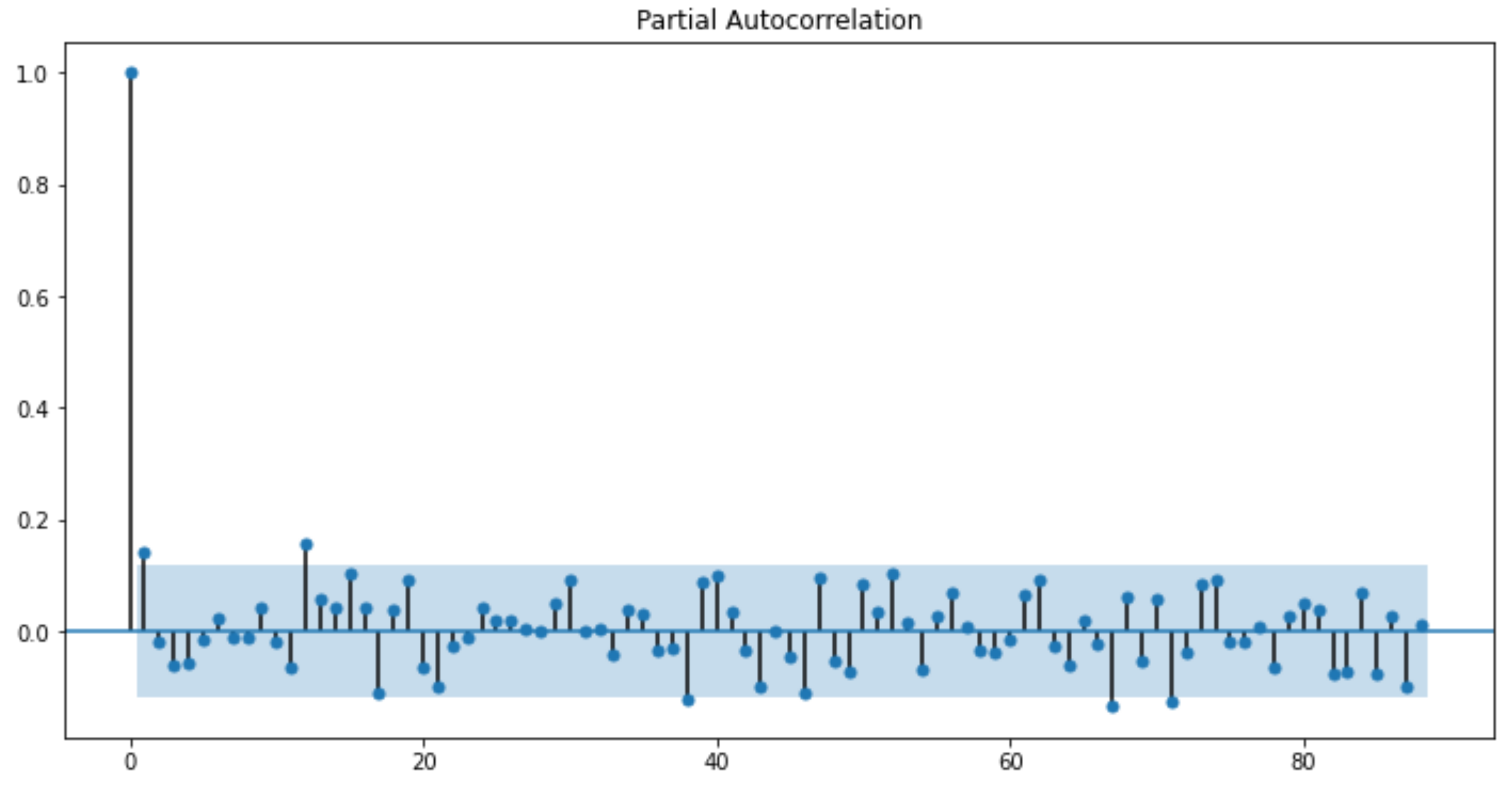
In [414]:
```
estprocess= df_sarima_garch11['Mean Est'] + df_sarima_garch11['Vol Est']
estprocess
```

Out[414]:
```
Date
2020-04-01      0.0389813
2020-04-02      0.0508362
2020-04-03     0.00509867
2020-04-04      0.0274586
2020-04-05    0.000517828
                  ...
2020-12-26      0.0848795
2020-12-27     0.00341033
2020-12-28      0.0385517
2020-12-29      0.0171226
2020-12-30      0.0533124
Length: 270, dtype: object
```

ARIMA Model on the New Process

In [415]:
```
plot_acf(estprocess,lags = 88);
plot_pacf(estprocess,lags = 88);
```

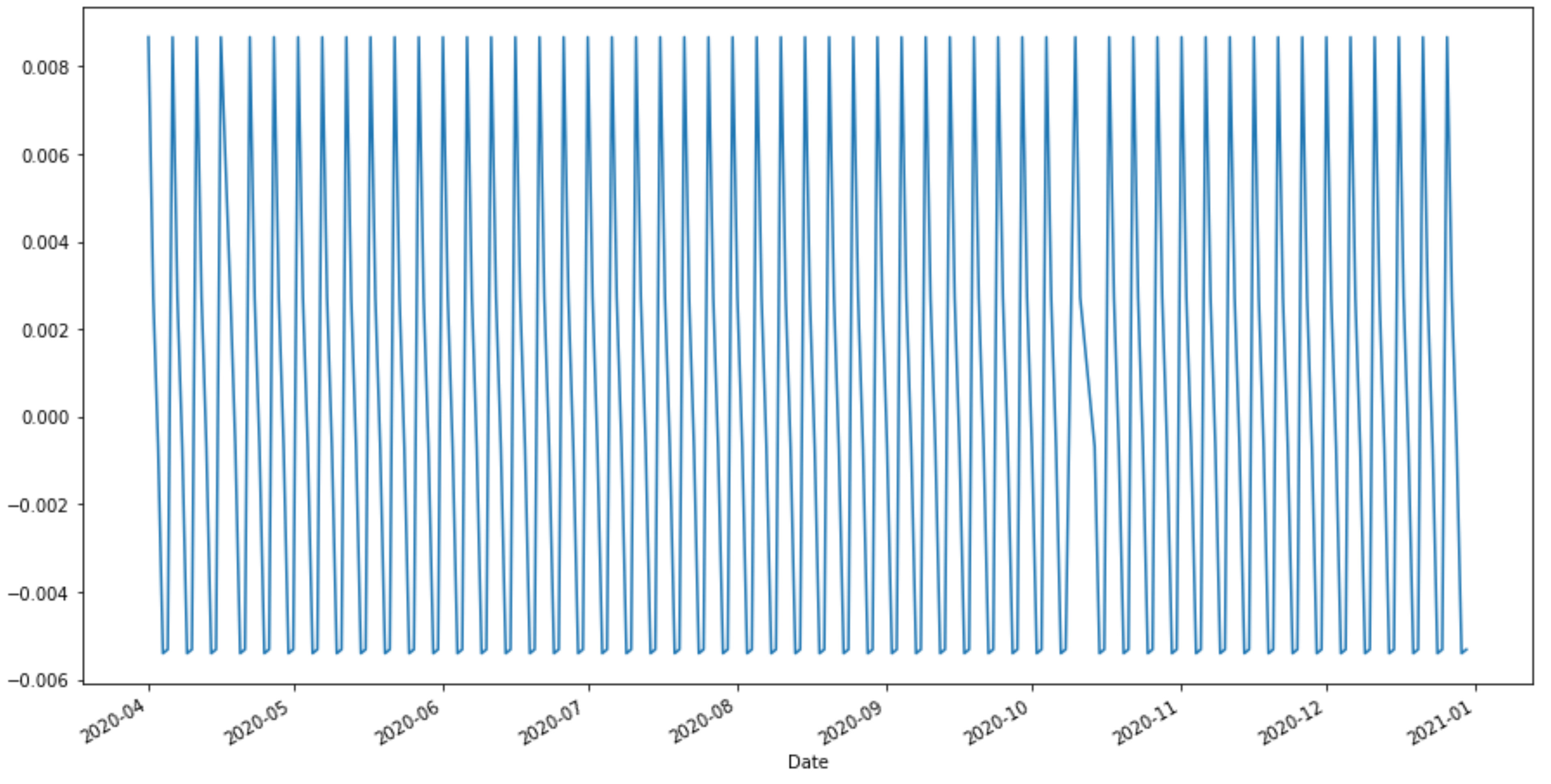




In [416]:
```
seasonal_decompose(estprocess, period = 5).plot();
```

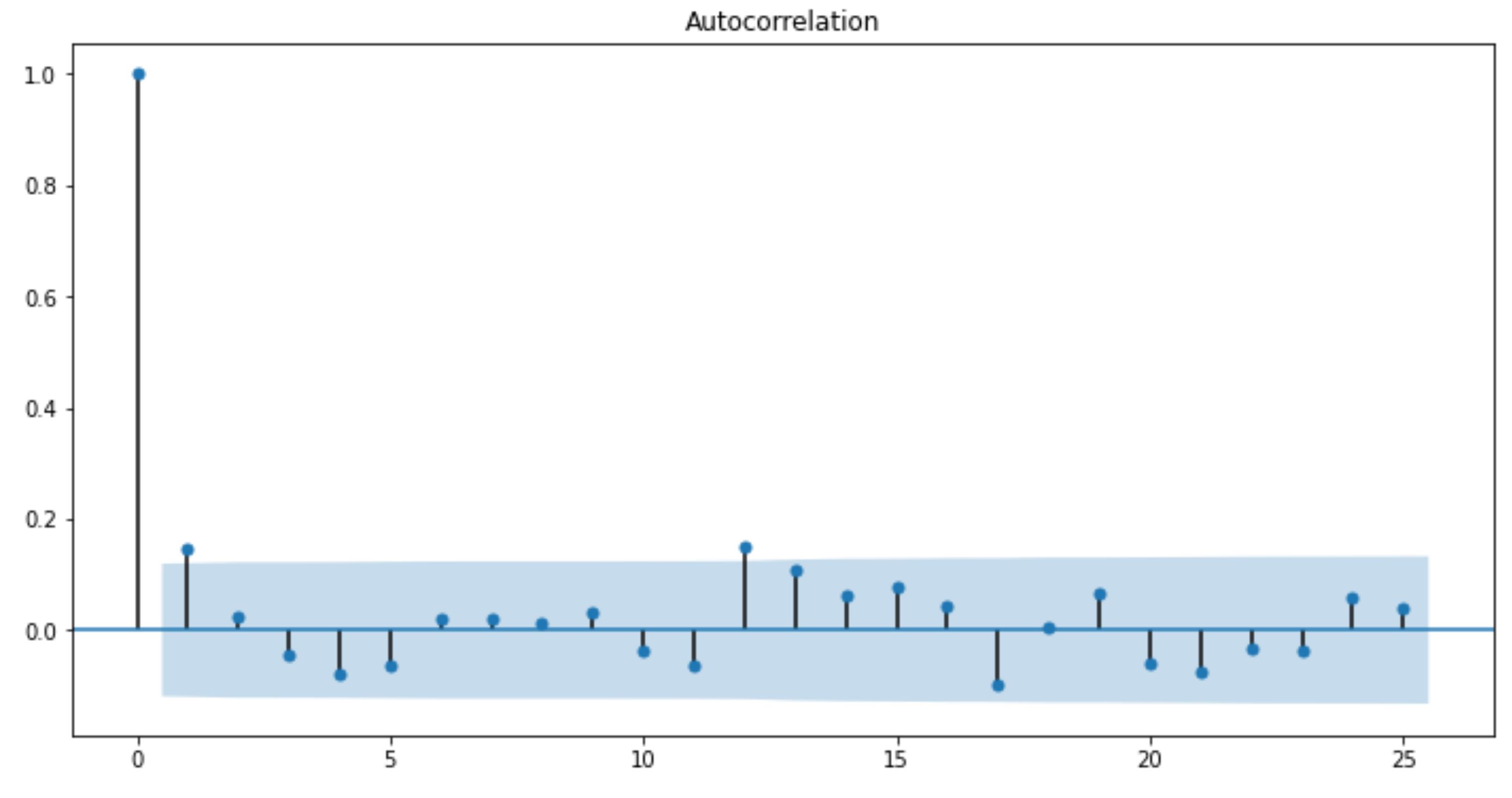
In [417]:

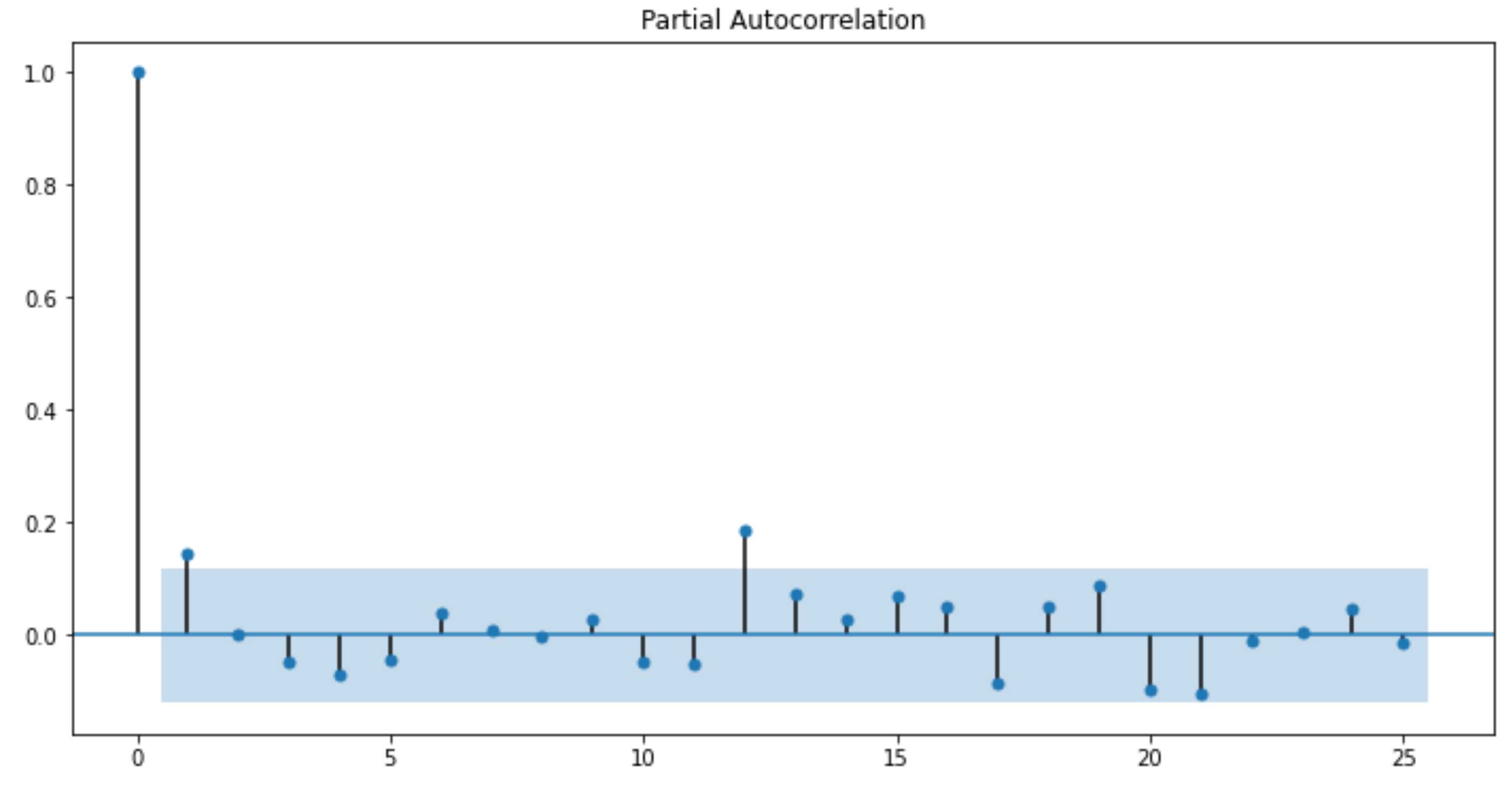
```
rcParams['figure.figsize'] = 12,6
sd = seasonal_decompose(estprocess, period = 5)
sd.seasonal.plot(figsize = (15,8));
```



In [418]:

```
#Plotting the ACF and PACF from the deseasonalized data
plot_acf(pd.Series(estprocess)-sd.seasonal);
plot_pacf(pd.Series(estprocess)-sd.seasonal);
```





In [419]:

```
auto_arima(estprocess, trace=True, seasonal=True, m=5).summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(1,0,1)[5] intercept   : AIC=-1129.515, Time=1.04 sec
 ARIMA(0,0,0)(0,0,0)[5] intercept   : AIC=-1134.830, Time=0.06 sec
 ARIMA(1,0,0)(1,0,0)[5] intercept   : AIC=-1136.618, Time=0.49 sec
 ARIMA(0,0,1)(0,0,1)[5] intercept   : AIC=-1136.621, Time=0.20 sec
 ARIMA(0,0,0)(0,0,0)[5]             : AIC=-1106.081, Time=0.04 sec
 ARIMA(0,0,1)(0,0,0)[5] intercept   : AIC=-1138.428, Time=0.15 sec
 ARIMA(0,0,1)(1,0,0)[5] intercept   : AIC=-1136.617, Time=0.22 sec
 ARIMA(0,0,1)(1,0,1)[5] intercept   : AIC=-1135.105, Time=0.88 sec
 ARIMA(1,0,1)(0,0,0)[5] intercept   : AIC=-1136.444, Time=0.20 sec
 ARIMA(0,0,2)(0,0,0)[5] intercept   : AIC=-1136.528, Time=0.16 sec
 ARIMA(1,0,0)(0,0,0)[5] intercept   : AIC=-1138.442, Time=0.06 sec
 ARIMA(1,0,0)(0,0,1)[5] intercept   : AIC=-1136.619, Time=0.24 sec
 ARIMA(1,0,0)(1,0,1)[5] intercept   : AIC=-1134.233, Time=0.49 sec
 ARIMA(2,0,0)(0,0,0)[5] intercept   : AIC=-1136.514, Time=0.11 sec
 ARIMA(2,0,1)(0,0,0)[5] intercept   : AIC=-1134.486, Time=0.27 sec
 ARIMA(1,0,0)(0,0,0)[5]             : AIC=-1119.210, Time=0.04 sec

Best model:  ARIMA(1,0,0)(0,0,0)[5] intercept
Total fit time: 4.688 seconds
```

Out[419]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	270
Model:	SARIMAX(1, 0, 0)	**Log Likelihood**	572.221
Date:	Thu, 08 Jul 2021	**AIC**	-1138.442
Time:	23:23:27	**BIC**	-1127.647
Sample:	0	**HQIC**	-1134.107
	- 270		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0088	0.002	4.682	0.000	0.005	0.012
ar.L1	0.1440	0.065	2.229	0.026	0.017	0.271
sigma2	0.0008	5.16e-05	16.375	0.000	0.001	0.001

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	77.99
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.99	**Skew:**	0.17
Prob(H) (two-sided):	0.97	**Kurtosis:**	5.61

Warnings:

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

In [420]:

```
estprocess_sarima = SARIMAX(estprocess.astype(float),order=(0,0,0), seasonal_order=(1,0,0,5))
estprocess_sarima = estprocess_sarima.fit()
estprocess_sarima.summary()
```

Out[420]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	270
Model:	SARIMAX(1, 0, 0, 5)	**Log Likelihood**	554.690
Date:	Thu, 08 Jul 2021	**AIC**	-1105.380
Time:	23:23:27	**BIC**	-1098.183
Sample:	0	**HQIC**	-1102.490
	- 270		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.S.L5	0.0709	0.069	1.031	0.302	-0.064	0.206
sigma2	0.0010	5.57e-05	17.251	0.000	0.001	0.001

Ljung-Box (L1) (Q):	5.92	**Jarque-Bera (JB):**	92.06
Prob(Q):	0.01	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.14	**Skew:**	0.14
Prob(H) (two-sided):	0.54	**Kurtosis:**	5.85

Warnings:

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

GARCH(1,1) Model on the New Process

In [421]:

```
estprocess_garch_11 = arch.arch_model(estprocess_sarima.resid, vol='GARCH', p=1, q=1)
estprocess_garch_11 = estprocess_garch_11.fit()
estprocess_garch_11.summary()
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 2510730.3844355354
Iteration:      2,   Func. Count:     16,   Neg. LLF: -569.1531373611193
Optimization terminated successfully    (Exit mode 0)
            Current function value: -569.1531392070641
            Iterations: 6
            Function evaluations: 16
            Gradient evaluations: 2
```

Out[421]:

Constant Mean - GARCH Model Results

Dep. Variable:	None	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	569.153
Distribution:	Normal	**AIC:**	-1130.31
Method:	Maximum Likelihood	**BIC:**	-1115.91
		No. Observations:	270
Date:	Thu, Jul 08 2021	**Df Residuals:**	266
Time:	23:23:27	**Df Model:**	4

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	8.9914e-03	2.165e-03	4.153	3.284e-05	[4.748e-03,1.324e-02]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.7448e-05	1.945e-11	8.968e+05	0.000	[1.745e-05,1.745e-05]
alpha[1]	1.0000e-02	2.338e-02	0.428	0.669	[-3.582e-02,5.582e-02]
beta[1]	0.9700	2.216e-02	43.768	0.000	[0.927, 1.013]

Covariance estimator: robust

Unconditional Variance for GARCH(1,1)-M

In [422]:
```
estprocess_garch_11.params[1]/(1-(estprocess_garch_11.params[2]+estprocess_garch_11.params[3]))
```

Out[422]: 0.0008722523564756791

EGARCH

We implement a symettrical EGARCH instead of standard EGARCH due to failure of convergence

In [423]:
```
egarchbitcoinreturnfit = arch.arch_model(sarimabitcoinreturnsfit.resid, p=1, q=1, o=0, vol='EGARCH').fit()
egarchbitcoinreturnfit.summary()
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 6312.464130764452
Iteration:      2,   Func. Count:     17,   Neg. LLF: -99.49681376064893
Iteration:      3,   Func. Count:     26,   Neg. LLF: 153163103.9093172
Iteration:      4,   Func. Count:     35,   Neg. LLF: -570.9144256865876
Iteration:      5,   Func. Count:     41,   Neg. LLF: -571.0226200126392
Iteration:      6,   Func. Count:     47,   Neg. LLF: -571.193015388999
Iteration:      7,   Func. Count:     53,   Neg. LLF: 87636923.71059847
Iteration:      8,   Func. Count:     59,   Neg. LLF: -571.2897646142441
Iteration:      9,   Func. Count:     65,   Neg. LLF: -572.6539688752918
Iteration:     10,   Func. Count:     71,   Neg. LLF: -575.0401146991694
Iteration:     11,   Func. Count:     77,   Neg. LLF: -575.1899876102225
Iteration:     12,   Func. Count:     82,   Neg. LLF: -575.1931425046045
Iteration:     13,   Func. Count:     87,   Neg. LLF: -575.1963486024905
Iteration:     14,   Func. Count:     92,   Neg. LLF: -575.1982534864077
Iteration:     15,   Func. Count:     97,   Neg. LLF: -575.1983273218382
Iteration:     16,   Func. Count:    102,   Neg. LLF: -575.1983357349532
Iteration:     17,   Func. Count:    107,   Neg. LLF: -575.1983360476484
Optimization terminated successfully    (Exit mode 0)
            Current function value: -575.1983360476484
            Iterations: 17
            Function evaluations: 107
            Gradient evaluations: 17
```

Out[423]:

Constant Mean - EGARCH Model Results

Dep. Variable:	None	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	EGARCH	**Log-Likelihood:**	575.198
Distribution:	Normal	**AIC:**	-1142.40
Method:	Maximum Likelihood	**BIC:**	-1128.00
		No. Observations:	270
Date:	Thu, Jul 08 2021	**Df Residuals:**	266
Time:	23:23:27	**Df Model:**	4

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	4.1899e-03	1.616e-03	2.593	9.526e-03	[1.022e-03,7.357e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	-0.2696	0.195	-1.380	0.168	[-0.652, 0.113]
alpha[1]	0.0517	5.969e-02	0.867	0.386	[-6.524e-02, 0.169]
beta[1]	0.9613	2.790e-02	34.461	3.044e-260	[0.907, 1.016]

Covariance estimator: robust

EGARCH unconditional variance

In [424]:
```
egarchbitcoinreturnfit.params[1]/(1-(egarchbitcoinreturnfit.params[2]+egarchbitcoinreturnfit.params[3]))
```

Out[424]: 20.64086489085204

TGARCH

We use Power = 1 and assume a normal distribution

In [425]:
```
tgarchbitcoinreturnfit = arch.arch_model(sarimabitcoinreturnsfit.resid**2, p=1, q=1, o=1, power=1).fit()
tgarchbitcoinreturnfit.summary()
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 2924138518.422355
Iteration:      2,   Func. Count:     21,   Neg. LLF: 205.01190372904765
Iteration:      3,   Func. Count:     30,   Neg. LLF: -1224.8248525453328
Iteration:      4,   Func. Count:     37,   Neg. LLF: -1227.6010952753861
Iteration:      5,   Func. Count:     44,   Neg. LLF: -1233.6397551497714
Iteration:      6,   Func. Count:     51,   Neg. LLF: 201450372.24855784
Iteration:      7,   Func. Count:     58,   Neg. LLF: 284263768194103.5
Iteration:      8,   Func. Count:     72,   Neg. LLF: 382040.5025065207
Iteration:      9,   Func. Count:     81,   Neg. LLF: -1182.4586071654048
Iteration:     10,   Func. Count:     88,   Neg. LLF: -1234.7757200034002
Iteration:     11,   Func. Count:     95,   Neg. LLF: 22403.527415876342
Optimization terminated successfully    (Exit mode 0)
            Current function value: -1315.3464295570911
            Iterations: 13
            Function evaluations: 103
            Gradient evaluations: 11
```

Out[425]:

Constant Mean - TARCH/ZARCH Model Results

Dep. Variable:	None	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	TARCH/ZARCH	**Log-Likelihood:**	1315.35
Distribution:	Normal	**AIC:**	-2620.69
Method:	Maximum Likelihood	**BIC:**	-2602.70
		No. Observations:	270
Date:	Thu, Jul 08 2021	**Df Residuals:**	265
Time:	23:23:28	**Df Model:**	5

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	8.6548e-04	3.924e-04	2.206	2.739e-02	[9.649e-05,1.634e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	5.0483e-04	6.035e-04	0.837	0.403	[-6.779e-04,1.688e-03]
alpha[1]	5.2955e-03	2.824e-02	0.188	0.851	[-5.005e-02,6.065e-02]
gamma[1]	3.6188e-03	0.856	4.225e-03	0.997	[-1.675, 1.682]
beta[1]	0.7256	0.144	5.050	4.415e-07	[0.444, 1.007]

Covariance estimator: robust

TGARCH unconditional variance

In [426]:
```
tgarchbitcoinreturnfit.params[1]/(1-(tgarchbitcoinreturnfit.params[2]+tgarchbitcoinreturnfit.params[3]))
```

Out[426]: 0.0005093746529800977

IGARCH

In [427]:
```
igarchbitcoinreturnfit = arch.arch_model(sarimabitcoinreturnsfit.resid**2, vol='FIGARCH', p =1, q=1,
                                          power=1, rescale=True).fit()
igarchbitcoinreturnfit.summary()
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 2253.4227699424277
Iteration:      2,   Func. Count:     17,   Neg. LLF: 1158.194814131491
Iteration:      3,   Func. Count:     25,   Neg. LLF: 559.6612934155214
Iteration:      4,   Func. Count:     31,   Neg. LLF: 560.7472278302471
Iteration:      5,   Func. Count:     38,   Neg. LLF: 552.9060646552722
Iteration:      6,   Func. Count:     44,   Neg. LLF: 550.4740209752246
Iteration:      7,   Func. Count:     50,   Neg. LLF: 550.2318591827047
Iteration:      8,   Func. Count:     56,   Neg. LLF: 550.0795128461546
Iteration:      9,   Func. Count:     62,   Neg. LLF: 550.0534902132371
Iteration:     10,   Func. Count:     68,   Neg. LLF: 550.0463620947446
Iteration:     11,   Func. Count:     74,   Neg. LLF: 550.0461693643997
Iteration:     12,   Func. Count:     80,   Neg. LLF: 550.0461655134559
Iteration:     13,   Func. Count:     86,   Neg. LLF: 550.0472500546598
Optimization terminated successfully    (Exit mode 0)
            Current function value: 550.0461655001502
            Iterations: 14
            Function evaluations: 89
            Gradient evaluations: 13
```

Out[427]:

Constant Mean - FIGARCH Model Results

Dep. Variable:	None	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	FIGARCH	**Log-Likelihood:**	-550.046
Distribution:	Normal	**AIC:**	1110.09
Method:	Maximum Likelihood	**BIC:**	1128.08
		No. Observations:	270
Date:	Thu, Jul 08 2021	**Df Residuals:**	265
Time:	23:23:28	**Df Model:**	5

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.8614	0.103	8.368	5.868e-17	[0.660, 1.063]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.7219	0.954	1.805	7.104e-02	[-0.148, 3.591]
phi	0.5000	0.159	3.147	1.651e-03	[0.189, 0.811]
d	0.0000	3.026e-02	0.000	1.000	[-5.931e-02,5.931e-02]
beta	0.5000	0.159	3.147	1.651e-03	[0.189, 0.811]

Covariance estimator: robust

Unconditional Variance for Integrated GARCH

In [428]:

```
igarchbitcoinreturnfit.params[1]/(1-(igarchbitcoinreturnfit.params[2]+igarchbitcoinreturnfit.params[3]))
```

Out[428]: 3.4436672261189902

Comparison of Long-term Variances from GARCH Models

In GARCH models, alpha and beta are two parameters which have the following economic interpretations

- Alpha is a reaction parameter. High α is generally associated with spiky or nervous market while low α indicates stable market. Across all models, GARCH-M has the highest alpha whereas TGARCH has the lowest alpha.
- Beta indicates volatility persistence (how much from past volatility is transferred into current volatility). High beta means high persistency and therefore volatility clustering appears. Beta for the GARCH and GARCH-M models is the highest whereas that reported by IGARCH is the lowest.

Step 6. Assessing Stationarity

In [430]:

```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Gold ETF
pvalgoldetf = round(adfuller(df_goldetf['Close'].loc['Apr-2020':'Dec-2020'])[1],5)
if (pvalgoldetf>0.05):
    print('The value of p is not significant with value being {} and \nhence we cannot reject the null hypothesis that a unit root exists and the data is
else:
    print('The value of p is significant with value being {} and \nhence we reject the null hypothesis\nThus, the unit root does not exists and the data


#Using KPSS Test for stationarity test
'''
H0: Data is stationary/n
H1: Data is not stationary
'''
#Testing the stationarity of the Gold ETF
kpsspvaluegoldetf = kpss(df_goldetf['Close'].loc['Apr-2020':'Dec-2020'])[1]
if (kpsspvaluegoldetf>0.05):
    print('The value of kpss p is not significant with value being {} and \nhence we cannot reject the null hypothesis. \nThus series is stationary'.form
else:
    print('The value of kpss p is significant with value being {} and \nhence we reject the null hypothesis\nThus series is not-stationary with a determi
         .format(kpsspvaluegoldetf))
```

```
The value of p is not significant with value being 0.12051 and
hence we cannot reject the null hypothesis that a unit root exists and the data is non-stationary
The value of kpss p is significant with value being 0.01 and
hence we reject the null hypothesis
Thus series is not-stationary with a deterministic trend
```

In [432]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Equity ETF
pvalequityetf = round(adfuller(df_equityetf['Close'].loc['Apr-2020':'Dec-2020'])[1],5)
if (pvalequityetf>0.05):
    print('The value of p is not significant with value being {} (>0.05) and \nhence we cannot reject the null hypothesis that a unit root exists and the
else:
    print('The value of p is significant with value being {} and \nhence we reject the null hypothesis\nThus, the unit root does not exists and the data

#Using KPSS Test for stationarity test
'''
H0: Data is stationary/n
H1: Data is not stationary
'''
#Testing the stationarity of the Equity ETF
kpsspvalueequityetf = kpss(df_equityetf['Close'].loc['Apr-2020':'Dec-2020'])[1]
if (kpsspvalueequityetf>0.05):
    print('The value of kpss p is not significant with value being {} and \nhence we cannot reject the null hypothesis. \nThus series is stationary with
else:
    print('The value of kpss p is significant with value being {} and \nhence we reject the null hypothesis\nThus series is non-stationary with a determi
```

```
The value of p is not significant with value being 0.56481 (>0.05) and
hence we cannot reject the null hypothesis that a unit root exists and the data is non-stationary
The value of kpss p is not significant with value being 0.08404460768024125 and
hence we cannot reject the null hypothesis.
Thus series is stationary with a deterministic trend
```

In [433]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Bitcoin-USD
pvalbitcoin = round(adfuller(df_bitcoin['Close'].loc['Apr-2020':'Dec-2020'])[1],5)
if (pvalbitcoin>0.05):
    print('The value of p is not significant with value being {} and \nhence we cannot reject the null hypothesis that a unit root exists and the data is
else:
    print('The value of p is significant with value being {} and \nhence we reject the null hypothesis\nThus, the unit root does not exists and the data


#Using KPSS Test for stationarity test
#Testing the stationarity of the Bitcoin -USD ETF
kpsspvaluebitcoin = kpss(df_bitcoin['Close'].loc['Apr-2020':'Dec-2020'])[1]
if (kpsspvaluebitcoin>0.05):
    print('The value of kpss p is not significant with value being {} and \nhence we cannot reject the null hypothesis. \nThus series is stationary with
else:
    print('The value of kpss p is significant with value being {} and \nhence we reject the null hypothesis\nThus series is not-stationary with a determi
```

```
The value of p is not significant with value being 1.0 and
hence we cannot reject the null hypothesis that a unit root exists and the data is non-stationary
The value of kpss p is significant with value being 0.01 and
hence we reject the null hypothesis
Thus series is not-stationary with a deterministic trend
```

Below we are performing a seasonality decomposition of all the asset types to identify trends which we can potentially use for ADF or KPSS tests

In [434]:
```
seasonal_decompose(df_bitcoin['Close'], period = 5).plot();
```

Performing ADF and KPSS Test on the daily prices of the 3 securities for Q2 2020

In [435]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Gold ETF
print ('ADF p-value for Gold ETF:',round(adfuller(df_goldetf['Close'].loc['Apr-2020':'Jun-2020'],
                                                  regression = 'ct')[1],5)) #Non-stationary
print ('KPSS p-value for Gold ETF:',round(kpss(df_goldetf['Close'].loc['Apr-2020':'Jun-2020'],
                                               regression = 'ct')[1],5)) #Stationary
```

```
ADF p-value for Gold ETF: 0.37066
KPSS p-value for Gold ETF: 0.1
```

In [436]:
```
#Using ADF Test to test the stationarity
#Using ADF Test to test the stationarity
#Testing the stationarity of the Gold ETF
print ('ADF p-value for Equity ETF:',round(adfuller(df_equityetf['Close'].loc['Apr-2020':'Jun-2020'],
                                                    regression = 'ct')[1],5)) #Non-stationary
print ('KPSS p-value for Equity ETF:',round(kpss(df_equityetf['Close'].loc['Apr-2020':'Jun-2020'],
                                                 regression = 'ct')[1],5)) #Stationary
```

```
ADF p-value for Equity ETF: 0.0335
KPSS p-value for Equity ETF: 0.1
```

In [437]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Gold ETF
print ('ADF p-value for Bitcoin USD ETF:',round(adfuller(df_bitcoin['Close'].loc['Apr-2020':'Jun-2020'],
                                                         regression = 'ct')[1],5)) #Non-stationary
print ('KPSS p-value for Bitcoin ETF:',round(kpss(df_bitcoin['Close'].loc['Apr-2020':'Jun-2020'],
                                                  regression = 'ct')[1],5)) #Non-stationary
```

```
ADF p-value for Bitcoin USD ETF: 0.70383
KPSS p-value for Bitcoin ETF: 0.02345
```

Step 7. Modelling Cointegration

In the cell immediately below we are performing a join so that we have the prices for the same days given that we had records for weekends for the Bitcoin-USD while we did not have data for the Equity and Gold ETF. Thus the engle granger analysis must be performed on the same set of records

In [438]:
```
dfb = pd.DataFrame(data = df_bitcoin['Close'])
dfb.columns = ['Bitcoin Closing Price']
dfe = pd.DataFrame(data = df_equityetf['Close'])
dfe.columns = ['Equity ETF Closing Price']
dfg = pd.DataFrame(data = df_goldetf['Close'])
dfg.columns = ['Gold ETF Closing Price']
df_close = dfg.join(dfb.join(dfe, how = 'inner'), how='inner')
df_close.head()
```

Out[438]:

Date	Gold ETF Closing Price	Bitcoin Closing Price	Equity ETF Closing Price
2020-03-02	149.199997	8869.669922	10100.0
2020-03-03	153.889999	8787.786133	10176.0
2020-03-04	154.160004	8755.246094	10332.0
2020-03-05	157.490005	9078.762695	10231.0
2020-03-06	157.550003	9122.545898	9854.0

In [439]:
```
#Since Equity ETF is stationary, we cannot perform an Engle-Granger test for Gold ETF with other securities
#Hence we can perform an engle granger of Bitcoin-USD with GoldETF where both have the differencing of 1

from arch.unitroot import engle_granger
egq2bitcoingold = engle_granger(df_close['Bitcoin Closing Price'].loc['Apr-2020':'Jun-2020'],
                                df_close['Gold ETF Closing Price'].loc['Apr-2020':'Jun-2020'], trend='ctt')
egq2bitcoingold
```

Out[439]:

Engle-Granger Cointegration Test

Test Statistic	-4.633
P-value	0.013
ADF Lag length	0
Estimated Root ρ (γ+1)	0.460

Trend: Constant

Critical Values: -4.09 (10%), -4.42 (5%), -5.10 (1%)

Null Hypothesis: No Cointegration

Alternative Hypothesis: Cointegration

Distribution Order: 1

Co-integration Vectors for Quadratic trend VECM - Bitcoin and Gold

The high volatility of bitcoin-usd secutity is offset by the gold etf which is less volatile and thus its has a higher co-efficient in the co-integration equation

In [440]:
```
egq2bitcoingold.cointegrating_vector
```

Out[440]:
```
Bitcoin Closing Price          1.000000
Gold ETF Closing Price        46.783984
const                     -13335.970341
trend                       -174.869517
quadratic_trend                1.937277
dtype: float64
```

Summary of finding based on analysis of Q2 data for Equity ETF, Gold ETF and Bitcoin USD

This is because of the following reasons:

1. Equity ETF is **trend stationary**. Thus, Gold ETF cannot be used for co-integration with either Equity ETF or Bitcoin-USD
2. The p-value of the co-integration test between Gold ETF and Bitcoin-USD is significant for a **quadratic trend**. Thus, we **reject the null hypothesis of non cointegration**

In [441]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Gold ETF
print ('ADF p-value for Gold ETF:',round(adfuller(df_goldetf['Close'].loc['Jul-2020':'Sep-2020'],
                                                  regression = 'ct')[1],5)) #Non-stationary
print ('KPSS p-value for Gold ETF:',round(kpss(df_goldetf['Close'].loc['Jul-2020':'Sep-2020'],
                                               regression = 'ct')[1],5)) #Non-Stationary
```

```
ADF p-value for Gold ETF: 0.8281
KPSS p-value for Gold ETF: 0.04086
```

In [442]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Equity ETF
print ('ADF p-value for Equity ETF:',round(adfuller(df_equityetf['Close'].loc['Jul-2020':'Sep-2020'],
                                                    regression = 'ct')[1],5)) #Non-stationary
print ('KPSS p-value for Equity ETF:',round(kpss(df_equityetf['Close'].loc['Jul-2020':'Sep-2020'],
                                                 regression = 'ct')[1],5)) #Non-Stationary
```

```
ADF p-value for Equity ETF: 0.01444
KPSS p-value for Equity ETF: 0.1
```

In [443]:
```
#Using ADF Test to test the stationarity
#Testing the stationarity of the Bitcoin ETF
print ('ADF p-value for Bitcoin USD ETF:',round(adfuller(df_bitcoin['Close'].loc['Jul-2020':'Sep-2020'],
                                                         regression = 'ct')[1],5)) #Non-stationary
print ('KPSS p-value for Bitcoin ETF:',round(kpss(df_bitcoin['Close'].loc['Jul-2020':'Sep-2020'],
                                                  regression = 'ct')[1],5)) #Non-stationary
```

```
ADF p-value for Bitcoin USD ETF: 0.83696
KPSS p-value for Bitcoin ETF: 0.02365
```

All the time series are non-stationary for Q3 and thus we can run the Engle Granger on all the pairs of value

Engle Granger for Bitcoin-USD and Gold ETF for Q3 2020

In [444]:
```
egq3bitcoingold = engle_granger(df_close['Bitcoin Closing Price'].loc['Jul-2020':'Sep-2020'],
                                df_close['Gold ETF Closing Price'].loc['Jul-2020':'Sep-2020'], trend='ct')
egq3bitcoingold
#The results depict no-cointegration
```

Out[444]:

Engle-Granger Cointegration Test	
Test Statistic	-2.481
P-value	0.534
ADF Lag length	0
Estimated Root ρ (γ+1)	0.818

Trend: Constant
Critical Values: -3.64 (10%), -3.97 (5%), -4.62 (1%)
Null Hypothesis: No Cointegration
Alternative Hypothesis: Cointegration
Distribution Order: 1

Engle Granger for Bitcoin-USD and Equity ETF for Q3 2020

In [445]:
```
#The results of Engle Granger test shows no-cointegration
egq3bitcoinequity = engle_granger(df_close['Bitcoin Closing Price'].loc['Jul-2020':'Sep-2020'],
                                  df_close['Equity ETF Closing Price'].loc['Jul-2020':'Sep-2020'], trend='c')
egq3bitcoinequity
```

Out[445]:

Engle-Granger Cointegration Test	
Test Statistic	-2.070
P-value	0.490
ADF Lag length	0
Estimated Root ρ (γ+1)	0.882

Trend: Constant
Critical Values: -3.14 (10%), -3.46 (5%), -4.11 (1%)
Null Hypothesis: No Cointegration
Alternative Hypothesis: Cointegration
Distribution Order: 1

Engle Granger for Gold ETF and Equity ETF for Q3 2020

In [446]:
```
egq3goldequity = engle_granger(df_close['Gold ETF Closing Price'].loc['Jul-2020':'Sep-2020'],
                               df_close['Equity ETF Closing Price'].loc['Jul-2020':'Sep-2020'], trend= 'n')
egq3goldequity
```

Out[446]:

Engle-Granger Cointegration Test	
Test Statistic	-2.060
P-value	0.214
ADF Lag length	0
Estimated Root ρ (γ+1)	0.901

Trend: Constant
Critical Values: -2.54 (10%), -2.87 (5%), -3.51 (1%)
Null Hypothesis: No Cointegration
Alternative Hypothesis: Cointegration
Distribution Order: 1

Q. 7.7 - If any 2 sets are cointegrated, Do the results from Quarter 2 cointegration testing help predict the coefficients from Quarter 3 cointegration?

We do not notice any cointegration in Q3 for any of the security pairs

The Q2 cointegration results did not help in any way to identify the Q3 cointegration results. As mentioned by the facuty, i will now run a VAR model on the **stationary time series of the asset returns**

VAR Model on Asset Returns

In [447]:
```
df_all_returns = df_bitcoin_daily_return.join(pd.DataFrame(df_equityetf_daily_return)
                                              .join(df_goldetf_daily_return, how='inner'), how='inner')
```

In [448]:
```
print('ADF test p-value for Bitcoin return: ',round(adfuller(df_all_returns['Bitcoin-USD daily Returns'])[1],7))
print('ADF test p-value for Equity return: ', round(adfuller(df_all_returns['Equity ETF Daily Returns'])[1],7))
print('ADF test p-value for Bitcoin return: ',round(adfuller(df_all_returns['Gold ETF Daily Returns'])[1],7))
```

```
ADF test p-value for Bitcoin return:  0.0
ADF test p-value for Equity return:  0.0
ADF test p-value for Bitcoin return:  0.0
```

Testing Vector Autoregressive Model for Q2

In [449]:

```
from statsmodels.tsa.api import VAR
aic = 1000
var_modelq2 = VAR(df_all_returns.loc['Apr-2020':'Jun-2020'])
for lag in range(15):
    score = var_modelq2.fit(lag).aic

    if(score<aic):
        aic = score
        p = lag

print('''The VAR model for the 3 dimensional timeseries has the best aic score as {}
      with lags of {}'''.format(round(aic,2),p))
```

```
The VAR model for the 3 dimensional timeseries has the best aic score as -29.84
      with lags of 14
```

The VAR model for the 3 dimensional timeseries has the best aic score as -30.31 with lags of 14 for Quarter 2

In [450]:

```
var_modelq2.fit(p).summary()
```

Out[450]:

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Thu, 08, Jul, 2021
Time:                     23:25:28
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                   -24.7134
Nobs:                     46.0000    HQIC:                  -27.9205
Log likelihood:           619.542    FPE:                1.04946e-11
AIC:                     -29.8415    Det(Omega_mle):     1.44901e-12
--------------------------------------------------------------------
Results for equation Bitcoin-USD daily Returns
==============================================================================================
                                  coefficient       std. error           t-stat            prob
----------------------------------------------------------------------------------------------
const                                0.003411         0.009955            0.343           0.732
L1.Bitcoin-USD daily Returns        -0.595927         0.514255           -1.159           0.247
L1.Equity ETF Daily Returns          0.159663         0.483229            0.330           0.741
L1.Gold ETF Daily Returns           -0.183055         1.211978           -0.151           0.880
L2.Bitcoin-USD daily Returns         0.070735         0.364168            0.194           0.846
L2.Equity ETF Daily Returns          0.187735         0.437807            0.429           0.668
L2.Gold ETF Daily Returns           -1.772342         0.817820           -2.167           0.030
L3.Bitcoin-USD daily Returns         0.344064         0.269414            1.277           0.202
L3.Equity ETF Daily Returns         -0.752451         0.391509           -1.922           0.055
L3.Gold ETF Daily Returns           -2.078432         1.196997           -1.736           0.082
L4.Bitcoin-USD daily Returns         0.214702         0.309319            0.694           0.488
L4.Equity ETF Daily Returns         -0.842141         0.554854           -1.518           0.129
L4.Gold ETF Daily Returns           -0.240722         1.239179           -0.194           0.846
L5.Bitcoin-USD daily Returns         0.052136         0.243404            0.214           0.830
L5.Equity ETF Daily Returns         -0.067874         0.538312           -0.126           0.900
L5.Gold ETF Daily Returns            0.267761         1.155924            0.232           0.817
L6.Bitcoin-USD daily Returns         0.199005         0.248379            0.801           0.423
L6.Equity ETF Daily Returns         -0.593242         0.475074           -1.249           0.212
L6.Gold ETF Daily Returns           -1.488821         0.988399           -1.506           0.132
L7.Bitcoin-USD daily Returns        -0.359651         0.288459           -1.247           0.212
L7.Equity ETF Daily Returns         -0.519775         0.666508           -0.780           0.435
L7.Gold ETF Daily Returns           -0.144244         1.220805           -0.118           0.906
L8.Bitcoin-USD daily Returns        -0.335439         0.357287           -0.939           0.348
L8.Equity ETF Daily Returns         -0.716595         0.475291           -1.508           0.132
L8.Gold ETF Daily Returns            0.262940         0.806287            0.326           0.744
L9.Bitcoin-USD daily Returns         0.397545         0.220575            1.802           0.071
L9.Equity ETF Daily Returns         -0.215931         0.609725           -0.354           0.723
L9.Gold ETF Daily Returns           -1.100643         0.711163           -1.548           0.122
L10.Bitcoin-USD daily Returns        0.616465         0.299404            2.059           0.039
L10.Equity ETF Daily Returns        -0.154533         0.482645           -0.320           0.749
L10.Gold ETF Daily Returns           1.103290         0.884962            1.247           0.213
L11.Bitcoin-USD daily Returns        0.195417         0.317444            0.616           0.538
L11.Equity ETF Daily Returns         0.077705         0.441641            0.176           0.860
L11.Gold ETF Daily Returns           0.981073         1.077818            0.910           0.363
L12.Bitcoin-USD daily Returns        0.243482         0.263218            0.925           0.355
L12.Equity ETF Daily Returns        -0.217967         0.459105           -0.475           0.635
L12.Gold ETF Daily Returns           0.867836         1.118580            0.776           0.438
L13.Bitcoin-USD daily Returns       -0.330865         0.268463           -1.232           0.218
L13.Equity ETF Daily Returns         0.766666         0.410652            1.867           0.062
L13.Gold ETF Daily Returns           1.405024         1.014668            1.385           0.166
L14.Bitcoin-USD daily Returns       -0.257673         0.360143           -0.715           0.474
L14.Equity ETF Daily Returns         0.578928         0.551807            1.049           0.294
L14.Gold ETF Daily Returns           0.401393         0.976601            0.411           0.681
==============================================================================================

Results for equation Equity ETF Daily Returns
==============================================================================================
                                  coefficient       std. error           t-stat            prob
----------------------------------------------------------------------------------------------
const                               -0.001645         0.011015           -0.149           0.881
L1.Bitcoin-USD daily Returns        -0.077773         0.569019           -0.137           0.891
L1.Equity ETF Daily Returns         -0.061693         0.534689           -0.115           0.908
L1.Gold ETF Daily Returns            0.714667         1.341044            0.533           0.594
L2.Bitcoin-USD daily Returns        -0.017434         0.402949           -0.043           0.965
L2.Equity ETF Daily Returns         -0.289049         0.484431           -0.597           0.551
L2.Gold ETF Daily Returns           -0.716811         0.904912           -0.792           0.428
L3.Bitcoin-USD daily Returns        -0.147219         0.298105           -0.494           0.621
L3.Equity ETF Daily Returns         -0.168861         0.433201           -0.390           0.697
L3.Gold ETF Daily Returns           -0.490121         1.324468           -0.370           0.711
L4.Bitcoin-USD daily Returns        -0.018769         0.342259           -0.055           0.956
L4.Equity ETF Daily Returns         -0.191858         0.613942           -0.313           0.755
L4.Gold ETF Daily Returns            0.415322         1.371142            0.303           0.762
L5.Bitcoin-USD daily Returns         0.009745         0.269325            0.036           0.971
L5.Equity ETF Daily Returns          0.050715         0.595638            0.085           0.932
L5.Gold ETF Daily Returns           -0.019461         1.279022           -0.015           0.988
L6.Bitcoin-USD daily Returns         0.258758         0.274830            0.942           0.346
L6.Equity ETF Daily Returns         -0.397964         0.525666           -0.757           0.449
L6.Gold ETF Daily Returns           -0.774582         1.093656           -0.708           0.479
L7.Bitcoin-USD daily Returns         0.036262         0.319177            0.114           0.910
L7.Equity ETF Daily Returns          0.222358         0.737486            0.302           0.763
L7.Gold ETF Daily Returns           -0.105797         1.350812           -0.078           0.938
L8.Bitcoin-USD daily Returns         0.042088         0.395336            0.106           0.915
L8.Equity ETF Daily Returns         -0.187554         0.525906           -0.357           0.721
L8.Gold ETF Daily Returns            0.265971         0.892150            0.298           0.766
L9.Bitcoin-USD daily Returns        -0.048044         0.244064           -0.197           0.844
L9.Equity ETF Daily Returns          0.214244         0.674656            0.318           0.751
L9.Gold ETF Daily Returns            0.407361         0.786897            0.518           0.605
L10.Bitcoin-USD daily Returns       -0.271438         0.331288           -0.819           0.413
L10.Equity ETF Daily Returns        -0.074856         0.534043           -0.140           0.889
L10.Gold ETF Daily Returns           0.638749         0.979204            0.652           0.514
L11.Bitcoin-USD daily Returns       -0.098615         0.351249           -0.281           0.779
L11.Equity ETF Daily Returns         0.187175         0.488672            0.383           0.702
L11.Gold ETF Daily Returns          -0.598747         1.192598           -0.502           0.616
L12.Bitcoin-USD daily Returns        0.167849         0.291249            0.576           0.564
```

```
L12.Equity ETF Daily Returns        -0.003950         0.507996           -0.008           0.994
L12.Gold ETF Daily Returns           0.112130         1.237700            0.091           0.928
L13.Bitcoin-USD daily Returns        0.057090         0.297052            0.192           0.848
L13.Equity ETF Daily Returns         0.293392         0.454383            0.646           0.518
L13.Gold ETF Daily Returns           0.620015         1.122723            0.552           0.581
L14.Bitcoin-USD daily Returns        0.324312         0.398495            0.814           0.416
L14.Equity ETF Daily Returns        -0.063512         0.610570           -0.104           0.917
L14.Gold ETF Daily Returns          -0.256037         1.080602           -0.237           0.813
===============================================================================================

Results for equation Gold ETF Daily Returns
===============================================================================================
                                   coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------------------------
const                                0.005221         0.003025            1.726           0.084
L1.Bitcoin-USD daily Returns         0.307940         0.156291            1.970           0.049
L1.Equity ETF Daily Returns         -0.081963         0.146862           -0.558           0.577
L1.Gold ETF Daily Returns           -0.513926         0.368341           -1.395           0.163
L2.Bitcoin-USD daily Returns         0.034371         0.110677            0.311           0.756
L2.Equity ETF Daily Returns          0.067669         0.133057            0.509           0.611
L2.Gold ETF Daily Returns            0.012702         0.248550            0.051           0.959
L3.Bitcoin-USD daily Returns         0.056560         0.081880            0.691           0.490
L3.Equity ETF Daily Returns         -0.112155         0.118986           -0.943           0.346
L3.Gold ETF Daily Returns            0.088984         0.363788            0.245           0.807
L4.Bitcoin-USD daily Returns        -0.125657         0.094007           -1.337           0.181
L4.Equity ETF Daily Returns          0.078244         0.168630            0.464           0.643
L4.Gold ETF Daily Returns            0.382896         0.376608            1.017           0.309
L5.Bitcoin-USD daily Returns        -0.113307         0.073975           -1.532           0.126
L5.Equity ETF Daily Returns          0.051457         0.163602            0.315           0.753
L5.Gold ETF Daily Returns            0.044931         0.351305            0.128           0.898
L6.Bitcoin-USD daily Returns        -0.009884         0.075487           -0.131           0.896
L6.Equity ETF Daily Returns         -0.204184         0.144383           -1.414           0.157
L6.Gold ETF Daily Returns           -0.721706         0.300391           -2.403           0.016
L7.Bitcoin-USD daily Returns        -0.041330         0.087668           -0.471           0.637
L7.Equity ETF Daily Returns         -0.052756         0.202563           -0.260           0.795
L7.Gold ETF Daily Returns           -0.050415         0.371023           -0.136           0.892
L8.Bitcoin-USD daily Returns         0.051851         0.108586            0.478           0.633
L8.Equity ETF Daily Returns         -0.080477         0.144449           -0.557           0.577
L8.Gold ETF Daily Returns           -0.162364         0.245044           -0.663           0.508
L9.Bitcoin-USD daily Returns         0.098614         0.067036            1.471           0.141
L9.Equity ETF Daily Returns         -0.007489         0.185306           -0.040           0.968
L9.Gold ETF Daily Returns           -0.162484         0.216135           -0.752           0.452
L10.Bitcoin-USD daily Returns        0.007767         0.090994            0.085           0.932
L10.Equity ETF Daily Returns        -0.148633         0.146684           -1.013           0.311
L10.Gold ETF Daily Returns           0.134628         0.268955            0.501           0.617
L11.Bitcoin-USD daily Returns       -0.082462         0.096477           -0.855           0.393
L11.Equity ETF Daily Returns        -0.152203         0.134222           -1.134           0.257
L11.Gold ETF Daily Returns          -0.773946         0.327567           -2.363           0.018
L12.Bitcoin-USD daily Returns        0.013891         0.079891            0.174           0.862
L12.Equity ETF Daily Returns        -0.346987         0.139530           -2.487           0.013
L12.Gold ETF Daily Returns          -0.506488         0.339956           -1.490           0.136
L13.Bitcoin-USD daily Returns       -0.177432         0.081590           -2.175           0.030
L13.Equity ETF Daily Returns         0.091040         0.124804            0.729           0.466
L13.Gold ETF Daily Returns          -0.279602         0.308375           -0.907           0.365
L14.Bitcoin-USD daily Returns        0.121721         0.109454            1.112           0.266
L14.Equity ETF Daily Returns        -0.207909         0.167704           -1.240           0.215
L14.Gold ETF Daily Returns          -0.166140         0.296806           -0.560           0.576
===============================================================================================

Correlation matrix of residuals
                              Bitcoin-USD daily Returns  Equity ETF Daily Returns  Gold ETF Daily Returns
Bitcoin-USD daily Returns                      1.000000                  0.413324               0.527063
Equity ETF Daily Returns                       0.413324                  1.000000              -0.534705
Gold ETF Daily Returns                         0.527063                 -0.534705               1.000000
```

Testing Vector Autoregressive Model for Q3

In [451]:

```python
from statsmodels.tsa.api import VAR
aic = 1000
var_modelq3 = VAR(df_all_returns.loc['Jul-2020':'Sep-2020'])
for lag in range(14):
    score = var_modelq3.fit(lag).aic

    if(score<aic):
        aic = score
        p = lag

print('''The VAR model for the 3 dimensional timeseries has the best aic score as {}
with lags of {}'''.format(round(aic,2),p))
```

```
The VAR model for the 3 dimensional timeseries has the best aic score as -25.04
with lags of 0
```

Testing the VAR model for the entire time frame for all 3 assets

In [452]:

```python
from statsmodels.tsa.api import VAR
aic = 1000
var_model = VAR(df_all_returns.loc['Apr-2020':'Dec-2020'])
for lag in range(45):
    score = var_model.fit(lag).aic
    #print('The VAR model for the 3 dimensional timeseries has the aic score as {} with lags of {}'.format(score,lag))
    if(score<aic):
        aic = score
        p = lag

print('''The VAR model for the 3 dimensional timeseries has the best aic
      score as {} with lags of {}'''.format(round(aic,2),p))
```

```
The VAR model for the 3 dimensional timeseries has the best aic
      score as -28.39 with lags of 44
```

Conducting Johansen Test

Creating the dataframe

In [465]:

```python
df_prices = pd.DataFrame(df_bitcoin['Close'].rename('Bitcoin-USD Closing')).join(pd.DataFrame(df_equityetf['Close'].rename('Equity ETF Closing')).join(pd
```

Creating a Johansen Function

In [466]:
```
def get_johansen(y, p):
        """
        Get the cointegration vectors at 95% level of significance
        given by the trace statistic test.
        """

        N, l = y.shape
        jres = coint_johansen(y, 0, p)          # 0: No deterministic trend and p = no of lagged differences
        trstat = jres.lr1                       # trace statistic
        tsignf = jres.cvt                       # critical values
        r = 0

        for i in range(l):
            if trstat[i] > tsignf[i, 1]:     # 0: 90%  1:95% 2: 99%
                r = i + 1
        print ("There are {} cointegration relationships".format(r))
        return jres
```

Since we obtain cointegration between Bitcoin and Gold ETF for Q2, hence we will conduct the Johansen Test for these 2 securities

Conducting Johansen Test for Q2

For Quarter 2, Since Equity ETF was non-stationary for ADF test while Bitcoin and Gold were non-stationary for ADF Test, we will use Johansen test for only these 2 assets

In [467]:
```
jsresq2 = get_johansen(df_prices[['Bitcoin-USD Closing','Gold ETF Closing']].loc['Apr-2020':'Jun-2020'], 15)
```

```
There are 1 cointegration relationships
```

In [468]:
```
v1=jsresq2.evec[:,0]
print(v1)
```

```
[ 0.01006442 -2.349604  ]
```

Conducting Johansen Test for Q3

For Quarter 3, All assets were non-stationary for ADF Test with a constant trend, we will use Johansen test for all 3 assets

In [457]:
```
jsresq3 = get_johansen(df_prices.loc['Jul-2020':'Sep-2020'], 14)
```

```
There are 3 cointegration relationships
```

In [458]:
```
v1=jsresq3.evec[:,0]
v2=jsresq3.evec[:,1]
v3=jsresq3.evec[:,2]
print (v1)
print (v2)
print (v3)
```

```
[ 0.04108241 -0.19963914 -9.23906812]
[-0.01838497 -0.01331267  2.75034474]
[-0.01367274  0.02331892  1.48661788]
```

Appendix: Correlation Estimates for Technical Reports

In [459]:
```
corr, _ = pearsonr(df_close["Gold ETF Closing Price"].loc['Apr-2020':'Jun-2020'],
                   df_close["Equity ETF Closing Price"].loc['Apr-2020':'Jun-2020'])
print('Pearsons correlation between Gold ETF and Equity ETF for month of October is: %.3f' % corr)
```

```
Pearsons correlation between Gold ETF and Equity ETF for month of October is: 0.580
```

In [460]:
```
corr, _ = pearsonr(df_close["Gold ETF Closing Price"].loc['Jul-2020':'Sep-2020'],
                   df_close["Equity ETF Closing Price"].loc['Jul-2020':'Sep-2020'])
print('Pearsons correlation between Gold ETF and Equity ETF for month of October is: %.3f' % corr)
```

```
Pearsons correlation between Gold ETF and Equity ETF for month of October is: -0.401
```

In [461]:
```
corr, _ = pearsonr(df_close["Gold ETF Closing Price"].loc['Oct-2020':'Dec-2020'],
                   df_close["Equity ETF Closing Price"].loc['Oct-2020':'Dec-2020'])
print('Pearsons correlation between Gold ETF and Equity ETF for month of October is: %.3f' % corr)
```

```
Pearsons correlation between Gold ETF and Equity ETF for month of October is: -0.600
```

In [462]:
```
corr, _ = pearsonr(df_close["Bitcoin Closing Price"].loc['Apr-2020':'Jun-2020'],
                   df_close["Equity ETF Closing Price"].loc['Apr-2020':'Jun-2020'])
print('Pearsons correlation between Bitcoin ETF and Equity ETF for month of October is: %.3f' % corr)
```

```
Pearsons correlation between Bitcoin ETF and Equity ETF for month of October is: 0.811
```

In [463]:
```
corr, _ = pearsonr(df_close["Bitcoin Closing Price"].loc['Jul-2020':'Sep-2020'],
                   df_close["Equity ETF Closing Price"].loc['Jul-2020':'Sep-2020'])
print('Pearsons correlation between Bitcoin ETF and Equity ETF for month of October is: %.3f' % corr)
```

```
Pearsons correlation between Bitcoin ETF and Equity ETF for month of October is: -0.399
```

In [464]:
```
corr, _ = pearsonr(df_close["Bitcoin Closing Price"].loc['Oct-2020':'Dec-2020'],
                   df_close["Equity ETF Closing Price"].loc['Oct-2020':'Dec-2020'])
print('Pearsons correlation between Bitcoin ETF and Equity ETF for month of October is: %.3f' % corr)
```

```
Pearsons correlation between Bitcoin ETF and Equity ETF for month of October is: 0.811
```