In [ ]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sms
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```python
df = pd.read_csv('airline.csv')
```

In [ ]:
```python
df.head()
```

Out[ ]:

	Month	Thousands of Passengers
0	1949-01	112.0
1	1949-02	118.0
2	1949-03	132.0
3	1949-04	129.0
4	1949-05	121.0

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

Out[ ]:
```
(145, 2)
```

In [ ]:
```python
df.isnull().sum()
```

Out[ ]:
```
Month                      0
Thousands of Passengers    1
dtype: int64
```

One missing value in the column Thousands of Passengers. So we will drop it.

In [ ]:
```python
df.dropna(axis=0, inplace=True)
```

In [ ]:
```python
df.isnull().sum()
```

Out[ ]:
```
Month                      0
Thousands of Passengers    0
dtype: int64
```

Now there is no missing value.

Checking the information about the dataset.

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Month                      144 non-null    object 
 1   Thousands of Passengers    144 non-null    float64
dtypes: float64(1), object(1)
memory usage: 3.4+ KB
```

Month column type is object so we need to convert it into date time.

In [ ]:
```
df['Month'] = pd.to_datetime(df['Month'])
```

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column                     Non-Null Count  Dtype         
---  ------                     --------------  -----         
 0   Month                      144 non-null    datetime64[ns]
 1   Thousands of Passengers    144 non-null    float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 3.4 KB
```

In [ ]:
```
df.head()
```

Out[ ]:

	Month	Thousands of Passengers
0	1949-01-01	112.0
1	1949-02-01	118.0
2	1949-03-01	132.0
3	1949-04-01	129.0
4	1949-05-01	121.0

In [ ]:
```
df.set_index('Month',inplace=True)
```

In [ ]:
```
df.head()
```

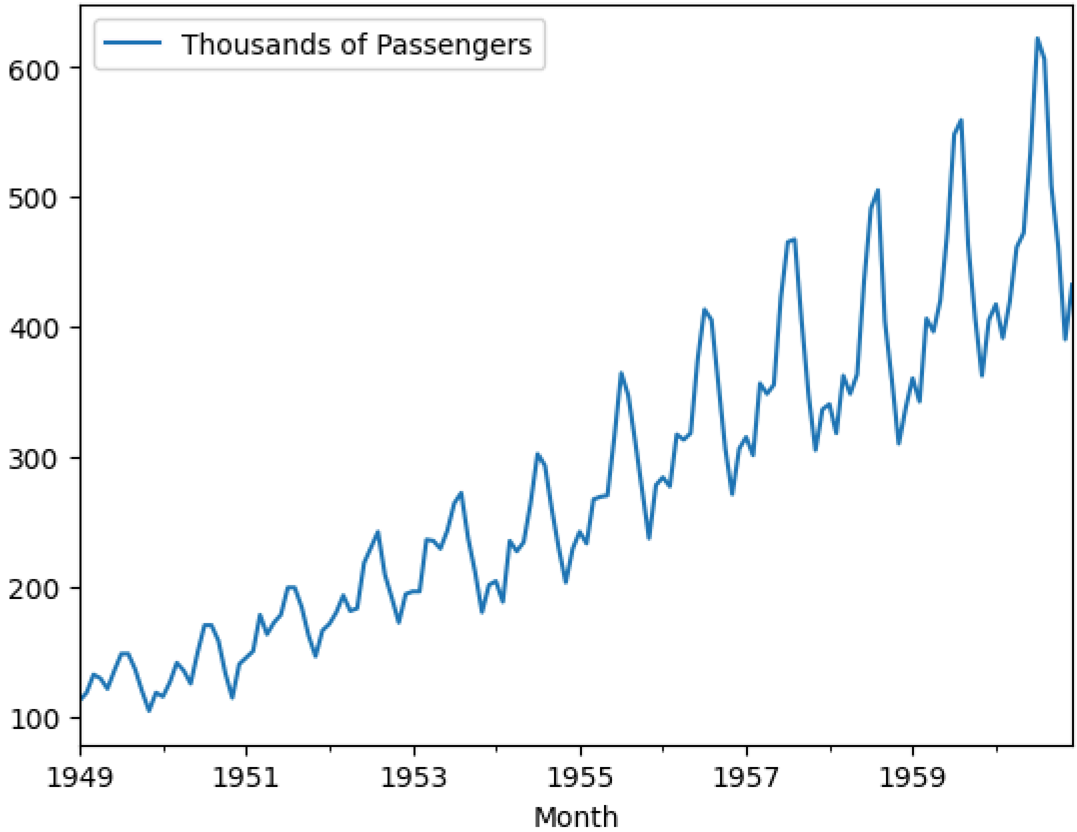
Out[ ]:

	Thousands of Passengers
Month	
1949-01-01	112.0
1949-02-01	118.0
1949-03-01	132.0
1949-04-01	129.0
1949-05-01	121.0

Let us plot the data

In [ ]:
```
df.plot()
```

Out[ ]: <Axes: xlabel='Month'>



The dateset is non-stationary.

Applying Dickey Fuller test.

In [ ]:
```
from statsmodels.tsa.stattools import adfuller
```

In [ ]:
```
def adf_test(series):
    result = adfuller(series)
    print('ADF Statistics: {}'.format(result[0]))
    print('p-value: {}'.format(result[1]))


    if result[1] <= 0.05:
        print('Reject the null hypothesis and hence the data is stationary.')

    else:
        print('The null hypothesis is accpeted and hence the data is non-station
```

In [ ]:
```
adf_test(df['Thousands of Passengers'])
```

```
ADF Statistics: 0.8153688792060482
p-value: 0.991880243437641
The null hypothesis is accpeted and hence the data is non-stationary.
```

Using the differencing technique

In [ ]:
```
df['First Difference'] = df['Thousands of Passengers'] - df['Thousands of Passen
```

In [ ]:
```
df.head()
```

Out[ ]:

	Thousands of Passengers	First Difference
Month		
1949-01-01	112.0	NaN
1949-02-01	118.0	6.0
1949-03-01	132.0	14.0
1949-04-01	129.0	-3.0
1949-05-01	121.0	-8.0

Furhter Applying Dickey-Fuller test

In [ ]:
```
adf_test(df['First Difference'].dropna())
```

```
ADF Statistics: -2.8292668241699994
p-value: 0.0542132902838255
The null hypothesis is accpeted and hence the data is non-stationary.
```

Still the data is non-stationary so further using differencing technique.

In [ ]:
```
df['Second Difference'] = df['First Difference'] - df['First Difference'].shift(
```

In [ ]:
```
adf_test(df['Second Difference'].dropna())
```

```
ADF Statistics: -16.384231542468505
p-value: 2.7328918500142407e-29
Reject the null hypothesis and hence the data is stationary.
```

Since the data is seasonal so we are going to to differencing 12 months because sometime it may happen that ARIMA is not working so we use SARIMA instead.

In [ ]:
```
df['12 Difference'] = df['Thousands of Passengers'] - df['Thousands of Passenger
```

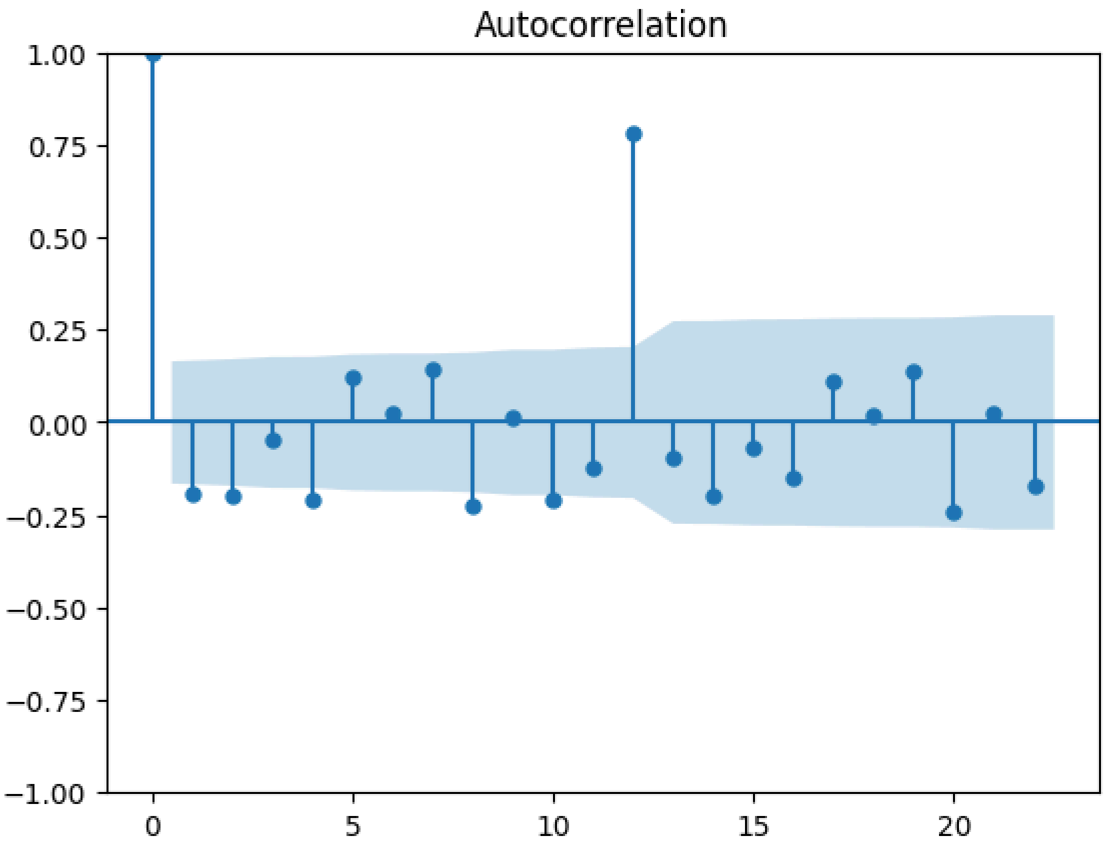
In [ ]:
```
adf_test(df['12 Difference'].dropna())
```

```
ADF Statistics: -3.383020726492481
p-value: 0.011551493085514954
Reject the null hypothesis and hence the data is stationary.
```

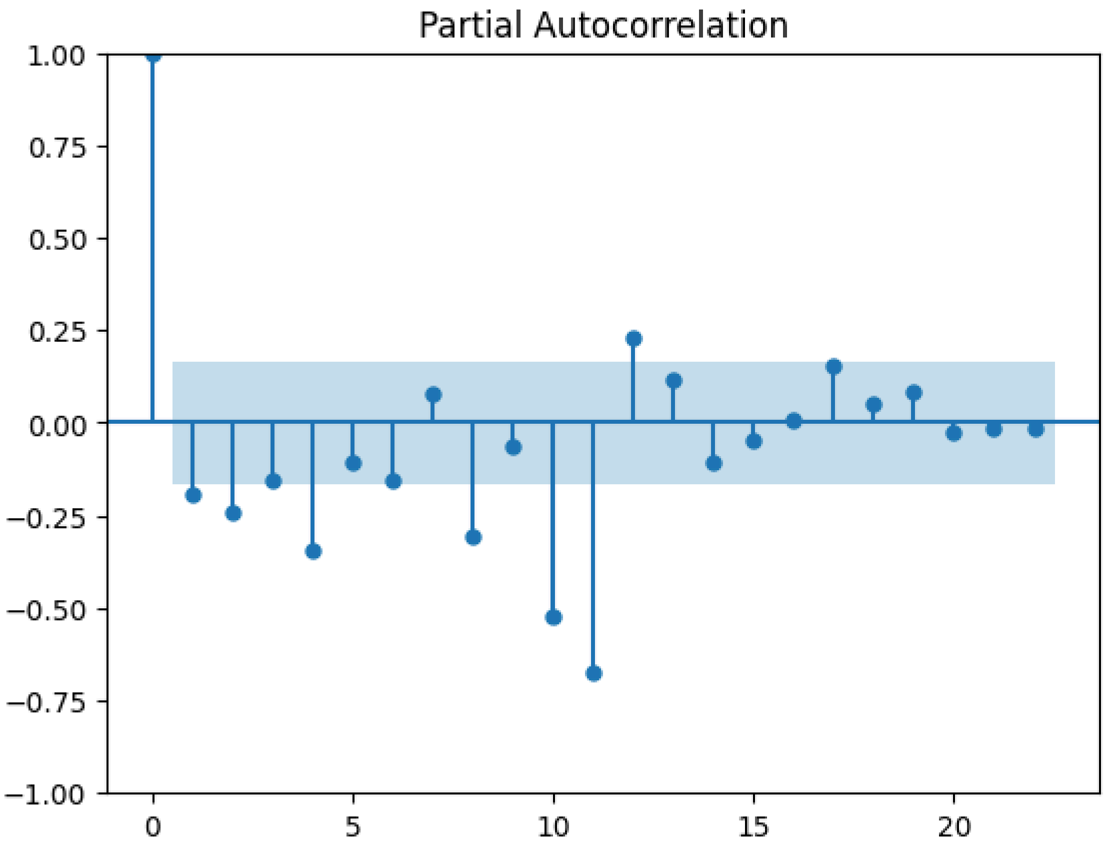
Now let's plot Autocorrelation and Partial correlation graph

In [ ]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

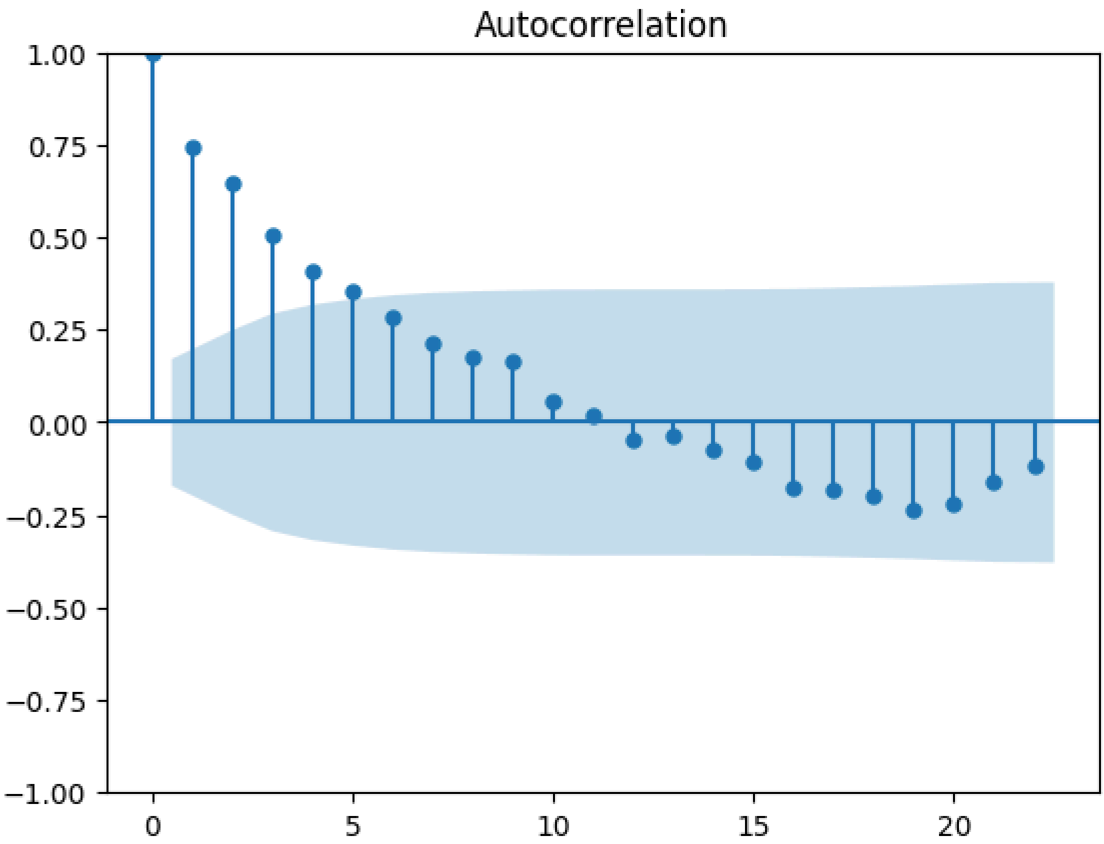
In [ ]:
```
acf = plot_acf(df['Second Difference'].dropna())
```

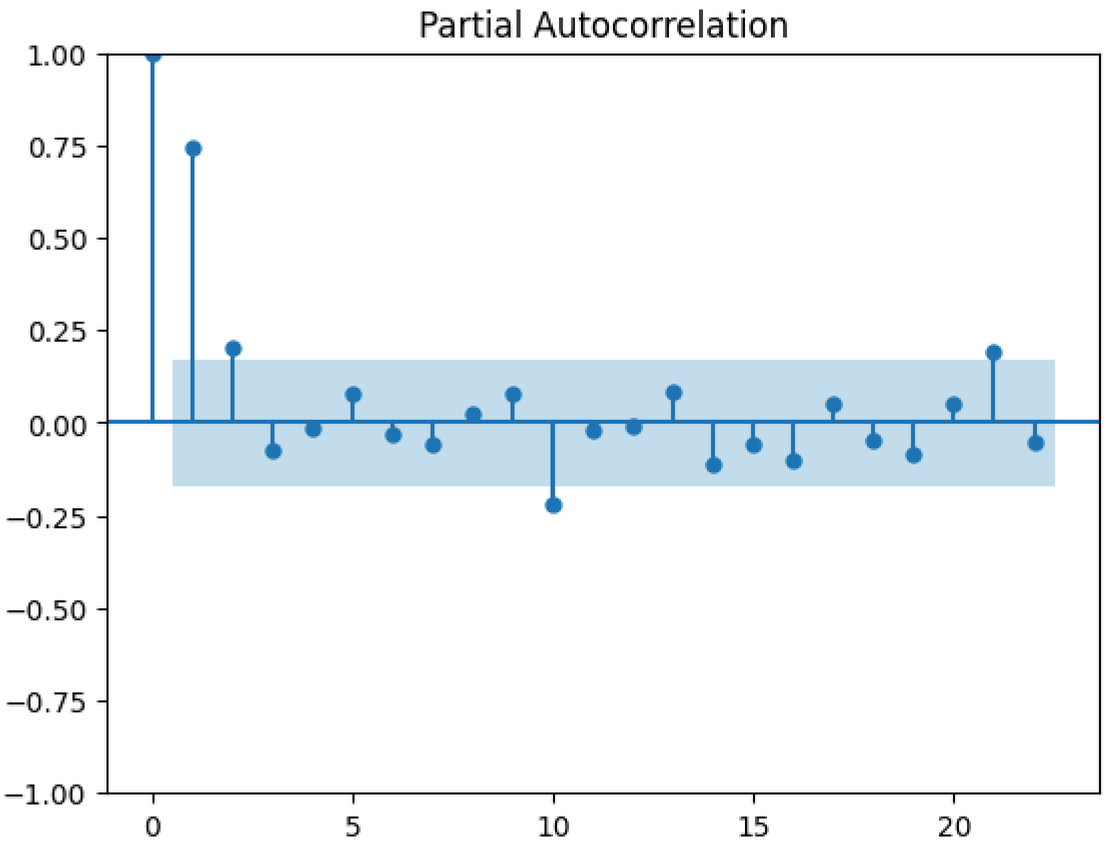



```
In [ ]: pacf = plot_pacf(df['Second Difference'].dropna())
```




```
In [ ]: acf12 = plot_acf(df['12 Difference'].dropna())
        pacf12 = plot_pacf(df['12 Difference'].dropna())
```





Split the data for train and test

In [ ]:
```
df
```

Out[ ]:

	Thousands of Passengers	First Difference	Second Difference	12 Difference
Month				
1949-01-01	112.0	NaN	NaN	NaN
1949-02-01	118.0	6.0	NaN	NaN
1949-03-01	132.0	14.0	8.0	NaN
1949-04-01	129.0	-3.0	-17.0	NaN
1949-05-01	121.0	-8.0	-5.0	NaN
...	...	...	...	...
1960-08-01	606.0	-16.0	-103.0	47.0
1960-09-01	508.0	-98.0	-82.0	45.0
1960-10-01	461.0	-47.0	51.0	54.0
1960-11-01	390.0	-71.0	-24.0	28.0
1960-12-01	432.0	42.0	113.0	27.0

144 rows × 4 columns

In [ ]:

```
from datetime import datetime, timedelta
train_dataset_end = datetime(1955,12,1)
test_data_end = datetime(1960,12,1)
```

In [ ]:

```
train_data = df[:train_dataset_end]
test_data = df[train_dataset_end + timedelta(days=1) : test_data_end]
```

Prediction

In [ ]:

```
pred_start_date = test_data.index[0]
pre_end_date =  test_data.index[-1]
```

In [ ]:

```
train_data.shape
```

Out[ ]:

```
(84, 4)
```

In [ ]:

```
test_data.shape
```

Out[ ]:

```
(60, 4)
```

In [ ]:
```
train_data
```

Out[ ]:

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference
1949-01-01	112.0	NaN	NaN	NaN
1949-02-01	118.0	6.0	NaN	NaN
1949-03-01	132.0	14.0	8.0	NaN
1949-04-01	129.0	-3.0	-17.0	NaN
1949-05-01	121.0	-8.0	-5.0	NaN
...	...	...	...	...
1955-08-01	347.0	-17.0	-66.0	54.0
1955-09-01	312.0	-35.0	-18.0	53.0
1955-10-01	274.0	-38.0	-3.0	45.0
1955-11-01	237.0	-37.0	1.0	34.0
1955-12-01	278.0	41.0	78.0	49.0

84 rows × 4 columns

In [ ]:
```
test_data
```

Out[ ]:

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference
1956-01-01	284.0	6.0	-35.0	42.0
1956-02-01	277.0	-7.0	-13.0	44.0
1956-03-01	317.0	40.0	47.0	50.0
1956-04-01	313.0	-4.0	-44.0	44.0
1956-05-01	318.0	5.0	9.0	48.0
1956-06-01	374.0	56.0	51.0	59.0
1956-07-01	413.0	39.0	-17.0	49.0
1956-08-01	405.0	-8.0	-47.0	58.0
1956-09-01	355.0	-50.0	-42.0	43.0
1956-10-01	306.0	-49.0	1.0	32.0
1956-11-01	271.0	-35.0	14.0	34.0
1956-12-01	306.0	35.0	70.0	28.0
1957-01-01	315.0	9.0	-26.0	31.0
1957-02-01	301.0	-14.0	-23.0	24.0
1957-03-01	356.0	55.0	69.0	39.0
1957-04-01	348.0	-8.0	-63.0	35.0
1957-05-01	355.0	7.0	15.0	37.0
1957-06-01	422.0	67.0	60.0	48.0
1957-07-01	465.0	43.0	-24.0	52.0
1957-08-01	467.0	2.0	-41.0	62.0

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference
1957-09-01	404.0	-63.0	-65.0	49.0
1957-10-01	347.0	-57.0	6.0	41.0
1957-11-01	305.0	-42.0	15.0	34.0
1957-12-01	336.0	31.0	73.0	30.0
1958-01-01	340.0	4.0	-27.0	25.0
1958-02-01	318.0	-22.0	-26.0	17.0
1958-03-01	362.0	44.0	66.0	6.0
1958-04-01	348.0	-14.0	-58.0	0.0
1958-05-01	363.0	15.0	29.0	8.0
1958-06-01	435.0	72.0	57.0	13.0
1958-07-01	491.0	56.0	-16.0	26.0
1958-08-01	505.0	14.0	-42.0	38.0
1958-09-01	404.0	-101.0	-115.0	0.0
1958-10-01	359.0	-45.0	56.0	12.0
1958-11-01	310.0	-49.0	-4.0	5.0
1958-12-01	337.0	27.0	76.0	1.0
1959-01-01	360.0	23.0	-4.0	20.0
1959-02-01	342.0	-18.0	-41.0	24.0
1959-03-01	406.0	64.0	82.0	44.0
1959-04-01	396.0	-10.0	-74.0	48.0

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference
1959-05-01	420.0	24.0	34.0	57.0
1959-06-01	472.0	52.0	28.0	37.0
1959-07-01	548.0	76.0	24.0	57.0
1959-08-01	559.0	11.0	-65.0	54.0
1959-09-01	463.0	-96.0	-107.0	59.0
1959-10-01	407.0	-56.0	40.0	48.0
1959-11-01	362.0	-45.0	11.0	52.0
1959-12-01	405.0	43.0	88.0	68.0
1960-01-01	417.0	12.0	-31.0	57.0
1960-02-01	391.0	-26.0	-38.0	49.0
1960-03-01	419.0	28.0	54.0	13.0
1960-04-01	461.0	42.0	14.0	65.0
1960-05-01	472.0	11.0	-31.0	52.0
1960-06-01	535.0	63.0	52.0	63.0
1960-07-01	622.0	87.0	24.0	74.0
1960-08-01	606.0	-16.0	-103.0	47.0
1960-09-01	508.0	-98.0	-82.0	45.0
1960-10-01	461.0	-47.0	51.0	54.0
1960-11-01	390.0	-71.0	-24.0	28.0
1960-12-01	432.0	42.0	113.0	27.0

Create an ARIMA model

In [ ]:
```
from statsmodels.tsa.arima.model import ARIMA
```

In [ ]:
```
arima = ARIMA(train_data['Thousands of Passengers'], order=(0,2,0))
```

```
c:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provide
d, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
c:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provide
d, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
c:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provide
d, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```

Fit the model

In [ ]:
```
arima_fit = arima.fit()
```

Summary of the Model

In [ ]:
```
arima_fit.summary()
```

Out[ ]:

SARIMAX Results

Dep. Variable:	Thousands of Passengers	**No. Observations:**	84
Model:	ARIMA(0, 2, 0)	**Log Likelihood**	-385.792
Date:	Sun, 04 Aug 2024	**AIC**	773.584
Time:	16:00:33	**BIC**	775.991
Sample:	01-01-1949	**HQIC**	774.550
	- 12-01-1955		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
sigma2	714.5859	102.414	6.977	0.000	513.858	915.314

Ljung-Box (L1) (Q):	4.59	**Jarque-Bera (JB):**	1.74
Prob(Q):	0.03	**Prob(JB):**	0.42
Heteroskedasticity (H):	3.19	**Skew:**	0.31
Prob(H) (two-sided):	0.00	**Kurtosis:**	3.36

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

In [ ]:

```
pred_start_date = test_data.index[0]
pre_end_date = test_data.index[-1]
print('The start date is :',pred_start_date)
print('The end date is :',pre_end_date)
```

```
The start date is : 1956-01-01 00:00:00
The end date is : 1960-12-01 00:00:00
```

Make Prediction

In [ ]:

```
pred  = arima_fit.predict(start=pred_start_date, end=pre_end_date)
```

In [ ]:

```
pred
```

```
Out[ ]: 1956-01-01     319.0
        1956-02-01     360.0
        1956-03-01     401.0
        1956-04-01     442.0
        1956-05-01     483.0
        1956-06-01     524.0
        1956-07-01     565.0
        1956-08-01     606.0
        1956-09-01     647.0
        1956-10-01     688.0
        1956-11-01     729.0
        1956-12-01     770.0
        1957-01-01     811.0
        1957-02-01     852.0
        1957-03-01     893.0
        1957-04-01     934.0
        1957-05-01     975.0
        1957-06-01    1016.0
        1957-07-01    1057.0
        1957-08-01    1098.0
        1957-09-01    1139.0
        1957-10-01    1180.0
        1957-11-01    1221.0
        1957-12-01    1262.0
        1958-01-01    1303.0
        1958-02-01    1344.0
        1958-03-01    1385.0
        1958-04-01    1426.0
        1958-05-01    1467.0
        1958-06-01    1508.0
        1958-07-01    1549.0
        1958-08-01    1590.0
        1958-09-01    1631.0
        1958-10-01    1672.0
        1958-11-01    1713.0
        1958-12-01    1754.0
        1959-01-01    1795.0
        1959-02-01    1836.0
        1959-03-01    1877.0
        1959-04-01    1918.0
        1959-05-01    1959.0
        1959-06-01    2000.0
        1959-07-01    2041.0
        1959-08-01    2082.0
        1959-09-01    2123.0
        1959-10-01    2164.0
        1959-11-01    2205.0
        1959-12-01    2246.0
        1960-01-01    2287.0
        1960-02-01    2328.0
        1960-03-01    2369.0
        1960-04-01    2410.0
        1960-05-01    2451.0
        1960-06-01    2492.0
        1960-07-01    2533.0
        1960-08-01    2574.0
        1960-09-01    2615.0
        1960-10-01    2656.0
        1960-11-01    2697.0
```

```
1960-12-01    2738.0
Freq: MS, Name: predicted_mean, dtype: float64
```

Residuals

In [ ]:
```
residuals=test_data['Thousands of Passengers']-pred
```

In [ ]:
```
residuals
```

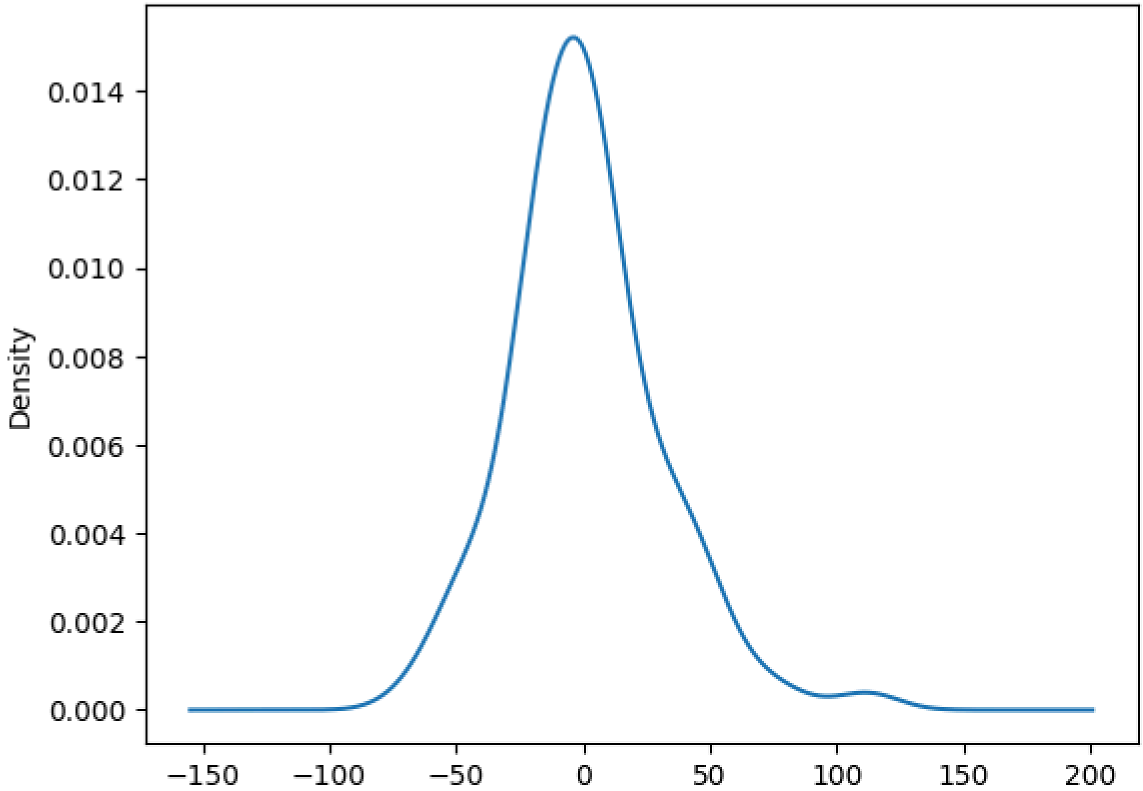
Out[ ]:

```
Month
1956-01-01      -35.0
1956-02-01      -83.0
1956-03-01      -84.0
1956-04-01     -129.0
1956-05-01     -165.0
1956-06-01     -150.0
1956-07-01     -152.0
1956-08-01     -201.0
1956-09-01     -292.0
1956-10-01     -382.0
1956-11-01     -458.0
1956-12-01     -464.0
1957-01-01     -496.0
1957-02-01     -551.0
1957-03-01     -537.0
1957-04-01     -586.0
1957-05-01     -620.0
1957-06-01     -594.0
1957-07-01     -592.0
1957-08-01     -631.0
1957-09-01     -735.0
1957-10-01     -833.0
1957-11-01     -916.0
1957-12-01     -926.0
1958-01-01     -963.0
1958-02-01    -1026.0
1958-03-01    -1023.0
1958-04-01    -1078.0
1958-05-01    -1104.0
1958-06-01    -1073.0
1958-07-01    -1058.0
1958-08-01    -1085.0
1958-09-01    -1227.0
1958-10-01    -1313.0
1958-11-01    -1403.0
1958-12-01    -1417.0
1959-01-01    -1435.0
1959-02-01    -1494.0
1959-03-01    -1471.0
1959-04-01    -1522.0
1959-05-01    -1539.0
1959-06-01    -1528.0
1959-07-01    -1493.0
1959-08-01    -1523.0
1959-09-01    -1660.0
1959-10-01    -1757.0
1959-11-01    -1843.0
1959-12-01    -1841.0
1960-01-01    -1870.0
1960-02-01    -1937.0
1960-03-01    -1950.0
1960-04-01    -1949.0
1960-05-01    -1979.0
1960-06-01    -1957.0
1960-07-01    -1911.0
1960-08-01    -1968.0
1960-09-01    -2107.0
1960-10-01    -2195.0
1960-11-01    -2307.0
```

```
1960-12-01   -2306.0
dtype: float64
```

In [ ]:
```
arima_fit.resid.plot(kind='kde')
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



The plot of residuals follows normal distribution

In [ ]:
```
test_data['Predicted_ARIMA'] = pred
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_18036\284031954.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data['Predicted_ARIMA'] = pred
```

In [ ]:
```
test_data.head()
```

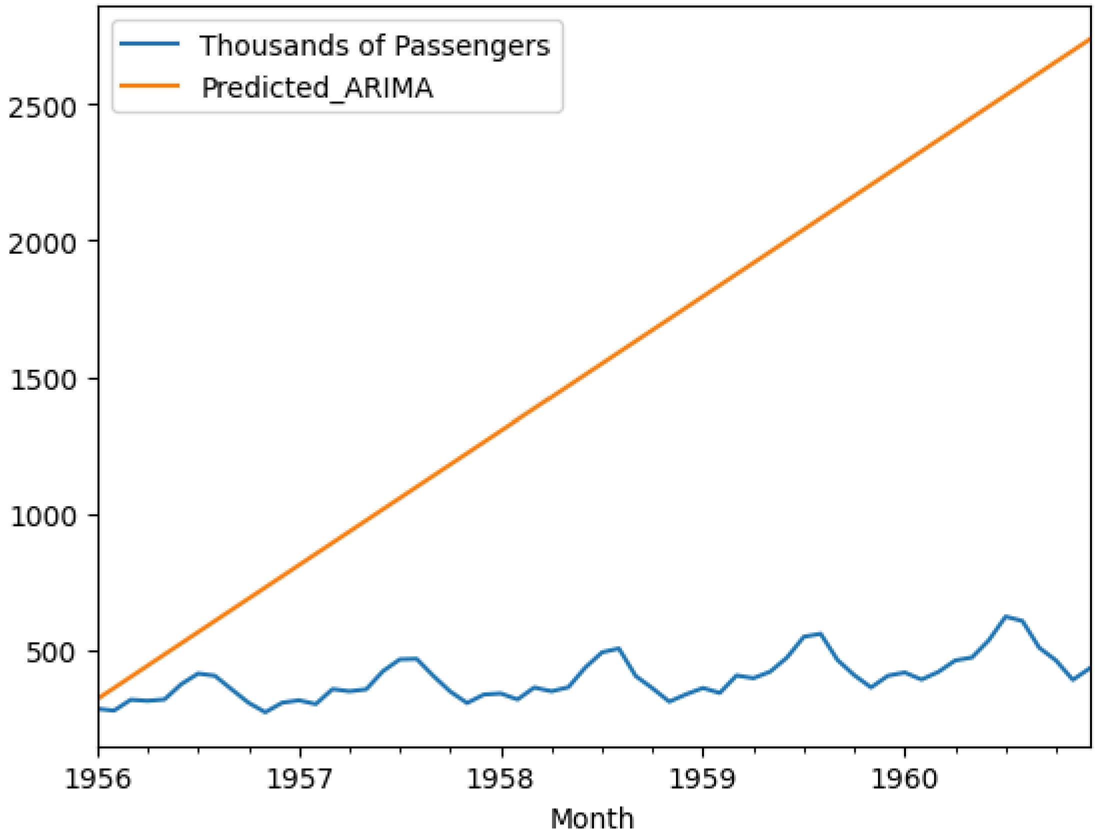
Out[ ]:

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference	Predicted_ARIMA
1956-01-01	284.0	6.0	-35.0	42.0	319.0
1956-02-01	277.0	-7.0	-13.0	44.0	360.0
1956-03-01	317.0	40.0	47.0	50.0	401.0
1956-04-01	313.0	-4.0	-44.0	44.0	442.0
1956-05-01	318.0	5.0	9.0	48.0	483.0

Let us plot Thousands of Passengers vs Predicted_ARIMA

In [ ]:
```
test_data[['Thousands of Passengers','Predicted_ARIMA']].plot()
```

Out[ ]: <Axes: xlabel='Month'>



This seems very poor prediction.

Let us create a SARIMAX Model now.

In [ ]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

In [ ]:
```
sarima = SARIMAX(train_data['Thousands of Passengers'], order=(3,0,5), seasonal_
```

```
c:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provide
d, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
c:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: No frequency information was provide
d, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```

Fit the model

In [ ]:
```
sarima_fit = sarima.fit()
```

```
c:\Users\User\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmode
ls\base\model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed
to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Summary

In [ ]:
```
sarima_fit.summary()
```

Out[ ]:

SARIMAX Results

Dep. Variable:	Thousands of Passengers	**No. Observations:**	84
Model:	SARIMAX(3, 0, 5)x(0, 1, [], 12)	**Log Likelihood**	-265.240
Date:	Sun, 04 Aug 2024	**AIC**	548.481
Time:	16:00:33	**BIC**	568.971
Sample:	01-01-1949	**HQIC**	556.638
	- 12-01-1955		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	0.5983	0.937	0.638	0.523	-1.239	2.436
ar.L2	0.8311	0.232	3.581	0.000	0.376	1.286
ar.L3	-0.4525	0.894	-0.506	0.613	-2.204	1.299
ma.L1	0.1837	1.165	0.158	0.875	-2.099	2.467
ma.L2	-0.5341	1.263	-0.423	0.672	-3.009	1.940
ma.L3	-0.0986	0.384	-0.257	0.798	-0.852	0.655
ma.L4	-0.1273	0.338	-0.377	0.706	-0.789	0.535
ma.L5	0.2471	0.357	0.693	0.489	-0.452	0.947
sigma2	87.7323	81.217	1.080	0.280	-71.451	246.915

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	2.68
Prob(Q):	0.88	**Prob(JB):**	0.26
Heteroskedasticity (H):	2.05	**Skew:**	0.46
Prob(H) (two-sided):	0.09	**Kurtosis:**	2.77

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

In [ ]:

```
test_data.head()
```

Out[ ]:

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference	Predicted_ARIMA
1956-01-01	284.0	6.0	-35.0	42.0	319.0
1956-02-01	277.0	-7.0	-13.0	44.0	360.0
1956-03-01	317.0	40.0	47.0	50.0	401.0
1956-04-01	313.0	-4.0	-44.0	44.0	442.0
1956-05-01	318.0	5.0	9.0	48.0	483.0

Prediction

In [ ]:

```
pred_start_date = test_data.index[0]
pre_end_date = test_data.index[-1]
print('The start date is :',pred_start_date)
print('The end date is :',pre_end_date)
```

```
The start date is : 1956-01-01 00:00:00
The end date is : 1960-12-01 00:00:00
```

In [ ]:

```
pred_Sarima=sarima_fit.predict(start=datetime(1956,6,6),end=datetime(1960,12,1))
```

Residuals

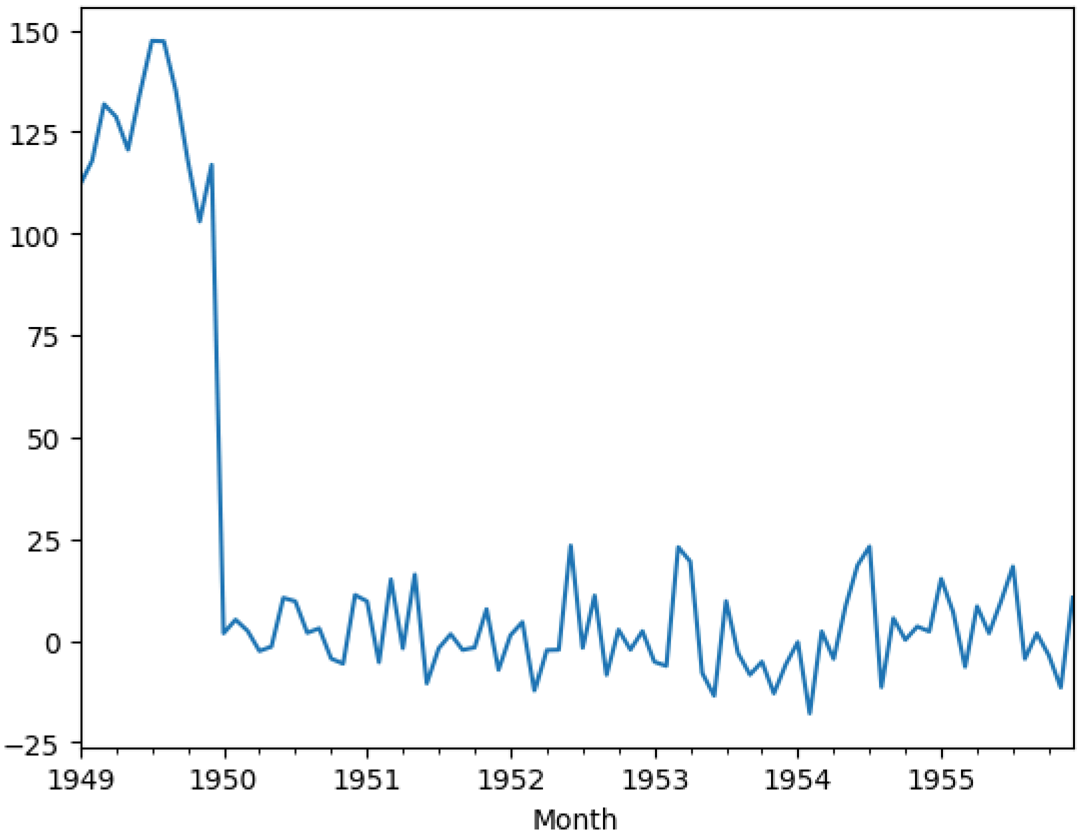
In [ ]:

```
residuals=test_data['Thousands of Passengers']-pred_Sarima
```

In [ ]:

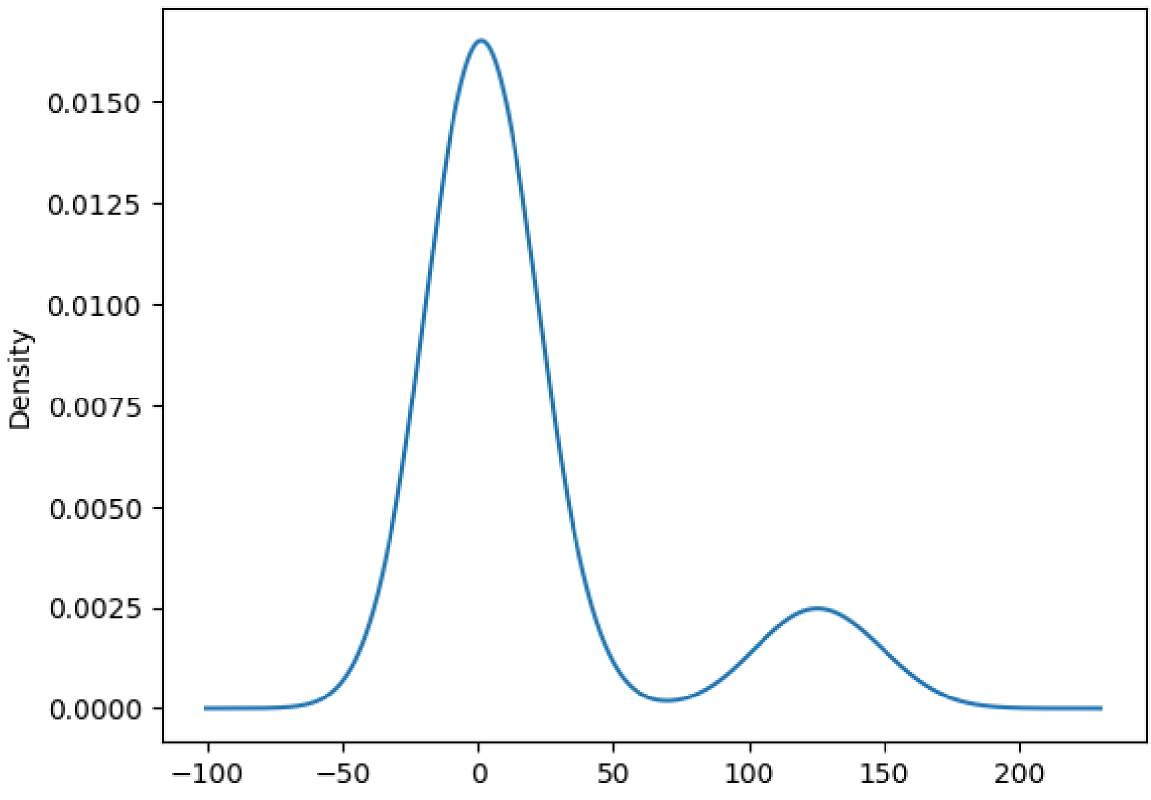
```
sarima_fit.resid.plot()
```

Out[ ]: <Axes: xlabel='Month'>



In [ ]:
```
sarima_fit.resid.plot(kind='kde')
```

Out[ ]: <Axes: ylabel='Density'>



In [ ]:
```
test_data['Predicted_SARIMA']=pred_Sarima
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_18036\1367177785.py:1: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data['Predicted_SARIMA']=pred_Sarima
```

In [ ]:
```
test_data.head()
```

Out[ ]:

Month	Thousands of Passengers	First Difference	Second Difference	12 Difference	Predicted_ARIMA	Predicted_SAR
1956-01-01	284.0	6.0	-35.0	42.0	319.0	
1956-02-01	277.0	-7.0	-13.0	44.0	360.0	
1956-03-01	317.0	40.0	47.0	50.0	401.0	
1956-04-01	313.0	-4.0	-44.0	44.0	442.0	
1956-05-01	318.0	5.0	9.0	48.0	483.0	

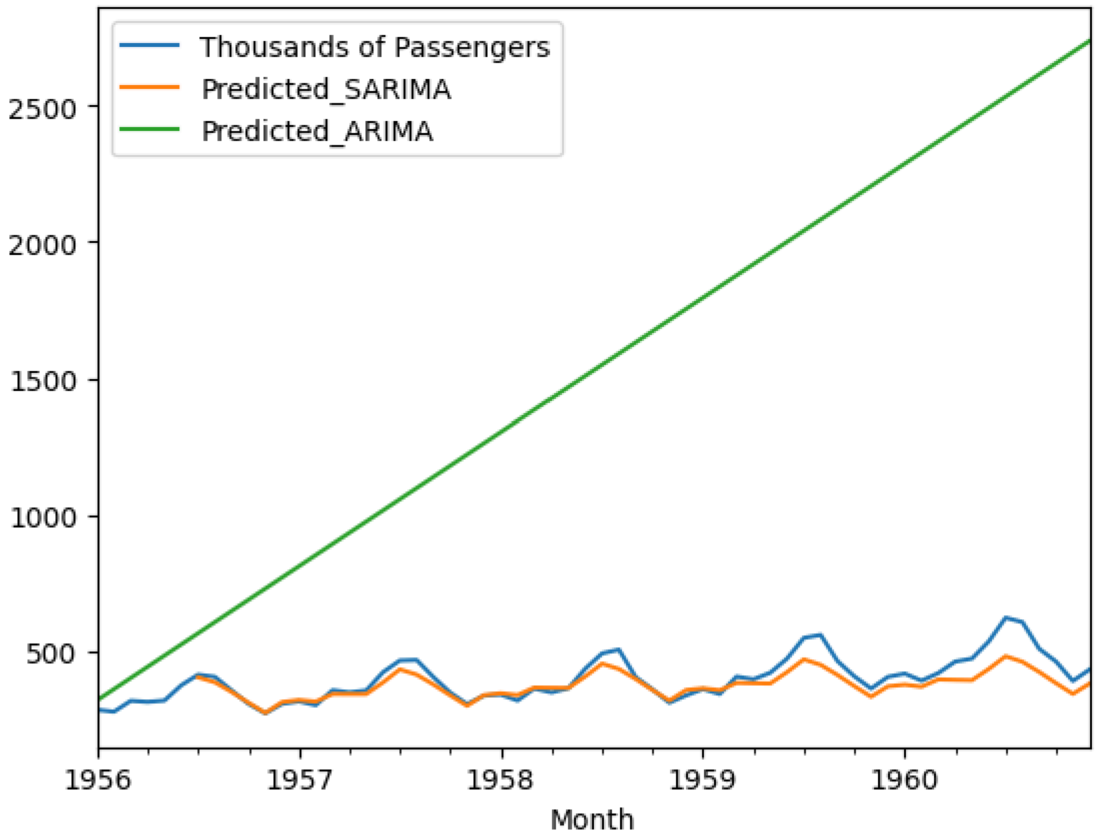
In [ ]:
```
test_data.shape
```

Out[ ]: (60, 6)

Let us plot Thousands of Passengers vs Predicted_ARIMA vs Predicted_SARIMA

In [ ]:
```
test_data[['Thousands of Passengers','Predicted_SARIMA','Predicted_ARIMA']].plot
```

Out[ ]: <Axes: xlabel='Month'>



SARIMAX model fits well to the test data.

Let us try with Random Forest

In [ ]:
```
df1 = pd.read_csv('airline.csv')
```

In [ ]:
```
df1.head()
```

Out[ ]:

	Month	Thousands of Passengers
0	1949-01	112.0
1	1949-02	118.0
2	1949-03	132.0
3	1949-04	129.0
4	1949-05	121.0

In [ ]:
```
df1.shape
```

Out[ ]:
```
(145, 2)
```

In [ ]:
```
df1.isnull().sum()
```

Out[ ]:
```
Month                       0
Thousands of Passengers     1
dtype: int64
```

In [ ]:
```
df1.dropna(axis=0, inplace=True)
```

In [ ]:
```
df1.shape
```

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

In [ ]:
```
df1['Month'] = pd.to_datetime(df1['Month'])
df1.set_index('Month', inplace=True)
print(df1.head())
```

```
            Thousands of Passengers
Month
1949-01-01                    112.0
1949-02-01                    118.0
1949-03-01                    132.0
1949-04-01                    129.0
1949-05-01                    121.0
```

In [ ]:
```
df1.columns
```

Out[ ]:
```
Index(['Thousands of Passengers'], dtype='object')
```

Let us create features

In [ ]:
```
def create_features(df1):
    df1['Month'] = df1.index.month
    df1['Year'] = df1.index.year
    for i in range(1, 13):
        df1[f'Lag_{i}'] = df1['Thousands of Passengers'].shift(i)
    df1.dropna(inplace=True)
    return df1

df1 = create_features(df1)
print(df1.head())
```

```
            Thousands of Passengers  Month  Year  Lag_1  Lag_2  Lag_3  Lag_4  \
Month
1950-01-01                    115.0      1  1950  118.0  104.0  119.0  136.0
1950-02-01                    126.0      2  1950  115.0  118.0  104.0  119.0
1950-03-01                    141.0      3  1950  126.0  115.0  118.0  104.0
1950-04-01                    135.0      4  1950  141.0  126.0  115.0  118.0
1950-05-01                    125.0      5  1950  135.0  141.0  126.0  115.0

            Lag_5  Lag_6  Lag_7  Lag_8  Lag_9  Lag_10  Lag_11  Lag_12
Month
1950-01-01  148.0  148.0  135.0  121.0  129.0   132.0   118.0   112.0
1950-02-01  136.0  148.0  148.0  135.0  121.0   129.0   132.0   118.0
1950-03-01  119.0  136.0  148.0  148.0  135.0   121.0   129.0   132.0
1950-04-01  104.0  119.0  136.0  148.0  148.0   135.0   121.0   129.0
1950-05-01  118.0  104.0  119.0  136.0  148.0   148.0   135.0   121.0
```

Train Test Split

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
X = df1.drop(columns='Thousands of Passengers', axis=1)
y = df1['Thousands of Passengers']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle
```

In [ ]:
```
X.shape
```
Out[ ]:
```
(132, 14)
```
In [ ]:
```
X_train.shape
```
Out[ ]:
```
(105, 14)
```
In [ ]:
```
y.shape
```
Out[ ]:
```
(132,)
```
In [ ]:
```
X.head()
```
Out[ ]:

Month	Month	Year	Lag_1	Lag_2	Lag_3	Lag_4	Lag_5	Lag_6	Lag_7	Lag_8	Lag_9
1950-01-01	1	1950	118.0	104.0	119.0	136.0	148.0	148.0	135.0	121.0	129.0
1950-02-01	2	1950	115.0	118.0	104.0	119.0	136.0	148.0	148.0	135.0	121.0
1950-03-01	3	1950	126.0	115.0	118.0	104.0	119.0	136.0	148.0	148.0	135.0
1950-04-01	4	1950	141.0	126.0	115.0	118.0	104.0	119.0	136.0	148.0	148.0
1950-05-01	5	1950	135.0	141.0	126.0	115.0	118.0	104.0	119.0	136.0	148.0

In [ ]:
```
y.head()
```
Out[ ]:
```
Month
1950-01-01    115.0
1950-02-01    126.0
1950-03-01    141.0
1950-04-01    135.0
1950-05-01    125.0
Name: Thousands of Passengers, dtype: float64
```
In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
```
In [ ]:
```
rf =  RandomForestRegressor(n_estimators=100, random_state=42)
```
In [ ]:
```
rf.fit(X_train, y_train)
```
Out[ ]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```
In [ ]:
```
rf_pred = rf.predict(X_test)
```

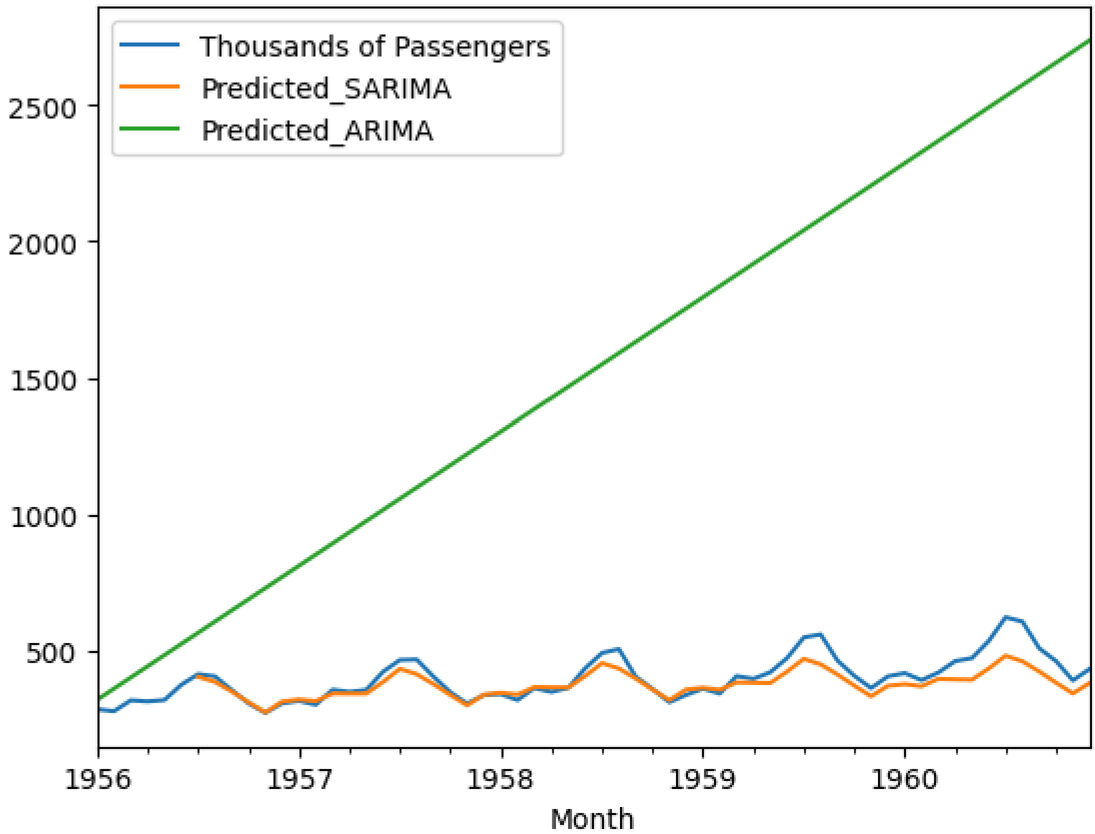
In [ ]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
```

In [ ]:
```
print('MSE :', mean_squared_error(y_test,rf_pred))
print('MAPE :', mean_absolute_percentage_error(y_test,rf_pred))
```

```
MSE : 2274.161266666666
MAPE : 0.07027829414813154
```

In [ ]:
```
test_data[['Thousands of Passengers','Predicted_SARIMA','Predicted_ARIMA']].plot
```

Out[ ]: <Axes: xlabel='Month'>



In [ ]: