In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/pandas/core/computation/expressions.py:20
: UserWarning: Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently install
ed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [3]:
```
df = pd.read_excel('airline_data.xlsx')
df.head()
```

Out[3]:

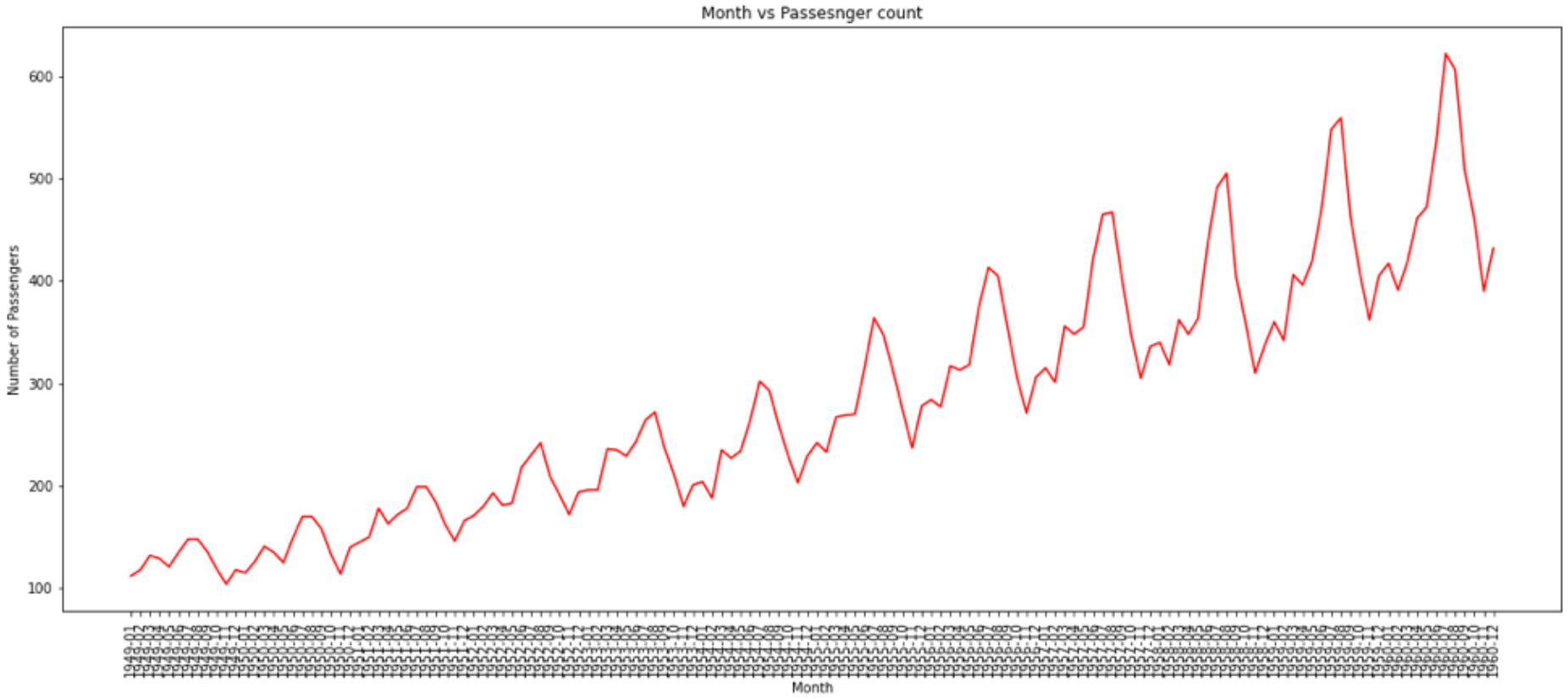
	Month	Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

In [4]:
```
df.shape
```

Out[4]:
```
(144, 2)
```

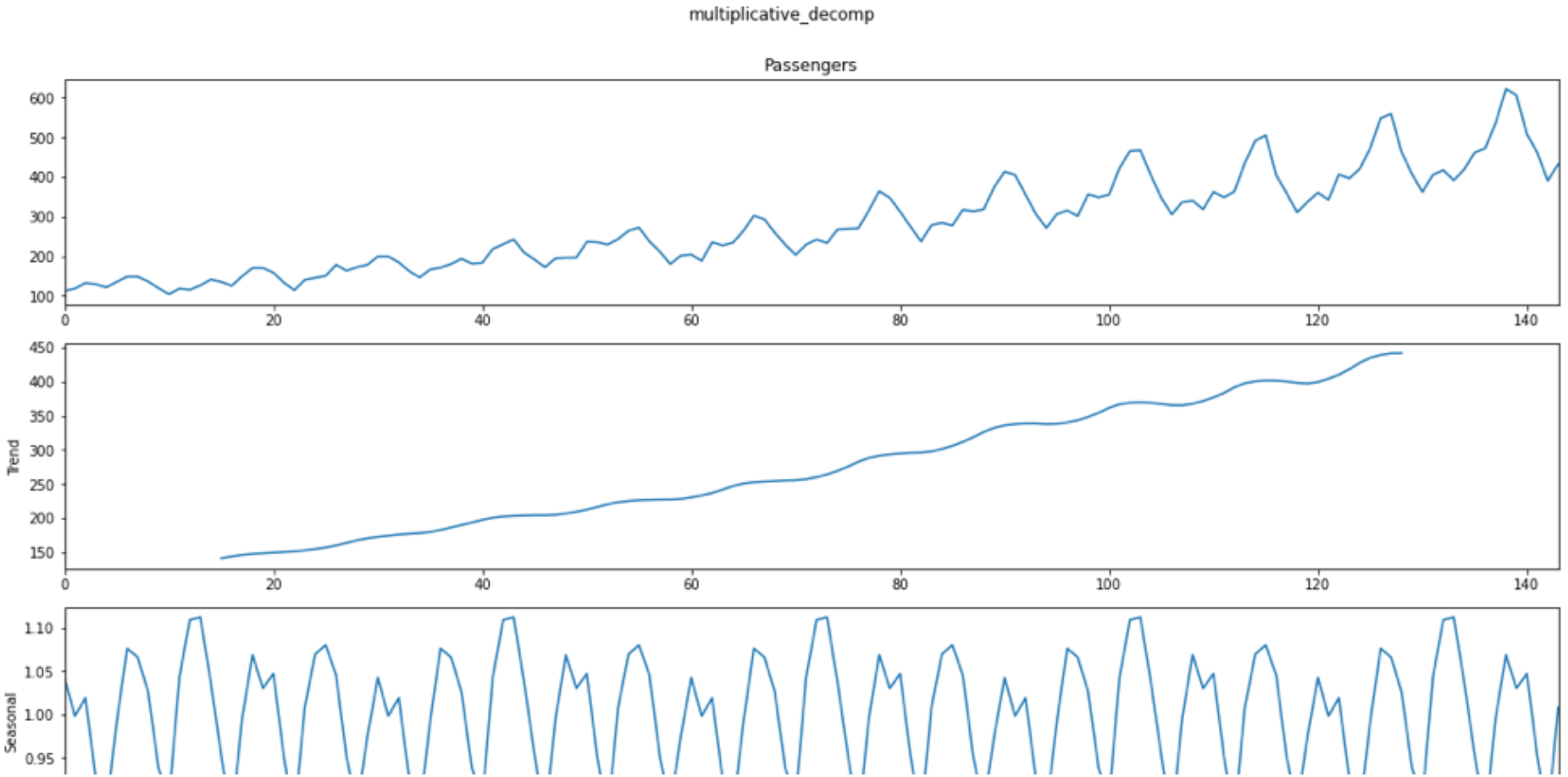
In [10]:

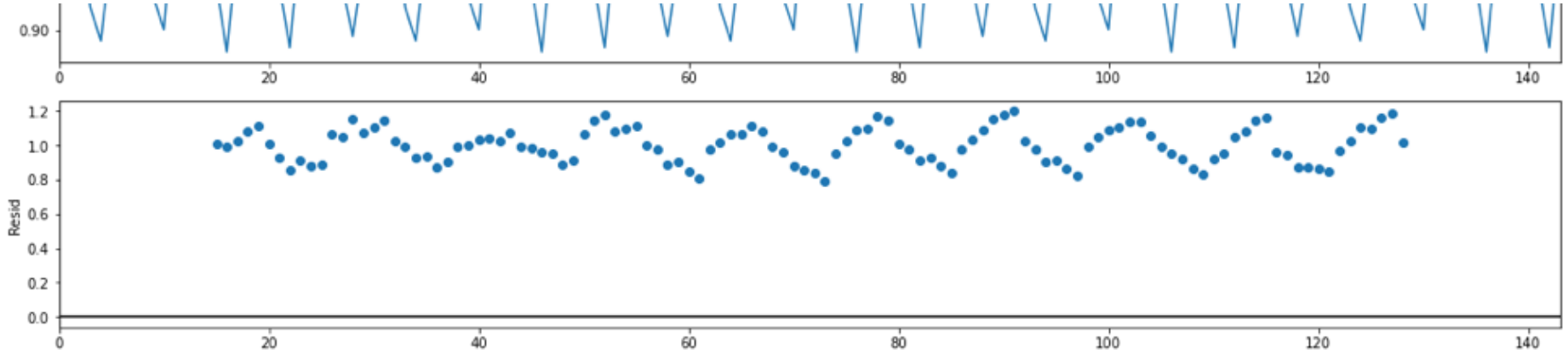
```
# plot data
x = df['Month']
x = np.array(x)
y = df['Passengers']
y = np.array(y)
plt.figure(figsize = (20,8))
plt.plot(x,y,color = 'red')
plt.xlabel("Month")
plt.xticks(rotation=90)
plt.ylabel("Number of Passengers")
plt.title('Month vs Passesnger count')
plt.show()
```



In [16]:

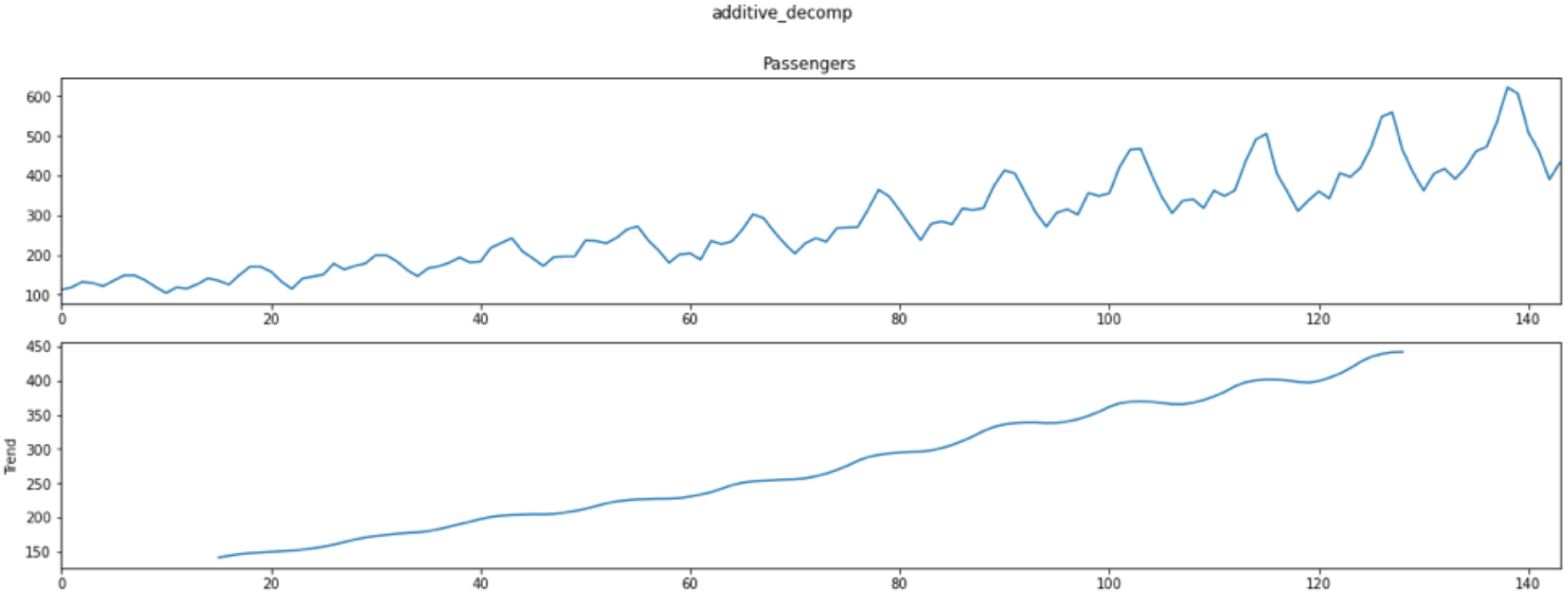
```python
from statsmodels.tsa.seasonal import seasonal_decompose

# multiplicative decomposition

multiplicative_decomp = seasonal_decompose(df['Passengers'],model = 'multiplicative',period = 30)
additive_decomp = seasonal_decompose(df['Passengers'],model = 'additive',period = 30)

# plot

plt.rcParams.update({'figure.figsize':(16,12)})
multiplicative_decomp.plot().suptitle("multiplicative_decomp")
plt.tight_layout(rect = [0,0.03,1,0.95])
plt.show()
```

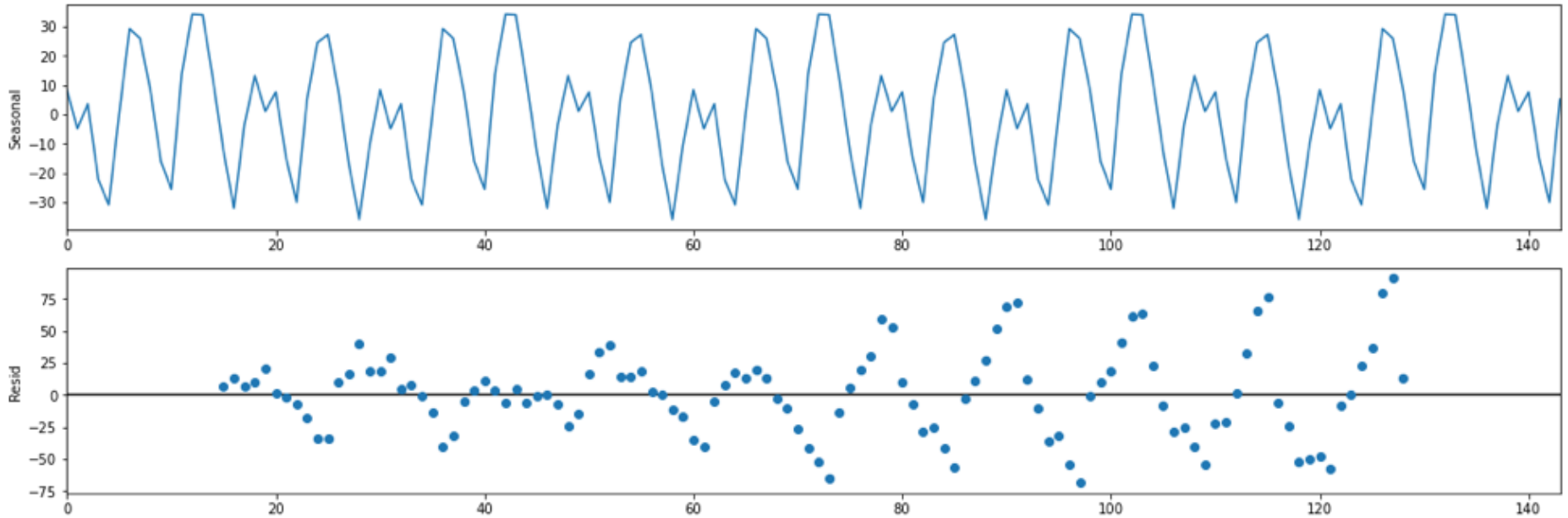




In [17]:

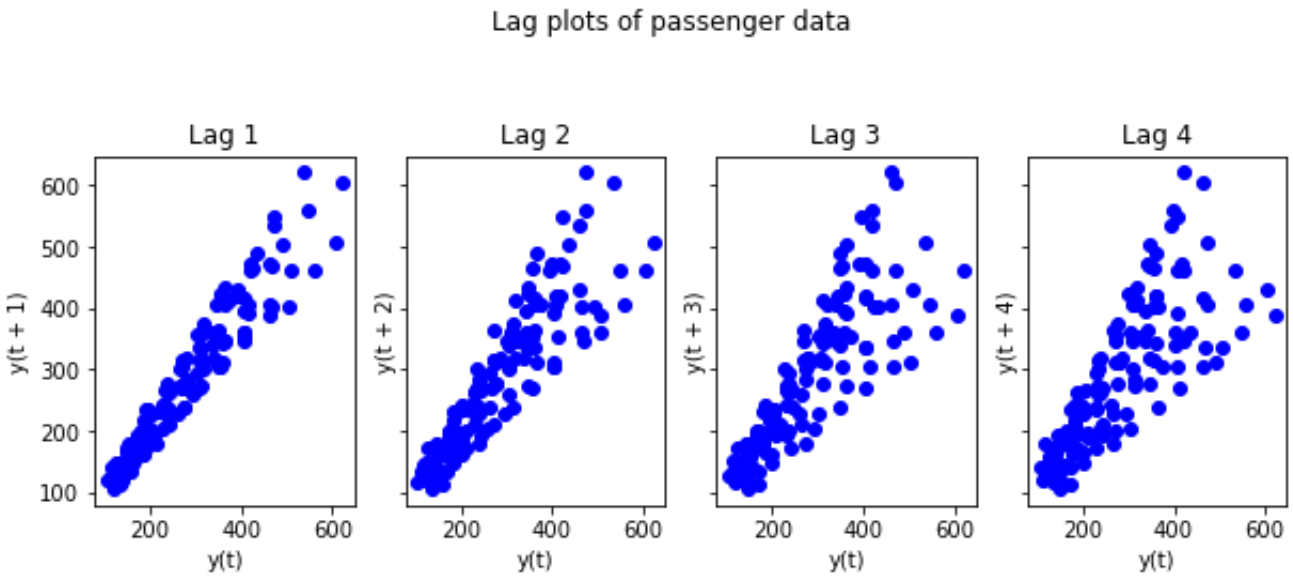
```
# plot

plt.rcParams.update({'figure.figsize':(16,12)})
additive_decomp.plot().suptitle("additive_decomp")
plt.tight_layout(rect = [0,0.03,1,0.95])
plt.show()
```




Seasonal
30
20
10
0
−10
−20
−30
0
20
40
60
80
100
120
140
Resid
75
50
25
0
−25
−50
−75
0
20
40
60
80
100
120
140

In [30]:

```
from pandas.plotting import lag_plot

# plot
fig,axes = plt.subplots(1,4,figsize = (10,3),sharex = True, sharey = True)
for i, ax in enumerate(axes.flatten()[:4]):
    lag_plot(df['Passengers'],lag = i+1,ax = ax,c = 'blue')
    ax.set_title('Lag '+ str(i+1))

fig.suptitle("Lag plots of passenger data",x = 0.5, y = 1.2)
plt.show()
```



In [33]:

```
df.drop(['Month'],axis = 1,inplace = True)
```

In [39]:

```
# test for stationarity

rolling_mean = df.rolling(7).mean()
rolling_std = df.rolling(7).std()
```

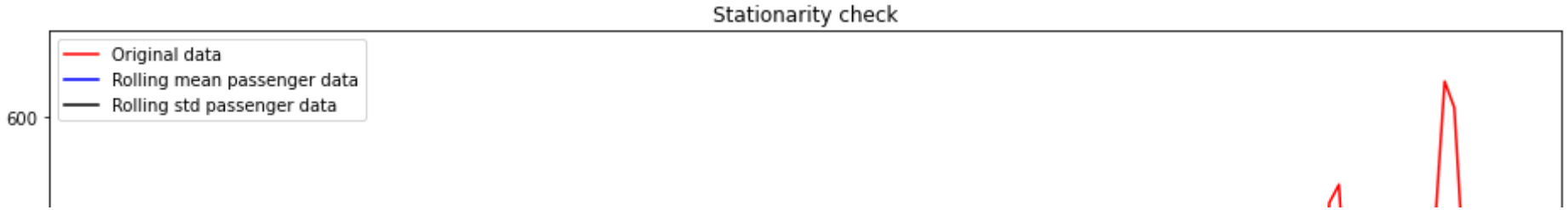
In [40]:
```
rolling_mean
```

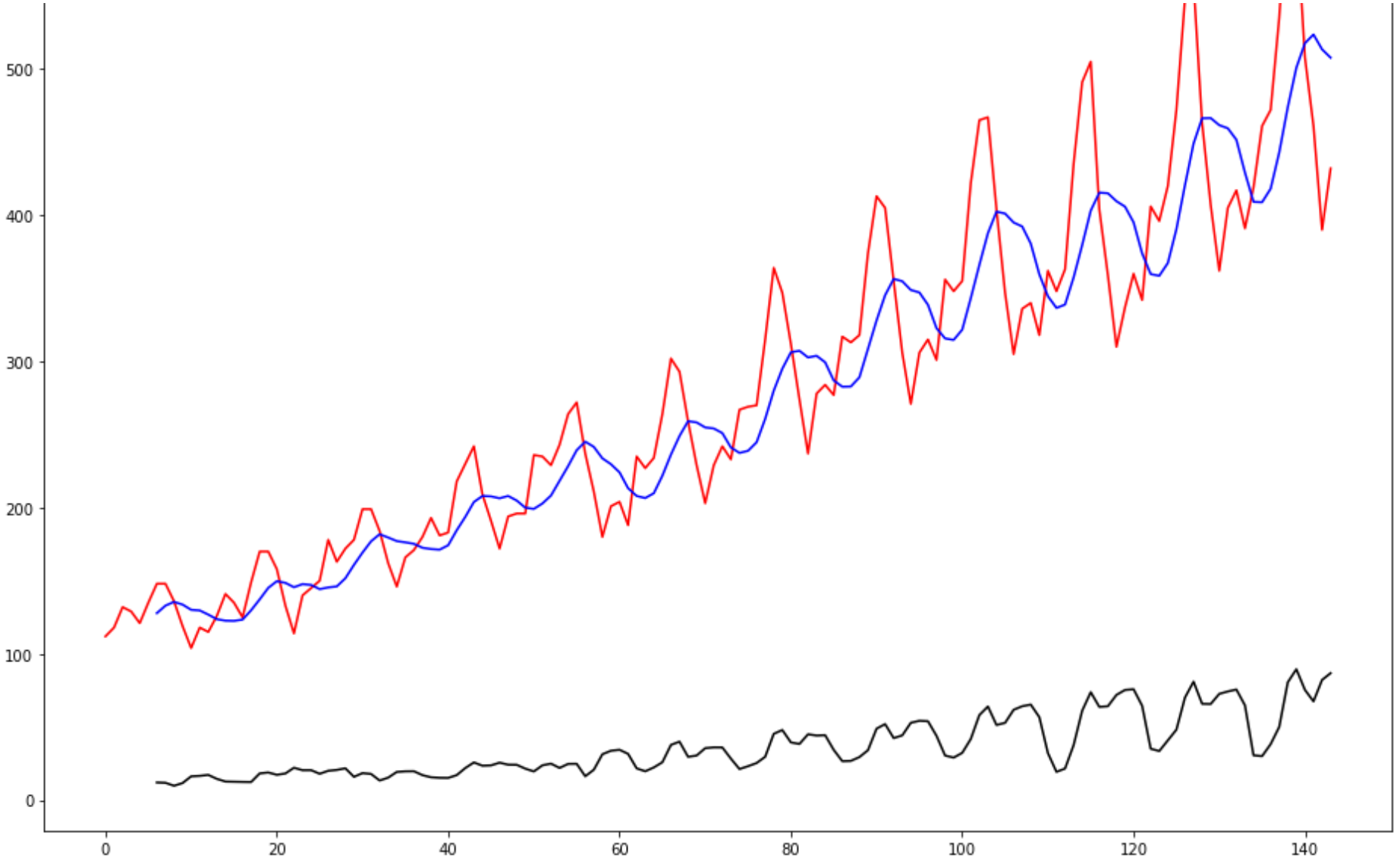
Out[40]:

	Passengers
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
...	...
139	500.857143
140	517.571429
141	523.571429
142	513.428571
143	507.714286

144 rows × 1 columns

In [42]:
```
plt.plot(df,color = 'red',label = 'Original data')
plt.plot(rolling_mean,color = 'blue',label = 'Rolling mean passenger data')
plt.plot(rolling_std,color = 'black',label = 'Rolling std passenger data')
plt.title("Stationarity check")
plt.legend()
plt.show()
```





In [43]:

```
from statsmodels.tsa.stattools import adfuller
adft = adfuller(df)
```

In [44]:
```
adft
```

Out[44]:
```
(0.8153688792060472,
 0.991880243437641,
 13,
 130,
 {'1%': -3.4816817173418295,
  '5%': -2.8840418343195267,
  '10%': -2.578770059171598},
 996.6929308390189)
```

In [59]:
```
# as p value is greater than 5% and from graph also we can conclude data
# is not stationary
```

In [65]:
```
df
```

Out[65]:

	Passengers
0	112
1	118
2	132
3	129
4	121
...	...
139	606
140	508
141	461
142	390
143	432

144 rows × 1 columns

In [61]:
```
# autocorrelation in lagged data

auto_corr_lag1 = df['Passengers'].autocorr(lag = 1)
print("One month lag : ", auto_corr_lag1)
```

```
One month lag :  0.9601946480498523
```

In [62]:
```
auto_corr_lag3 = df['Passengers'].autocorr(lag = 3)
print("Three month lag : ", auto_corr_lag3)
```

```
Three month lag :  0.837394765081794
```

In [63]:
```
auto_corr_lag6 = df['Passengers'].autocorr(lag = 6)
print("Six month lag : ", auto_corr_lag6)
```

```
Six month lag :  0.7839187959206183
```

In [64]:
```
auto_corr_lag9 = df['Passengers'].autocorr(lag = 9)
print("Nine month lag : ", auto_corr_lag9)
```

```
Nine month lag :  0.8278519011167601
```

In [66]:
```
# forecasting
```

In [67]:
```

```

Out[67]:

	Passengers
0	112
1	118
2	132
3	129
4	121
...	...
139	606
140	508
141	461
142	390
143	432

144 rows × 1 columns

In [87]:
```
df = pd.read_excel('airline_data.xlsx')
df.head()
```

Out[87]:

	Month	Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

In [88]:
```
df.set_index('Month')
```

Out[88]:

	Passengers
Month	
1949-01	112
1949-02	118
1949-03	132
1949-04	129
1949-05	121
...	...
1960-08	606
1960-09	508
1960-10	461
1960-11	390

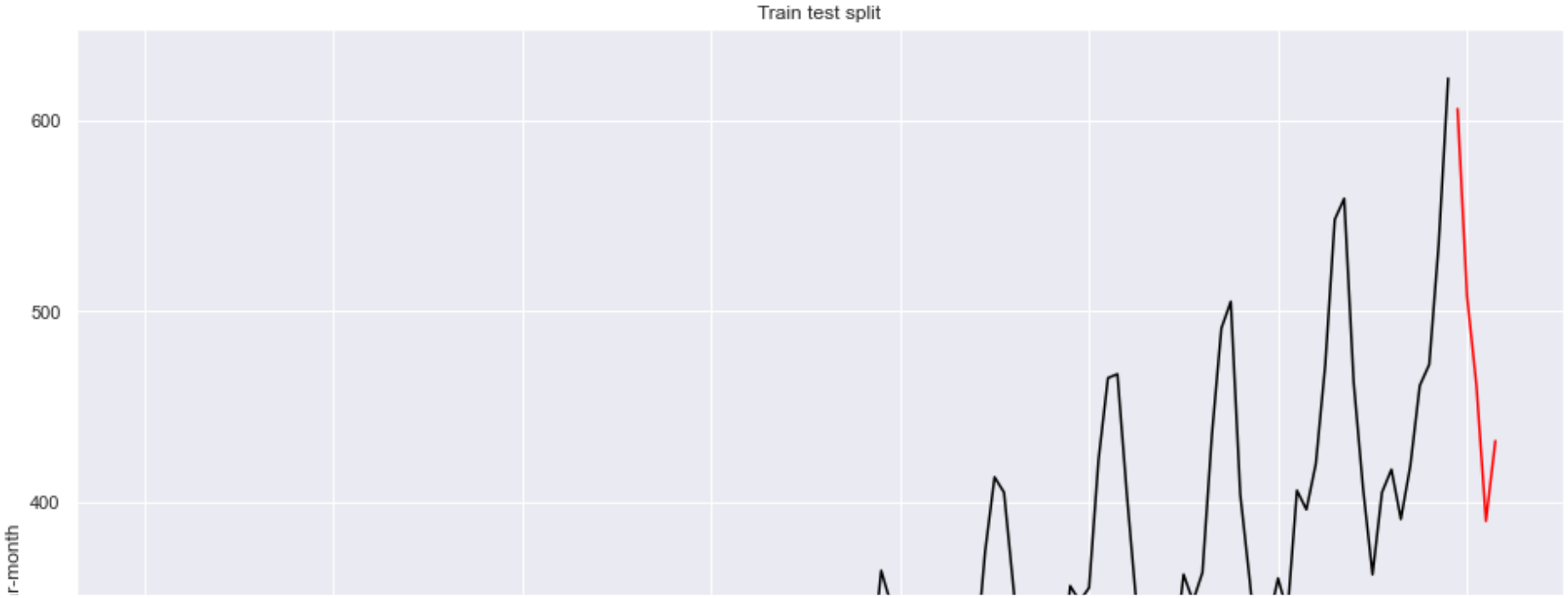
In [80]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Month       144 non-null    datetime64[ns]
 1   Passengers  144 non-null    int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 2.4 KB
```

In [89]:
```
df['Month'] = pd.to_datetime(df['Month'],format = '%Y-%m')
train = df[df['Month']<"1960-08"]
train['train'] = train['Passengers']
test = df[df['Month']>='1960-08']
train['test'] = test['Passengers']
train.set_index('Month')
test.set_index('Month')
```

In [98]:

```
plt.plot(train['Passengers'],color = 'black')
plt.plot(test['Passengers'],color = 'red')
plt.title("Train test split")
plt.xlabel("Passenger count")
plt.ylabel("Year-month")
plt.show()
```



In [99]:

```
# modelling

from pmdarima.arima import auto_arima
model = auto_arima(train['Passengers'],trace =True )
model.fit(train['Passengers'])
forecast = model.predict(n_periods = len(test['Passengers']))
forecast = pd.DataFrame(forecast, index = test.index,columns = ['Prediction'])
```

```
Performing stepwise search to minimize aic

/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/metaestimators.py:121: Futu
reWarning: if_delegate_has_method was deprecated in version 1.1 and will be removed in version 1.3. Use
available_if instead.
  warnings.warn(
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/metaestimators.py:121: Futu
reWarning: if_delegate_has_method was deprecated in version 1.1 and will be removed in version 1.3. Use
available_if instead.
  warnings.warn(
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/metaestimators.py:121: Futu
reWarning: if_delegate_has_method was deprecated in version 1.1 and will be removed in version 1.3. Use
available_if instead.
  warnings.warn(
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/metaestimators.py:121: Futu
reWarning: if_delegate_has_method was deprecated in version 1.1 and will be removed in version 1.3. Use
available_if instead.
  warnings.warn(
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/metaestimators.py:121: Futu
reWarning: if_delegate_has_method was deprecated in version 1.1 and will be removed in version 1.3. Use
```

In [100]:
```
forecast
```

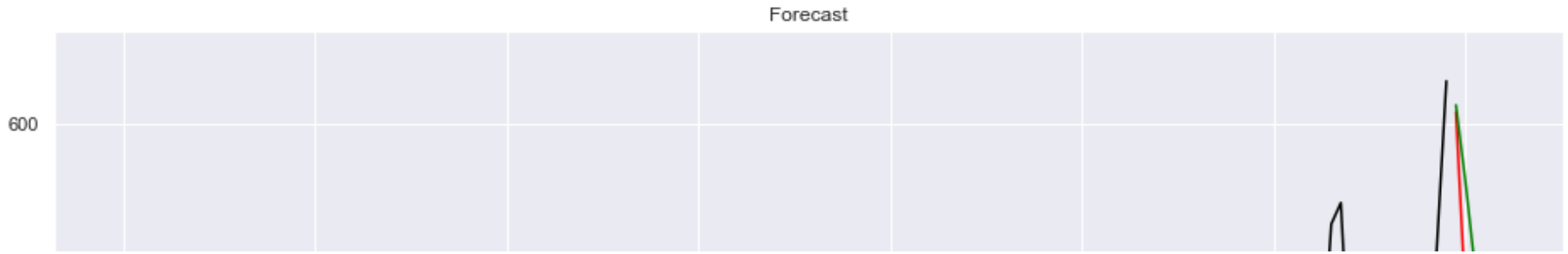
Out[100]:

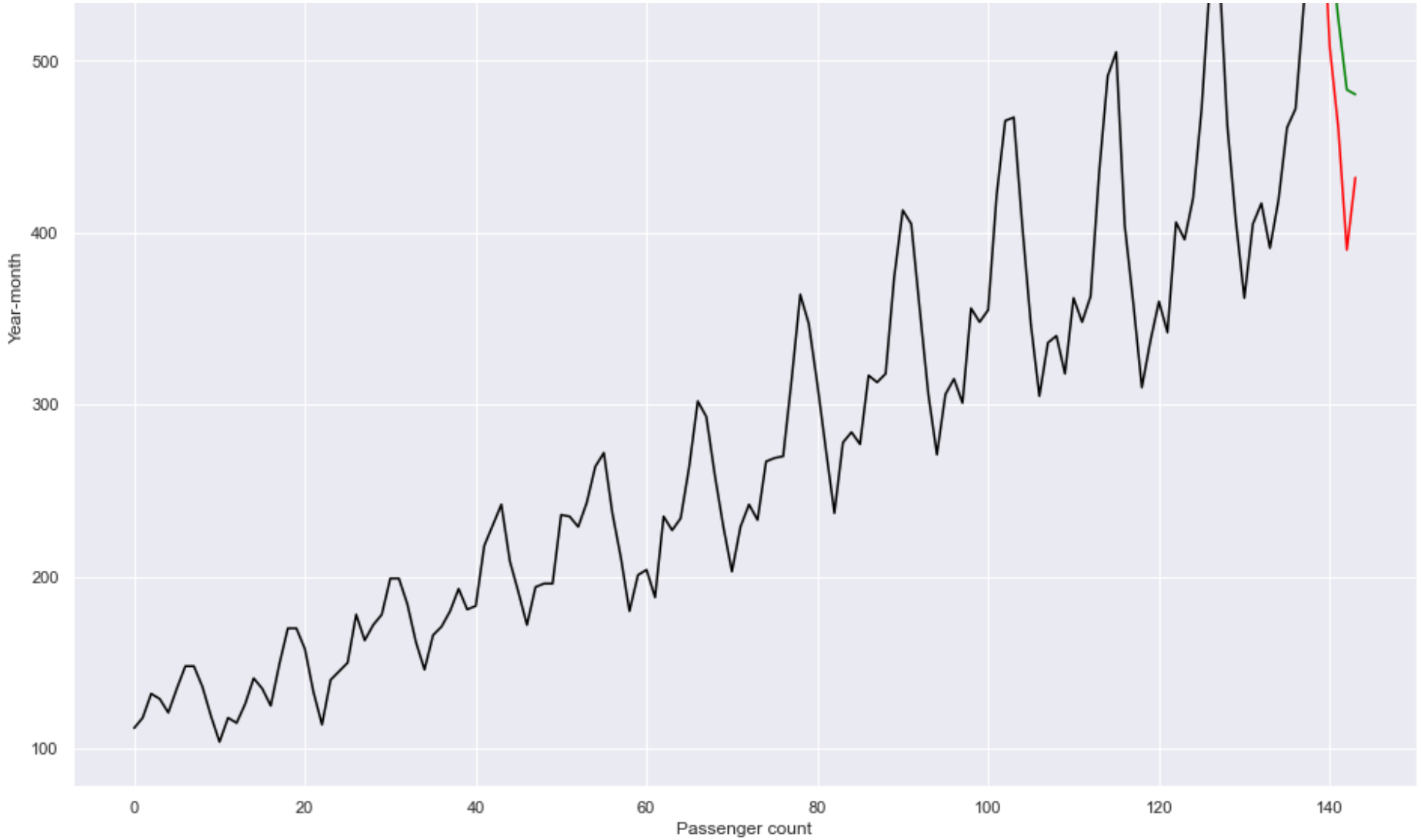
	Prediction
139	609.538031
140	569.574197
141	524.519205
142	483.044010
143	480.304842

In [102]:
```
test['Passengers']
```

Out[102]:
```
139    606
140    508
141    461
142    390
143    432
Name: Passengers, dtype: int64
```

In [101]:
```
plt.plot(train['Passengers'],color = 'black')
plt.plot(test['Passengers'],color = 'red')
plt.plot(forecast, color = 'green')
plt.title("Forecast")
plt.xlabel("Passenger count")
plt.ylabel("Year-month")
plt.show()
```





In [104]:

```
from math import sqrt
from sklearn.metrics import mean_squared_error
rmse = sqrt(mean_squared_error(test['Passengers'],forecast))
print(rmse)
```

```
61.36633338813995
```

In [ ]:

```
1
```