In [ ]:

```python
import pandas as pd
from pandas import read_csv
import numpy as np
import matplotlib.pyplot as plt
import statistics
```

Import the relevant subset

In [ ]:

```python
url = 'https://raw.githubusercontent.com/Jneny/Hospitalcapacity/main/Data/icu_beds.csv'
data = read_csv(url, header=0, parse_dates=[0], index_col=0)
data = data.asfreq('d')
adultcrit = pd.DataFrame(data, columns=['adult_icu_crci_patients'])
sadultcrit = pd.Series(adultcrit.adult_icu_crci_patients)
```

Print the attribute type : Numeric integer, the data is quantitative and discrete as it measures each patient occupancy in terms of beds

In [ ]:

```python
print(adultcrit.info())
print(adultcrit.head())
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 655 entries, 2020-05-01 to 2022-02-14
Freq: D
Data columns (total 1 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   adult_icu_crci_patients  655 non-null    int64
dtypes: int64(1)
memory usage: 10.2 KB
None
            adult_icu_crci_patients
date
2020-05-01                      244
2020-05-02                      236
2020-05-03                      246
2020-05-04                      243
2020-05-05                      243
```

check for missing observations in feature attribute

In [ ]:

```python
data.isnull().values.any()
```

Out[ ]:

```
False
```

In [ ]:

```python
data.duplicated(keep='first')
missing1 = (data['adult_icu_crci_patients'] == 0).sum()
print(missing1)
```

```
0
```

In [ ]:

```python
#check for times when there a provincial shortage of occupancy space when no beds were aviailable. Could mean t
icumax = (data['available_adult_icu_beds'] == 0).sum()
print(icumax)
```

```
0
```

In [ ]:

```python
adultcrit.tail()
```

Out[ ]:

	adult_icu_crci_patients
date	
2022-02-10	442
2022-02-11	429
2022-02-12	407
2022-02-13	395
2022-02-14	386

In [ ]:

```python
#Find max, min, mean and standard deviation of main attributes.
adultcrit.describe()
```

Out[ ]:

	adult_icu_crci_patients
count	655.000000
mean	251.783206
std	204.279555
min	18.000000
25%	114.500000
50%	172.000000
75%	349.500000
max	889.000000

In [ ]:

```
statistics.median(adultcrit['adult_icu_crci_patients'])
```

Out[ ]:

```
172
```

Attribute elimination: We will want to focus on the patients in critical conditions rather than amount of beds available or total because it will be more useful to know now many are absolutely necessary to set aside for CRCI patients within the ICU, and the focus will be on adult CRCI patients as the adult population including seniors has been most fatally affected by COVID. Therefore, the main variable of focus will be adult CRCI patient numbers. As the objective is univariate time series analysis, the statistical method means all other attributes can be eliminated.

In [ ]:

```
import seaborn as sns
```

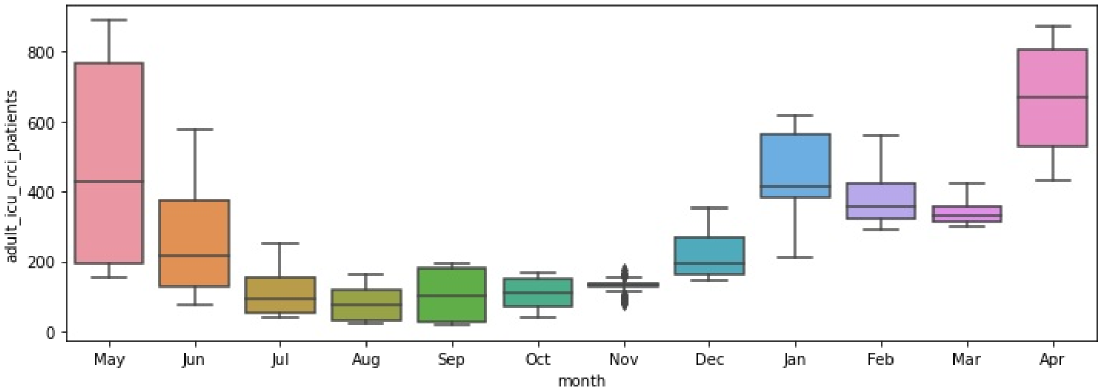
In [ ]:

```
adultcrit2 = adultcrit.copy(deep=True)
```

Distribution of data points by month

In [ ]:

```
adultcrit2['month'] = adultcrit2.index.strftime('%b')
fig, ax = plt.subplots()
fig.set_size_inches((12,4))
sns.boxplot(x='month',y='adult_icu_crci_patients',data=adultcrit2,ax=ax)
plt.show()
```

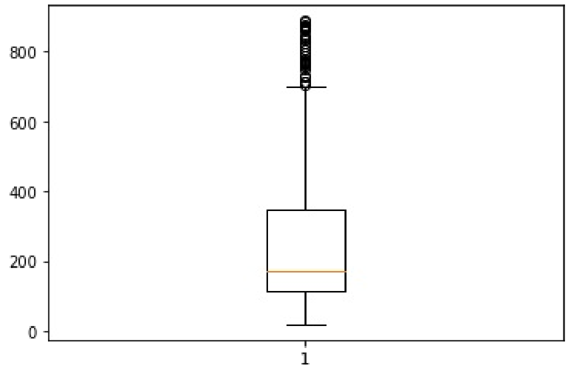


In [ ]:

```
# convert dataframe values to array for boxplot, see that there are a large amount of outliers in the upper ran
boxarray = adultcrit.values
plt.boxplot(boxarray)
```

Out[ ]:

```
{'boxes': [<matplotlib.lines.Line2D at 0x7fa4be76a210>],
 'caps': [<matplotlib.lines.Line2D at 0x7fa4be770290>,
  <matplotlib.lines.Line2D at 0x7fa4be7707d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fa4be6f82d0>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7fa4be770d50>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7fa4be76a7d0>,
  <matplotlib.lines.Line2D at 0x7fa4be76ad10>]}
```



In [ ]:

```
# print(boxarray)
```

In [ ]:
```
# print(boxarray)
```

In [ ]:
```
# finding the 1st and 3rd quartile
q1 = np.quantile(boxarray, 0.25)

q3 = np.quantile(boxarray, 0.75)
med = np.median(boxarray)

# finding the iqr region
iqr = q3-q1

# finding upper and lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
print(f' The interquartile range is: {iqr} \n The upper whiskers are: {upper_bound} \n The lower whiskers are:
```

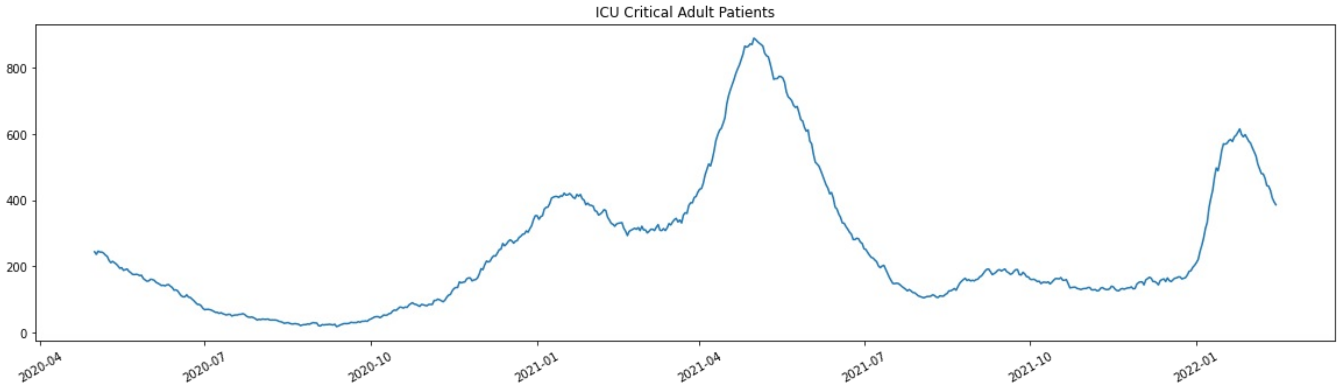
```
 The interquartile range is: 235.0
 The upper whiskers are: 702.0
 The lower whiskers are: -238.0
```

In [ ]:
```
#Finding outliers
outliers = boxarray[(boxarray <= lower_bound) | (boxarray >= upper_bound)]
print(f'There are {len(outliers)} Outliers \n The outliers in the boxplot are :{outliers}')
```

```
There are 35 Outliers
 The outliers in the boxplot are :[715 733 748 764 782 796 808 824 841 865 862 865 872 870 889 885 879 874
 870 865 844 836 833 813 790 765 767 767 774 773 769 756 727 712 707]
```

Graphing the data, finding majority of critical patients needed care around after Jan and during May, early in the year Outliers cannot be disregarded as every datapoint is important for the timeseries

In [ ]:
```
plt.plot(sadultcrit) #main focus
plt.title('ICU Critical Adult Patients')
plt.xticks(rotation = 30)
plt.show()
```



Distribution Correlation Balance - all class variables are balanced as there are no missing values and we are only using one attribute The feature attribute does not need to be transformed as adult_icu_crci_patients is already measured as patients in critical care occupying a bed

Check for additive/ multiplicative components of adult_icu_crci_patients. Shows trend, seasonality, and noise

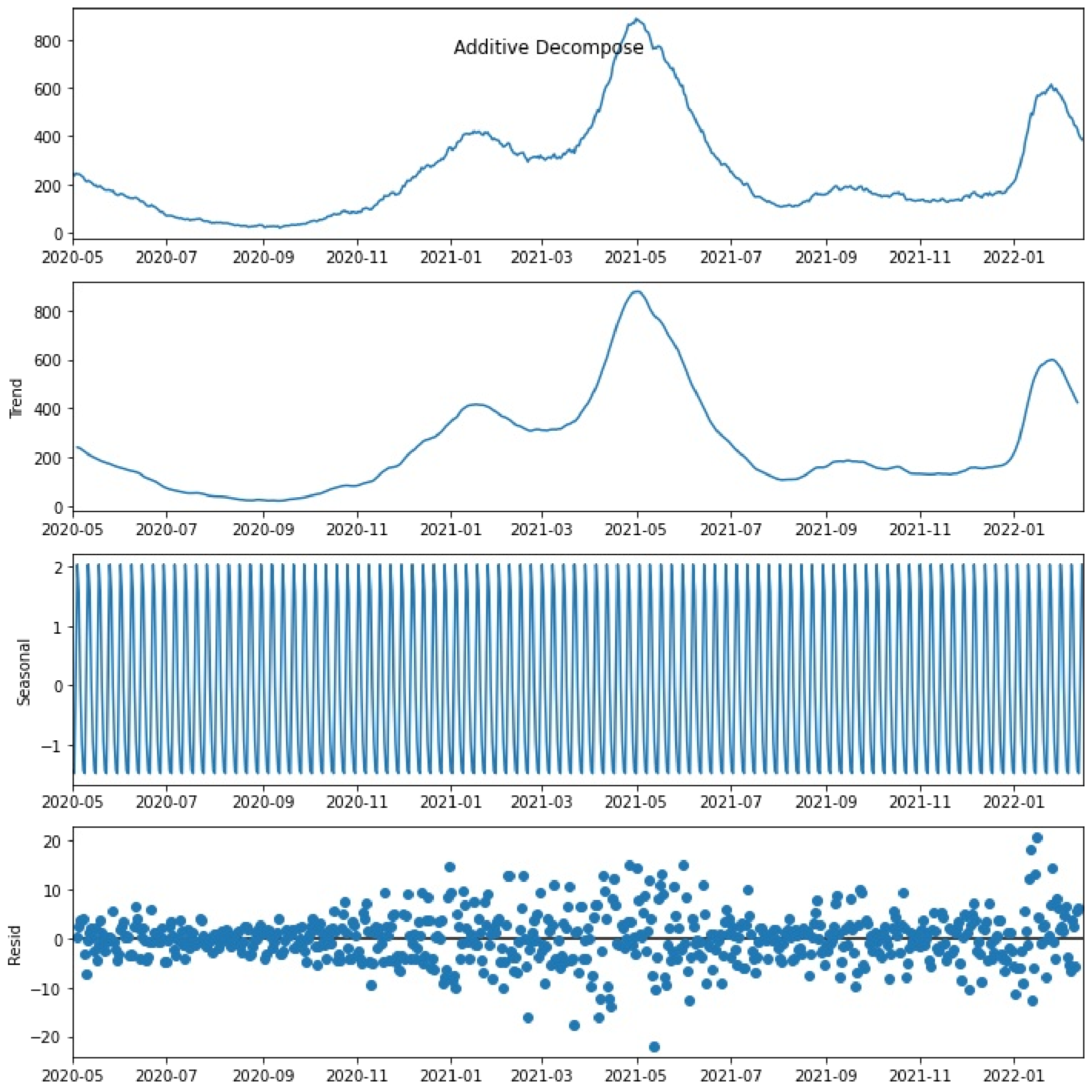
Results show trend exists, not much seasonality in both, considerable noise in additive model meaning more variability.

In [ ]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
from dateutil.parser import parse
```

In [ ]:
```
add_result = seasonal_decompose(adultcrit, model = "additive")
mul_result = seasonal_decompose(adultcrit, model = "multiplicative")
```

In [ ]:
```
plt.rcParams.update({'figure.figsize': (10,10)})
add_result.plot().suptitle('\n Additive Decompose', fontsize = 12)
```

Out[ ]:
```
Text(0.5, 0.98, '\n Additive Decompose')
```

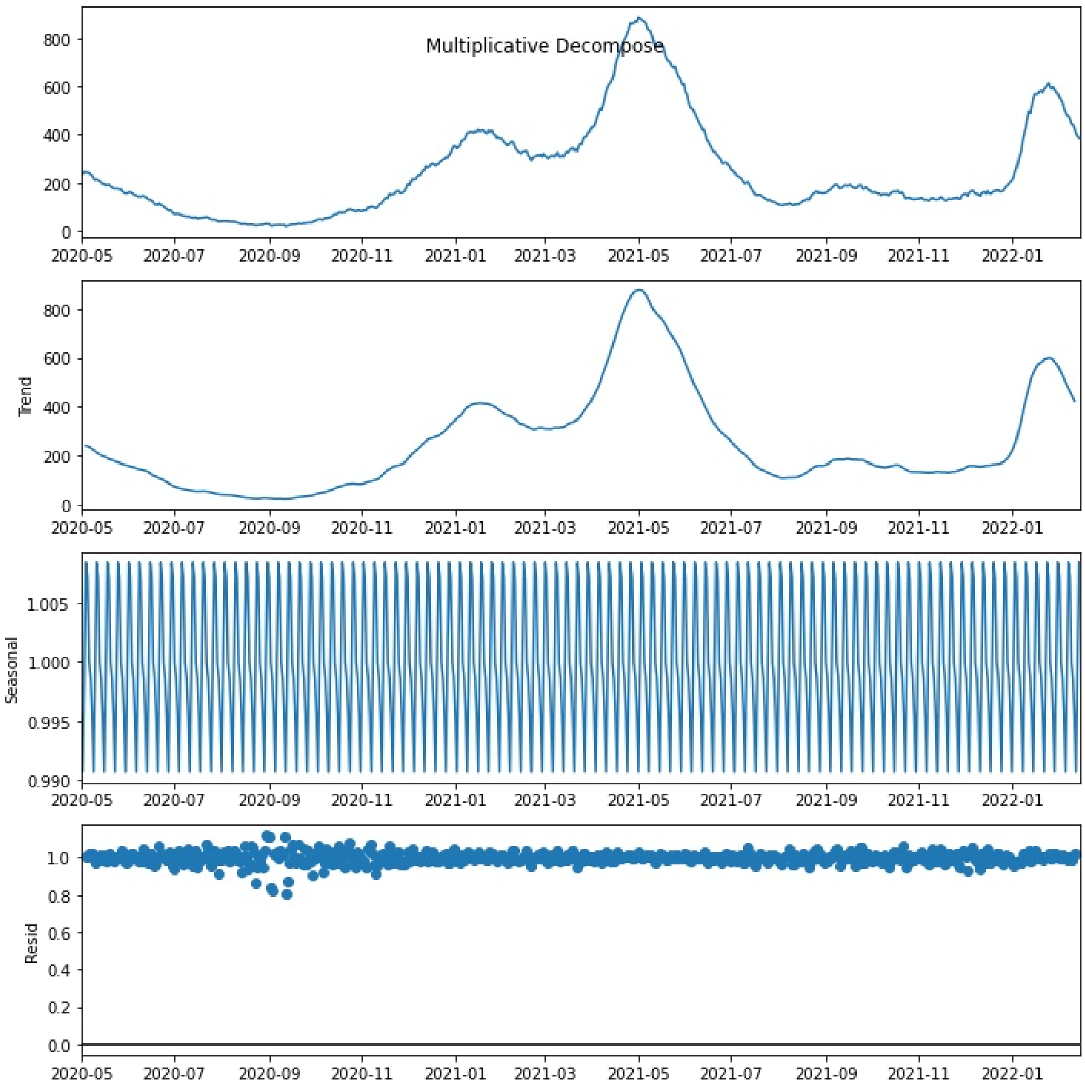


In [ ]:

```
plt.rcParams.update({'figure.figsize': (10,10)})
mul_result.plot().suptitle('\n Multiplicative Decompose', fontsize = 12)
```

Out[ ]:

```
Text(0.5, 0.98, '\n Multiplicative Decompose')
```



Checking stationarity of dataset as to run the time series through ARIMA will require constant mean and variance.

In [ ]:

```
from statsmodels.tsa.stattools import adfuller
```

In [ ]:

```
adfuller(adultcrit.adult_icu_crci_patients.values)
```

Out[ ]:

```
(-2.415029400292793,
 0.1375393258204045,
 17,
 637,
 {'1%': -3.44065745275905,
  '5%': -2.8660879520543534,
  '10%': -2.5691919933016076},
 4278.403262572921)
```

Not stationary as-is, will need to smooth to use in ARIMA - done in the "Modelling.ipynb" file

In [ ]:

```
#End of EDA
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js