In [1]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\Downloads\\archive (1)\\KO.csv")
a
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1962-01-02	0.263021	0.270182	0.263021	0.263021	0.048145	806400
1	1962-01-03	0.259115	0.259115	0.253255	0.257161	0.047072	1574400
2	1962-01-04	0.257813	0.261068	0.257813	0.259115	0.047430	844800
3	1962-01-05	0.259115	0.262370	0.252604	0.253255	0.046357	1420800
4	1962-01-08	0.251302	0.251302	0.245768	0.250651	0.045881	2035200
...	...	...	...	...	...	...	...
15584	2023-11-29	58.580002	58.669998	58.099998	58.230000	57.770000	11263600
15585	2023-11-30	57.959999	58.459999	57.599998	58.439999	58.439999	22727500
15586	2023-12-01	58.270000	58.689999	58.240002	58.639999	58.639999	15369600
15587	2023-12-04	58.590000	58.959999	58.439999	58.570000	58.570000	14942200
15588	2023-12-05	58.549999	58.830002	58.419998	58.660000	58.660000	11880000

15589 rows × 7 columns

In [3]:
```
a.head(10)
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1962-01-02	0.263021	0.270182	0.263021	0.263021	0.048145	806400
1	1962-01-03	0.259115	0.259115	0.253255	0.257161	0.047072	1574400
2	1962-01-04	0.257813	0.261068	0.257813	0.259115	0.047430	844800
3	1962-01-05	0.259115	0.262370	0.252604	0.253255	0.046357	1420800
4	1962-01-08	0.251302	0.251302	0.245768	0.250651	0.045881	2035200
5	1962-01-09	0.250651	0.256510	0.248698	0.255208	0.046715	960000
6	1962-01-10	0.255208	0.260091	0.252604	0.256510	0.046953	1612800
7	1962-01-11	0.256510	0.259115	0.255208	0.259115	0.047430	614400
8	1962-01-12	0.259115	0.259115	0.254557	0.257161	0.047072	883200
9	1962-01-15	0.256510	0.256510	0.253906	0.254557	0.046595	614400

In [4]:
```
a.tail(10)
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
15579	2023-11-21	57.459999	58.040001	57.330002	58.029999	57.571579	13891600
15580	2023-11-22	58.259998	58.540001	58.130001	58.419998	57.958496	11320600
15581	2023-11-24	58.459999	58.750000	58.340000	58.570000	58.107315	4816000
15582	2023-11-27	58.540001	58.689999	58.270000	58.459999	57.998184	16246500
15583	2023-11-28	58.400002	58.830002	58.360001	58.580002	58.117237	13739600
15584	2023-11-29	58.580002	58.669998	58.099998	58.230000	57.770000	11263600
15585	2023-11-30	57.959999	58.459999	57.599998	58.439999	58.439999	22727500
15586	2023-12-01	58.270000	58.689999	58.240002	58.639999	58.639999	15369600
15587	2023-12-04	58.590000	58.959999	58.439999	58.570000	58.570000	14942200
15588	2023-12-05	58.549999	58.830002	58.419998	58.660000	58.660000	11880000

In [5]:
```
a.describe()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	15589.000000	15589.000000	15589.000000	15589.000000	15589.000000	1.558900e+04
mean	17.471842	17.613457	17.326797	17.475785	12.222739	9.215234e+06
std	18.424374	18.557859	18.285563	18.425274	15.662293	7.941726e+06
min	0.192708	0.193359	0.182292	0.192057	0.035643	7.680000e+04
25%	0.875000	0.882813	0.869792	0.875000	0.231685	2.985600e+06
50%	10.250000	10.343750	10.156250	10.250000	4.885883	7.866000e+06
75%	30.625000	30.937500	30.250000	30.605000	17.404667	1.315200e+07
max	67.000000	67.199997	65.720001	66.209999	62.817848	1.241690e+08

In [6]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15589 entries, 0 to 15588
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       15589 non-null  object 
 1   Open       15589 non-null  float64
 2   High       15589 non-null  float64
 3   Low        15589 non-null  float64
 4   Close      15589 non-null  float64
 5   Adj Close  15589 non-null  float64
 6   Volume     15589 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 852.6+ KB
```

In [7]:
```
a.isna().sum()
```

Out[7]:
```
Date         0
Open         0
High         0
```

```
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [8]:

```
from datetime import datetime
a['year'] = pd.DatetimeIndex(a['Date']).year
a['month'] = pd.DatetimeIndex(a['Date']).month
a.head(10)
```

Out[8]:

	Date	Open	High	Low	Close	Adj Close	Volume	year	month
0	1962-01-02	0.263021	0.270182	0.263021	0.263021	0.048145	806400	1962	1
1	1962-01-03	0.259115	0.259115	0.253255	0.257161	0.047072	1574400	1962	1
2	1962-01-04	0.257813	0.261068	0.257813	0.259115	0.047430	844800	1962	1
3	1962-01-05	0.259115	0.262370	0.252604	0.253255	0.046357	1420800	1962	1
4	1962-01-08	0.251302	0.251302	0.245768	0.250651	0.045881	2035200	1962	1
5	1962-01-09	0.250651	0.256510	0.248698	0.255208	0.046715	960000	1962	1
6	1962-01-10	0.255208	0.260091	0.252604	0.256510	0.046953	1612800	1962	1
7	1962-01-11	0.256510	0.259115	0.255208	0.259115	0.047430	614400	1962	1
8	1962-01-12	0.259115	0.259115	0.254557	0.257161	0.047072	883200	1962	1
9	1962-01-15	0.256510	0.256510	0.253906	0.254557	0.046595	614400	1962	1

In [9]:

```
b=a.drop(['Date','month'],axis=1)
b
```

Out[9]:

	Open	High	Low	Close	Adj Close	Volume	year
0	0.263021	0.270182	0.263021	0.263021	0.048145	806400	1962
1	0.259115	0.259115	0.253255	0.257161	0.047072	1574400	1962
2	0.257813	0.261068	0.257813	0.259115	0.047430	844800	1962
3	0.259115	0.262370	0.252604	0.253255	0.046357	1420800	1962
4	0.251302	0.251302	0.245768	0.250651	0.045881	2035200	1962
...	...	...	...	...	...	...	...
15584	58.580002	58.669998	58.099998	58.230000	57.770000	11263600	2023
15585	57.959999	58.459999	57.599998	58.439999	58.439999	22727500	2023
15586	58.270000	58.689999	58.240002	58.639999	58.639999	15369600	2023
15587	58.590000	58.959999	58.439999	58.570000	58.570000	14942200	2023
15588	58.549999	58.830002	58.419998	58.660000	58.660000	11880000	2023

15589 rows × 7 columns

In [10]:

```
c=pd.get_dummies(b,dtype=int)
c
```

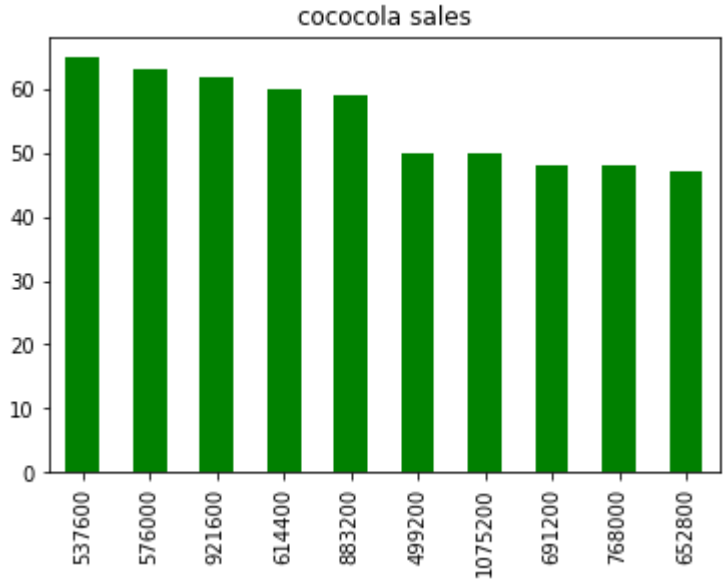
Out[10]:

	Open	High	Low	Close	Adj Close	Volume	year
0	0.263021	0.270182	0.263021	0.263021	0.048145	806400	1962
1	0.259115	0.259115	0.253255	0.257161	0.047072	1574400	1962
2	0.257813	0.261068	0.257813	0.259115	0.047430	844800	1962
3	0.259115	0.262370	0.252604	0.253255	0.046357	1420800	1962
4	0.251302	0.251302	0.245768	0.250651	0.045881	2035200	1962
...	...	...	...	...	...	...	...
15584	58.580002	58.669998	58.099998	58.230000	57.770000	11263600	2023
15585	57.959999	58.459999	57.599998	58.439999	58.439999	22727500	2023
15586	58.270000	58.689999	58.240002	58.639999	58.639999	15369600	2023
15587	58.590000	58.959999	58.439999	58.570000	58.570000	14942200	2023
15588	58.549999	58.830002	58.419998	58.660000	58.660000	11880000	2023

15589 rows × 7 columns

In [11]:

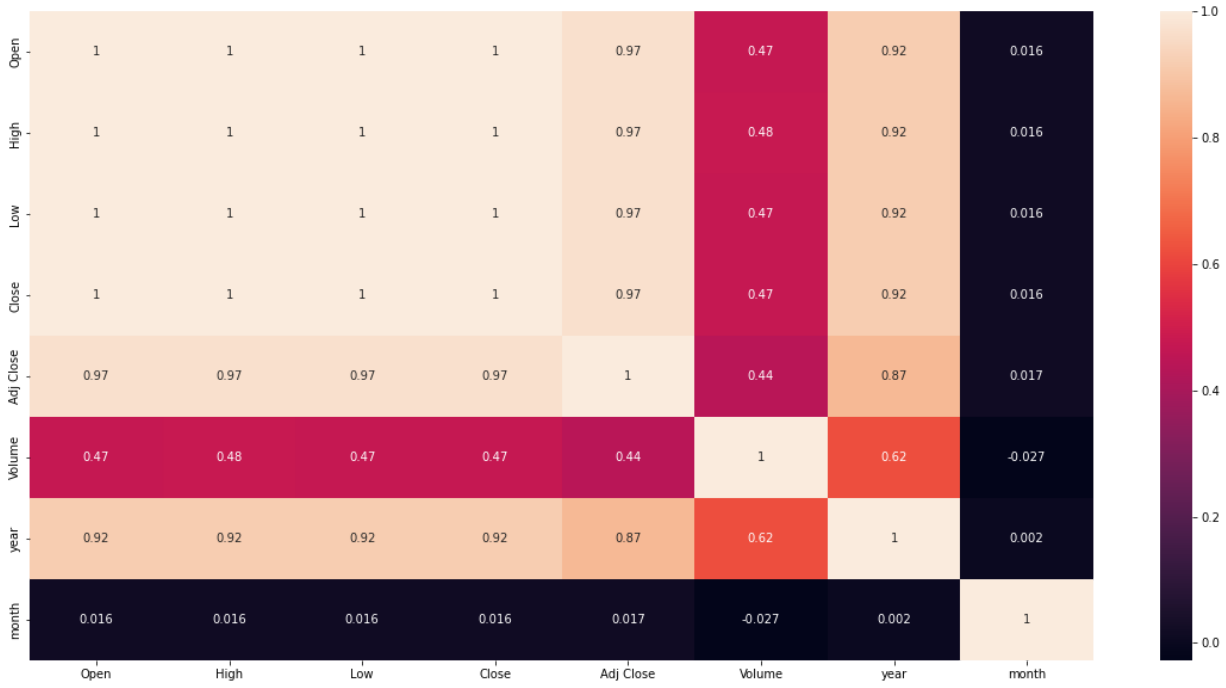
```
import seaborn as sns
import matplotlib.pyplot as plt
c['Volume'].value_counts().head(10).plot(kind='bar',color='g')
plt.title('cococola sales');
```



In [12]:

```
import seaborn as sns
plt.figure(figsize=(20,10))
cor=a.corr()
sns.heatmap(cor,annot=True)
```

Out[12]: <AxesSubplot:>



In [13]:
```
y=c['year']
y
```

Out[13]:
```
0        1962
1        1962
2        1962
3        1962
4        1962
         ...
15584    2023
15585    2023
15586    2023
15587    2023
15588    2023
Name: year, Length: 15589, dtype: int64
```

In [14]:
```
x=c.drop(['year'],axis=1)
x
```

Out[14]:

	Open	High	Low	Close	Adj Close	Volume
0	0.263021	0.270182	0.263021	0.263021	0.048145	806400
1	0.259115	0.259115	0.253255	0.257161	0.047072	1574400
2	0.257813	0.261068	0.257813	0.259115	0.047430	844800
3	0.259115	0.262370	0.252604	0.253255	0.046357	1420800
4	0.251302	0.251302	0.245768	0.250651	0.045881	2035200
...	...	...	...	...	...	...
15584	58.580002	58.669998	58.099998	58.230000	57.770000	11263600
15585	57.959999	58.459999	57.599998	58.439999	58.439999	22727500
15586	58.270000	58.689999	58.240002	58.639999	58.639999	15369600
15587	58.590000	58.959999	58.439999	58.570000	58.570000	14942200
15588	58.549999	58.830002	58.419998	58.660000	58.660000	11880000

15589 rows × 6 columns

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

In [16]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[16]:
```
LinearRegression()
```

In [17]:
```
ypred=reg.predict(x_test)
ypred
```

Out[17]:
```
array([1979.64415143, 2016.95580904, 2023.79100424, ..., 1974.40547042,
       2014.20756587, 1979.45175131])
```

In [18]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[18]:
```
0.8992109729591107
```

In [19]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(ypred,y_test)
```

Out[19]:
```
31.67036075083715
```

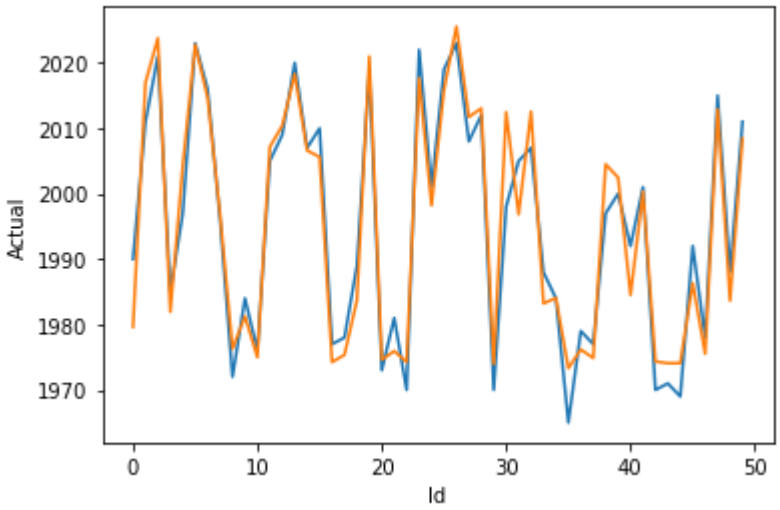
In [20]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[20]:

	index	Actual	Predicted	Id
0	7217	1990	1979.644151	0
1	12508	2011	2016.955809	1
2	14997	2021	2023.791004	2
3	5852	1985	1981.934693	3
4	8857	1997	2004.232780	4
5	15521	2023	2022.794253	5
6	13765	2016	2014.644263	6
7	8731	1996	1996.819042	7
8	2626	1972	1976.295565	8
9	5605	1984	1981.244481	9

In [21]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[21]:
```
[]
```



In [22]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
ridge=Ridge()
alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
parameters={'alpha':alpha}
r_reg=GridSearchCV(ridge,parameters)
r_reg.fit(x_train,y_train)
```

Out[22]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [23]:
```
r_reg.best_params_
```

Out[23]:
```
{'alpha': 10}
```

In [24]:
```
ridge=Ridge(10)
ridge.fit(x_train,y_train)
pred_ridge=ridge.predict(x_test)
pred_ridge
```

Out[24]:
```
array([1979.65227408, 2016.92329661, 2023.71341518, ..., 1974.41510976,
       2014.18636051, 1979.44272611])
```

In [25]:
```
r2_score(y_test,pred_ridge)
```

Out[25]:
```
0.8992316436703145
```

In [26]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=pred_ridge
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[26]:

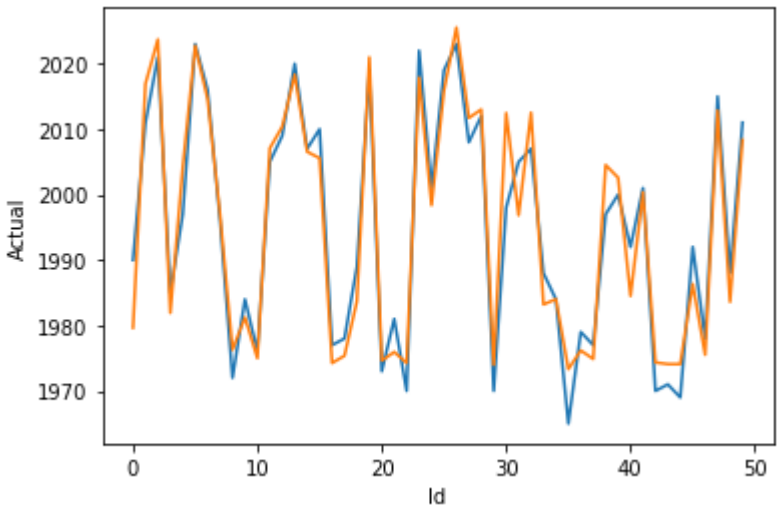
	index	Actual	Predicted	Id
0	7217	1990	1979.652274	0
1	12508	2011	2016.923297	1
2	14997	2021	2023.713415	2
3	5852	1985	1981.917152	3
4	8857	1997	2004.184202	4
5	15521	2023	2022.747537	5
6	13765	2016	2014.570360	6
7	8731	1996	1996.900512	7
8	2626	1972	1976.296453	8
9	5605	1984	1981.228557	9

In [27]:

```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[27]:

```
[]
```



In [28]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
lasso=Lasso()
alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
parameters={'alpha':alpha}
l_reg=GridSearchCV(lasso,parameters)
l_reg.fit(x_train,y_train)
```

Out[28]:

```
GridSearchCV(estimator=Lasso(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [29]:

```
l_reg.best_params_
```

Out[29]:

```
{'alpha': 1e-15}
```

In [30]:
```
lasso=Lasso(1e-15)
lasso.fit(x_train,y_train)
pred_lasso=lasso.predict(x_test)
pred_lasso
```

Out[30]:
```
array([1979.86271896, 2016.0406195 , 2021.72567914, ..., 1974.6820599 ,
       2013.86873454, 1979.19342394])
```

In [31]:
```
r2_score(y_test,pred_lasso)
```

Out[31]:
```
0.8931230923141643
```

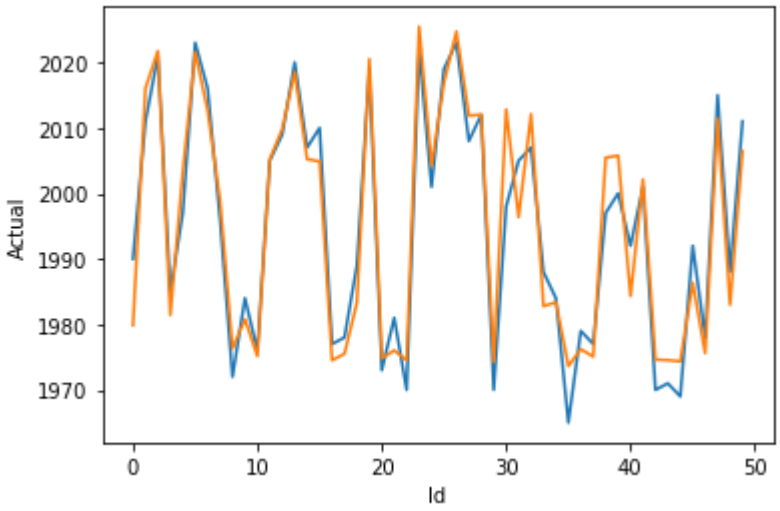
In [32]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=pred_lasso
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[32]:

	index	Actual	Predicted	Id
0	7217	1990	1979.862719	0
1	12508	2011	2016.040620	1
2	14997	2021	2021.725679	2
3	5852	1985	1981.447613	3
4	8857	1997	2003.196639	4
5	15521	2023	2021.577167	5
6	13765	2016	2012.673407	6
7	8731	1996	1998.927418	7
8	2626	1972	1976.327968	8
9	5605	1984	1980.784171	9

In [33]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

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



In [34]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet
elastic=ElasticNet()
alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
parameters={'alpha':alpha}
e_reg=GridSearchCV(elastic,parameters)
e_reg.fit(x_train,y_train)
```

Out[34]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [35]:
```
e_reg.best_params_
```

Out[35]:
```
{'alpha': 1e-15}
```

In [36]:
```
elastic=ElasticNet(1e-15)
elastic.fit(x_train,y_train)
pred_elastic=elastic.predict(x_test)
pred_elastic
```

Out[36]:
```
array([1979.86271896, 2016.0406195 , 2021.72567914, ..., 1974.6820599 ,
       2013.86873454, 1979.19342394])
```

In [37]:
```
r2_score(y_test,pred_lasso)
```

Out[37]:
```
0.8931230923141643
```

In [38]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=pred_elastic
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

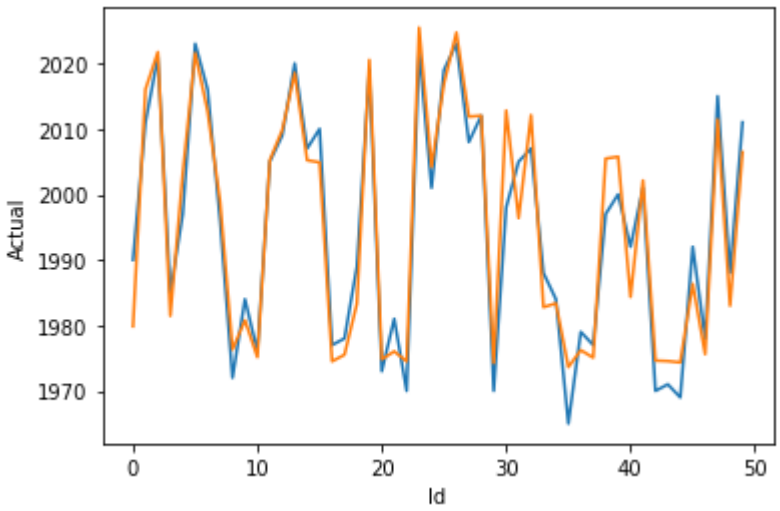
Out[38]:

	index	Actual	Predicted	Id
0	7217	1990	1979.862719	0
1	12508	2011	2016.040620	1
2	14997	2021	2021.725679	2

	index	Actual	Predicted	Id
3	5852	1985	1981.447613	3
4	8857	1997	2003.196639	4
5	15521	2023	2021.577167	5
6	13765	2016	2012.673407	6
7	8731	1996	1998.927418	7
8	2626	1972	1976.327968	8
9	5605	1984	1980.784171	9

In [39]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[39]:
```
[]
```



In [40]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['mse']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[40]:
```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse'], 'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [41]:
```
rfc_reg.best_params_
```

Out[41]:
```
{'criterion': 'mse', 'max_depth': 10, 'n_estimators': 125}
```

In [42]:
```
reg=RandomForestRegressor(n_estimators=125,criterion='mse',max_depth=10)
reg.fit(x_train,y_train)
ypred=reg.predict(x_test)
ypred
```

Out[42]:
```
array([1990.00371208, 2011.4812819 , 2020.45082414, ..., 1972.27549546,
       2012.65204876, 1984.18400752])
```

In [43]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[43]:
```
0.9984859942899521
```

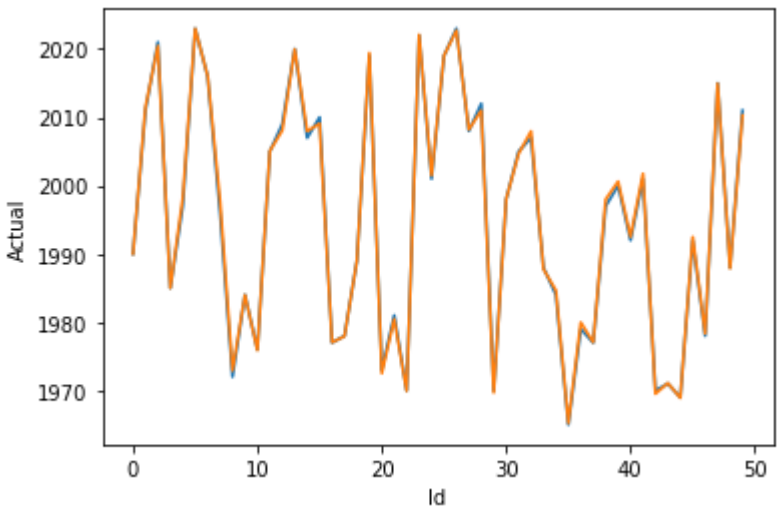
In [44]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[44]:

	index	Actual	Predicted	Id
0	7217	1990	1990.003712	0
1	12508	2011	2011.481282	1
2	14997	2021	2020.450824	2
3	5852	1985	1985.000000	3
4	8857	1997	1998.157757	4
5	15521	2023	2022.983769	5
6	13765	2016	2016.070955	6
7	8731	1996	1997.716235	7
8	2626	1972	1972.923865	8
9	5605	1984	1983.986132	9

In [45]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[45]:
```
[]
```



In [ ]: