Advance Monthly Sales for Retail and Food Services Units: Millions of Dollars. https://fred.stlouisfed.org/series/RSCCASN. Prediction and forecasting using Recurrent Neural Network(RNN).Tensorflow and keras

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv(r'C:\Users\chumj\Downloads\RSCCASN.csv',index_col='DATE',parse_dates=True)
df.columns=['Sales']
```

In [3]:

```
df.head(3)
```

Out[3]:

	Sales
DATE	
1992-01-01	6938
1992-02-01	7524
1992-03-01	8475

In [4]:

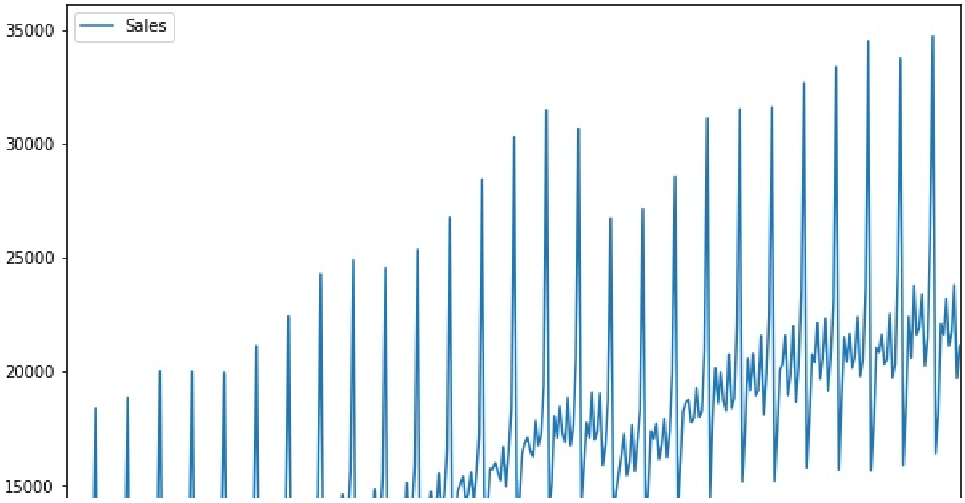
```
df.tail(3)
```

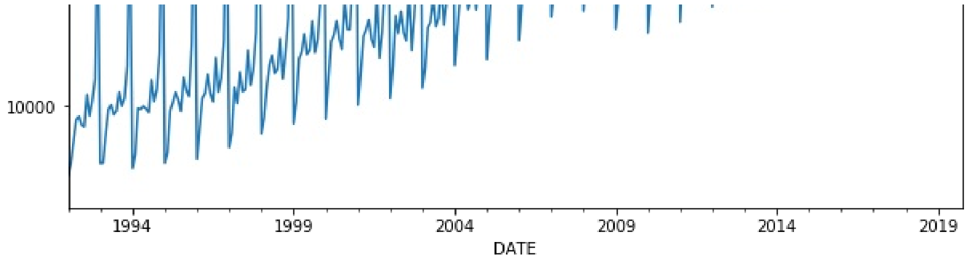
Out[4]:

	Sales
DATE	
2019-08-01	23791
2019-09-01	19695
2019-10-01	21113

In [5]:

```
df.plot(figsize=(10,8));
```





In [6]:

```
len(df)
```

Out[6]:

```
334
```

In [7]:

```
#lets take 2years or 24months for our test set
test_size=24
test_index=len(df)-test_size
test_index
```

Out[7]:

```
310
```

In [8]:

```
#splitting our data to train and test dataset
train=df.iloc[:310]
test=df.iloc[310:]
```

In [9]:

```
#Scaling and transforming our dataset.
from sklearn.preprocessing import MinMaxScaler
```

In [10]:

```
scaler=MinMaxScaler()
```

In [11]:

```
scaler_train=scaler.fit_transform(train)
scaler_test=scaler.transform(test)
```

In [12]:

```
# fit our data into Timeseries Generator
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
```

In [13]:

```
length=12
batch_size=1
generator=TimeseriesGenerator(scaler_train,scaler_train,length=length,batch_size=batch_size)
```

In [14]:

```
#lets take a look at what the first batch and 5 batches look like

X,y=generator[0]
```

In [15]:

```
X
```

Out[15]:

```
array([[[0.        ],
        [0.02127505],
        [0.05580163],
        [0.08942056],
        [0.09512053],
        [0.08146965],
        [0.07860151],
        [0.12979233],
        [0.09566512],
        [0.1203892 ],
        [0.15426227],
        [0.41595266]]])
```

In [16]:

```
len(X[0])
```

Out[16]:

```
12
```

In [ ]:

```

```

In [17]:

```
X
```

Out[17]:

```
array([[[0.        ],
        [0.02127505],
        [0.05580163],
        [0.08942056],
        [0.09512053],
        [0.08146965],
        [0.07860151],
        [0.12979233],
        [0.09566512],
        [0.1203892 ],
        [0.15426227],
        [0.41595266]]])
```

In [18]:

```
y
```

Out[18]:

```
array([[0.02047633]])
```

In [19]:

```
scaler_train
```

Out[19]:

```
array([[0.        ],
       [0.02127505],
       [0.05580163],
       [0.08942056],
       [0.09512053],
       [0.08146965],
       [0.07860151],
       [0.07666131],
       [0.12979233],
       [0.09566512],
       [0.1203892 ],
       [0.15426227],
       [0.41595266],
       [0.02047633],
       [0.02127505],
       [0.06636654],
       [0.10633895],
       [0.11345484],
       [0.09791606],
       [0.10368864],
       [0.13396747],
       [0.11163956],
       [0.12372931],
       [0.17506535],
       [0.43250799],
       [0.0124165 ],
       [0.03499855],
       [0.10829945],
       [0.10641156],
       [0.11149434],
       [0.10728289],
       [0.10154662],
       [0.1531731 ],
       [0.11904589],
       [0.13767064],
       [0.19663085],
       [0.47480395],
       [0.02105722],
       [0.03714058],
       [0.10445106],
       [0.11726692],
       [0.13367703],
       [0.12209556],
       [0.10332559],
       [0.1570578 ],
       [0.13694453],
       [0.12710572],
       [0.21104415],
       [0.47429567],
       [0.02708394],
       [0.07453529],
       [0.12383822],
       [0.13186175],
       [0.16217688],
       [0.13128086],
       [0.11850131],
       [0.18864362],
       [0.13378594],
       [0.15654952],
       [0.21554604],
       [0.47218995],
       [0.04559977],
       [0.06887162],
       [0.14166425],
       [0.11592361],
       [0.16580744],
       [0.13491141],
       [0.13781586],
       [0.20051554],
       [0.14536741],
       [0.17052716],
       [0.22894278],
       [0.51481266],
       [0.06760093],
       [0.09297851],
       [0.14249927],
       [0.17742521],
       [0.19245571],
       [0.16384694],
       [0.17045455],
       [0.21903137],
       [0.15524252],
       [0.19772001],
       [0.25108917],
       [0.56200988]
       [0.56200988],
       [0.08346645],
       [0.11737584],
       [0.18671943],
       [0.19779262],
       [0.22614726],
       [0.19394423],
       [0.20051554],
       [0.2467325 ],
       [0.19673976],
       [0.21819634],
       [0.27973424],
       [0.62982864],
       [0.09109062],
       [0.15364508],
       [0.21431165],
       [0.22458612],
       [0.24676881],
       [0.21837787],
       [0.20193146],
       [0.27853616],
       [0.23369881],
       [0.23348098],
       [0.31720157],
       [0.65121261],
       [0.11334592],
       [0.16381063],
       [0.2225167 ],
       [0.23322684],
       [0.24865669],
       [0.21772437],
       [0.20556201],
       [0.28608771],
       [0.18755446],
       [0.22799884],
       [0.30547488],
       [0.63883241],
       [0.12365669],
       [0.17397618],
       [0.25555475],
       [0.22716381],
       [0.26183561],
       [0.22770839],
       [0.21547342],
       [0.29679785],
       [0.19993465],
       [0.25660761],
       [0.32576968],
       [0.66845774],
       [0.14035725],
       [0.17150741],
       [0.23649434],
       [0.24517136],
       [0.28289283],
       [0.23809178],
       [0.25108917],
       [0.31164682],
       [0.24066947],
       [0.27820941],
       [0.34915045],
       [0.71964856],
       [0.17615452],
       [0.23257334],
       [0.28398199],
       [0.29480105],
       [0.30645513],
       [0.26499419],
       [0.28060558],
       [0.31404299],
       [0.26452222],
       [0.31338949],
       [0.37365669],
       [0.77940749],
       [0.18544874],
       [0.2519605 ],
       [0.31908946],
       [0.31839965],
       [0.32860151]
       [0.32860151],
       [0.31299012],
       [0.30060993],
       [0.35430584],
       [0.2912431 ],
       [0.3474804 ],
       [0.4162068 ],
       [0.84729887],
       [0.21619954],
       [0.27359861],
       [0.34301481],
       [0.35978798],
       [0.36904589],
       [0.34733517],
       [0.3389849 ],
       [0.39551263],
       [0.35684723],
       [0.37449172],
       [0.45211298],
       [0.89035725],
       [0.25373947],
       [0.29549085],
       [0.40324572],
       [0.36879175],
       [0.4188208 ],
       [0.37579872],
       [0.36105867],
       [0.43247168],
       [0.35677462],
       [0.38759802],
       [0.49419111],
       [0.86033256],
       [0.26267064],
       [0.32449898],
       [0.39326169],
       [0.36904589],
       [0.44096718],
       [0.36556056],
       [0.37870317],
       [0.43940604],
       [0.32471682],
       [0.35790009],
       [0.42905896],
       [0.71841417],
       [0.23413448],
       [0.28122277],
       [0.31313535],
       [0.34123584],
       [0.37489108],
       [0.30845193],
       [0.32925501],
       [0.38912286],
       [0.31556782],
       [0.36723061],
       [0.41366541],
       [0.733009  ],
       [0.22792623],
       [0.2860151 ],
       [0.37946558],
       [0.36683125],
       [0.39170055],
       [0.33401104],
       [0.35957014],
       [0.39881644],
       [0.33742376],
       [0.3802643 ],
       [0.47433198],
       [0.78445396],
       [0.24560703],
       [0.32649579],
       [0.41076096],
       [0.42510166],
       [0.42960354],
       [0.39373366],
       [0.39932472],
       [0.44746587],
       [0.40157566],
       [0.41086988],
       [0.41086988],
       [0.50896747],
       [0.87750508],
       [0.26938716],
       [0.39910688],
       [0.48010456],
       [0.4234316 ],
       [0.4724804 ],
       [0.42764304],
       [0.41126924],
       [0.50145222],
       [0.41580744],
       [0.43229015],
       [0.54741505],
       [0.89188208],
       [0.29832268],
       [0.37917514],
       [0.49509875],
       [0.44430729],
       [0.50254139],
       [0.43570288],
       [0.44430729],
       [0.5307871 ],
       [0.40564188],
       [0.46895876],
       [0.56269968],
       [0.89456869],
       [0.29919402],
       [0.38371333],
       [0.47676445],
       [0.48591345],
       [0.53115016],
       [0.43599332],
       [0.46797851],
       [0.54658002],
       [0.42506535],
       [0.48221028],
       [0.60096573],
       [0.93305257],
       [0.32043276],
       [0.40088586],
       [0.50152483],
       [0.48834592],
       [0.55209846],
       [0.46213331],
       [0.49469939],
       [0.5582341 ],
       [0.44339965],
       [0.4973497 ],
       [0.5864072 ],
       [0.95872059],
       [0.31789137],
       [0.43570288],
       [0.52839094],
       [0.48976184],
       [0.53434505],
       [0.48003195],
       [0.49843886],
       [0.56092071],
       [0.4665989 ],
       [0.49237584],
       [0.60652048],
       [1.        ],
       [0.3167659 ],
       [0.39235405],
       [0.51154516],
       [0.50515539],
       [0.53252977],
       [0.48591345],
       [0.49364653],
       [0.56629393],
       [0.46405751],
       [0.48536886]])
```

In [20]:

```
#creating the model
```

```
#creating the model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM
#Number of features, just one
n_features=1
```

In [21]:

```
model=Sequential()
```

In [22]:

```
model.add(LSTM(100,activation='relu',input_shape=(length,n_features)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')
model.summary()
```

```
WARNING:tensorflow:From C:\Users\chumj\Anaconda3\Ben\lib\site-
packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm (LSTM)                  (None, 100)               40800     
_________________________________________________________________
dense (Dense)                (None, 1)                 101       
=================================================================
Total params: 40,901
Trainable params: 40,901
Non-trainable params: 0
_________________________________________________________________
```

In [23]:

```
# implementing early stopping
from tensorflow.keras.callbacks import EarlyStopping
```

In [24]:

```
early_stop=EarlyStopping(monitor='val_loss',patience=2)
validation_generator=TimeseriesGenerator(scaler_test,scaler_test,length=length,batch_size=1)
```

In [25]:

```
model.fit_generator(generator,epochs=15,validation_data=validation_generator,callbacks=[early_stop
])
```

```
Epoch 1/15
WARNING:tensorflow:From C:\Users\chumj\Anaconda3\Ben\lib\site-
packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
298/298 [==============================] - 8s 26ms/step - loss: 0.0248 - val_loss: 0.0328
Epoch 2/15
298/298 [==============================] - 5s 17ms/step - loss: 0.0188 - val_loss: 0.0223
Epoch 3/15
298/298 [==============================] - 7s 23ms/step - loss: 0.0154 - val_loss: 0.0194
Epoch 4/15
298/298 [==============================] - 7s 23ms/step - loss: 0.0094 - val_loss: 0.0061
Epoch 5/15
298/298 [==============================] - 6s 21ms/step - loss: 0.0040 - val_loss: 8.9834e-04
Epoch 6/15
298/298 [==============================] - 5s 17ms/step - loss: 0.0027 - val_loss: 0.0015
Epoch 7/15
298/298 [==============================] - 6s 20ms/step - loss: 0.0024 - val_loss: 7.7742e-04
Epoch 8/15
298/298 [==============================] - 5s 17ms/step - loss: 0.0021 - val_loss: 0.0023
```

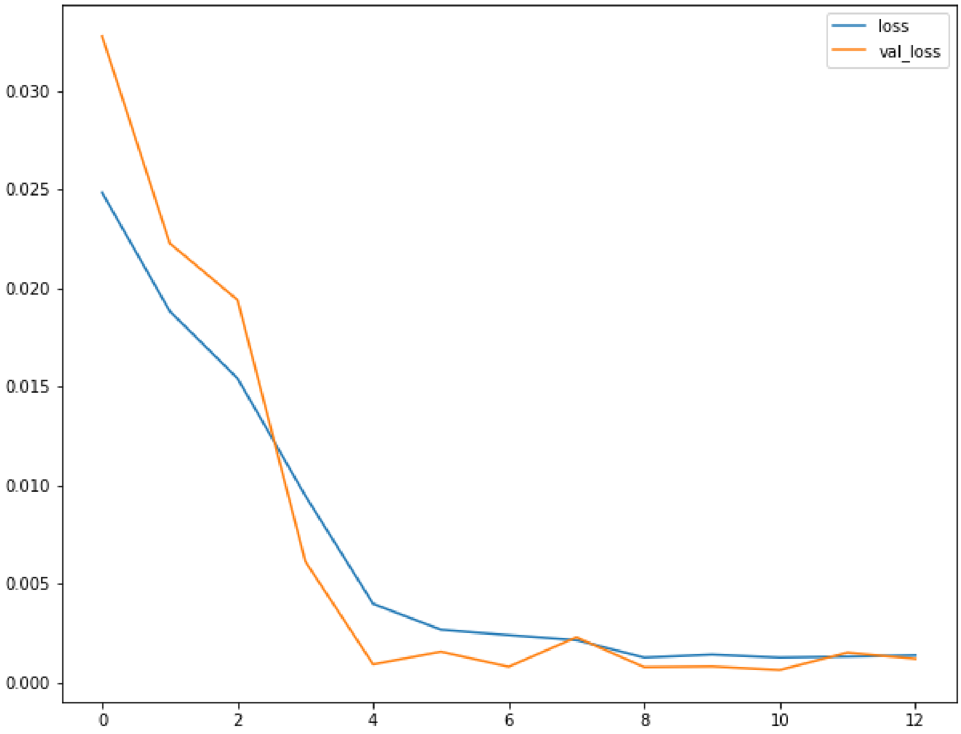
```
Epoch 9/15
298/298 [==============================] - 7s 22ms/step - loss: 0.0012 - val_loss: 7.5629e-04 0s -
loss
Epoch 10/15
298/298 [==============================] - 5s 18ms/step - loss: 0.0014 - val_loss: 7.8937e-04
Epoch 11/15
298/298 [==============================] - 5s 18ms/step - loss: 0.0012 - val_loss: 6.1079e-04
Epoch 12/15
298/298 [==============================] - 5s 18ms/step - loss: 0.0013 - val_loss: 0.0015
Epoch 13/15
298/298 [==============================] - 5s 18ms/step - loss: 0.0014 - val_loss: 0.0012
```

Out[25]:

```
<tensorflow.python.keras.callbacks.History at 0x28a369ad708>
```

In [48]:

```
#visualise the losses
loss=pd.DataFrame(model.history.history)
loss.plot(figsize=(10,8));
```



In [56]:

```
import numpy as np
```

In [57]:

```
test_predictions = []

first_eval_batch = scaler_train[-length:]
current_batch = first_eval_batch.reshape((1, length, n_features))

for i in range(len(test)):

    # get prediction 1 time stamp ahead ([0] is for grabbing just the number instead of [array])
    current_pred = model.predict(current_batch)[0]

    # store prediction
    test_predictions.append(current_pred)

    # update batch to now include prediction and drop first value
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

In [58]:

```
true_predictions=scaler.inverse_transform(test_predictions)
```

In [59]:

```
test['predictions']=true_predictions
```

In [60]:

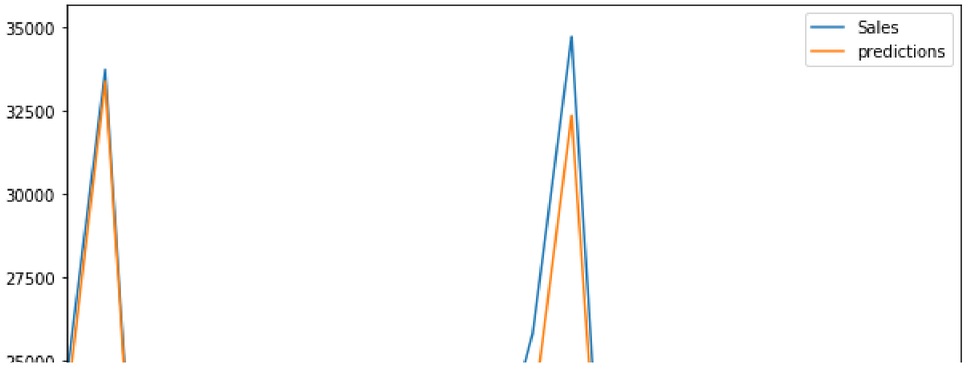
```
test
```

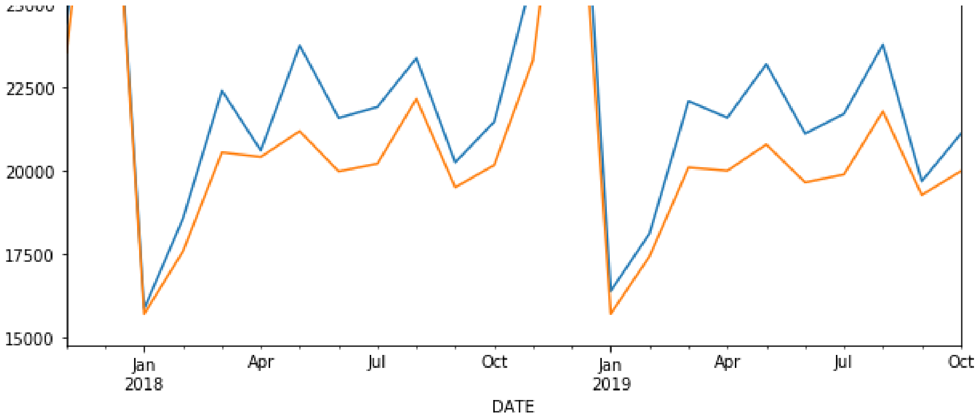
Out[60]:

	Sales	predictions
DATE		
2017-11-01	24438	23555.788689
2017-12-01	33720	33377.917974
2018-01-01	15881	15722.589042
2018-02-01	18585	17600.460015
2018-03-01	22404	20560.135283
2018-04-01	20616	20423.109054
2018-05-01	23764	21196.342269
2018-06-01	21589	19990.039942
2018-07-01	21919	20222.103814
2018-08-01	23381	22162.364584
2018-09-01	20260	19511.344946
2018-10-01	21473	20177.742079
2018-11-01	25831	23323.929214
2018-12-01	34706	32344.163607
2019-01-01	16410	15728.320392
2019-02-01	18134	17449.799049
2019-03-01	22093	20109.957491
2019-04-01	21597	20011.268595
2019-05-01	23200	20796.300248
2019-06-01	21123	19659.816638
2019-07-01	21714	19899.947250
2019-08-01	23791	21789.923670
2019-09-01	19695	19281.816757
2019-10-01	21113	19996.963203

In [61]:

```
test.plot(figsize=(10,8));
```





In [62]:

```
from sklearn import metrics
```

In [77]:

```
print('MSE:',metrics.mean_squared_error(test['Sales'],test['predictions']))
print('\n')

print('RMSE:',np.sqrt(metrics.mean_squared_error(test['Sales'],test['predictions'])))
```

```
MSE: 2373022.710653571


RMSE: 1540.4618497884233
```

In [78]:

```
test['Sales'].mean()
```

Out[78]:

```
22393.208333333332
```

In [79]:

```
#After satisfied with our model is time to do some forecast.
All_scaler=MinMaxScaler()
```

In [80]:

```
Ful_scaled=All_scaler.fit_transform(df)
```

In [81]:

```
length=12
generator=TimeseriesGenerator(Ful_scaled,Ful_scaled,length=length,batch_size=1)
```

In [83]:

```
model=Sequential()
model.add(LSTM(100,activation='relu',input_shape=(length,n_features)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')
model.fit_generator(generator,epochs=7)
```

```
Epoch 1/7
322/322 [==============================] - 8s 24ms/step - loss: 0.0231
Epoch 2/7
322/322 [==============================] - 5s 16ms/step - loss: 0.0184
Epoch 3/7
322/322 [==============================] - 7s 21ms/step - loss: 0.0144
```

```
Epoch 4/7
322/322 [==============================] - 5s 16ms/step - loss: 0.0084
Epoch 5/7
322/322 [==============================] - 5s 16ms/step - loss: 0.0038
Epoch 6/7
322/322 [==============================] - 7s 22ms/step - loss: 0.0030
Epoch 7/7
322/322 [==============================] - 6s 17ms/step - loss: 0.0018
```

Out[83]:

```
<tensorflow.python.keras.callbacks.History at 0x28a3c28e048>
```

In [88]:

```
forecast = []
periods=24 # use whatever forcast length you want
first_eval_batch =Ful_scaled [-length:]
current_batch = first_eval_batch.reshape((1, length, n_features))

for i in range(len(test)):

    # get prediction 1 time stamp ahead ([0] is for grabbing just the number instead of [array])
    current_pred = model.predict(current_batch)[0]

    # store prediction
    forecast.append(current_pred)

    # update batch to now include prediction and drop first value
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

In [89]:

```
forecast=scaler.inverse_transform(forecast)
```

In [90]:

```
df
```

Out[90]:

	Sales
DATE	
1992-01-01	6938
1992-02-01	7524
1992-03-01	8475
1992-04-01	9401
1992-05-01	9558
...	...
2019-06-01	21123
2019-07-01	21714
2019-08-01	23791
2019-09-01	19695
2019-10-01	21113

334 rows × 1 columns

In [91]:

```
forecast
```

Out[91]:

```
array([[27287.74988222],
       [35895.22197247],
       [17885.01721287],
       [19663.67310905],
       [23243.72971058],
       [22932.90861177],
       [24427.70587492],
       [22784.30722189],
       [23420.12593365],
       [25275.22172546],
       [21975.50220251],
       [23408.46950626],
       [29236.13813686],
       [37680.00488091],
       [19352.97678328],
       [21319.72801542],
       [24739.22143412],
       [24567.55509377],
       [26022.72542906],
       [24648.1896615 ],
       [25374.49836826],
       [27148.12065792],
       [24390.21158075],
       [25910.30657482]])
```

In [92]:

```
forecast_index=pd.date_range(start='2019-10-01',periods=periods,freq='MS')
```

In [93]:

```
forecast_index
```

Out[93]:

```
DatetimeIndex(['2019-10-01', '2019-11-01', '2019-12-01', '2020-01-01',
               '2020-02-01', '2020-03-01', '2020-04-01', '2020-05-01',
               '2020-06-01', '2020-07-01', '2020-08-01', '2020-09-01',
               '2020-10-01', '2020-11-01', '2020-12-01', '2021-01-01',
               '2021-02-01', '2021-03-01', '2021-04-01', '2021-05-01',
               '2021-06-01', '2021-07-01', '2021-08-01', '2021-09-01'],
              dtype='datetime64[ns]', freq='MS')
```

In [94]:

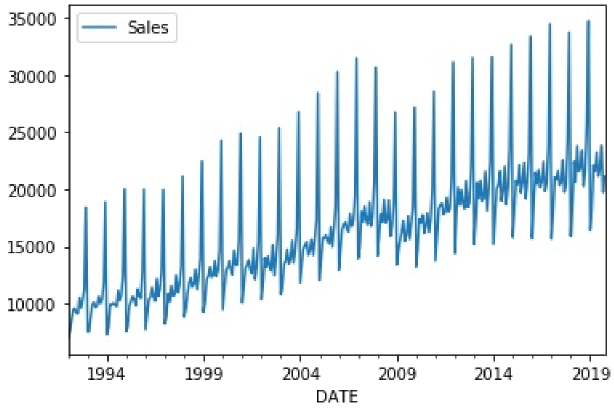
```
forecast_df=pd.DataFrame(data=forecast,index=forecast_index,columns=['forecast'])
```

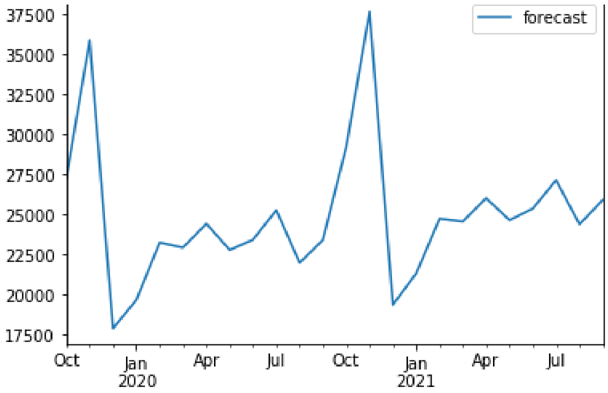
In [95]:

```
df.plot()
forecast_df.plot()
```

Out[95]:

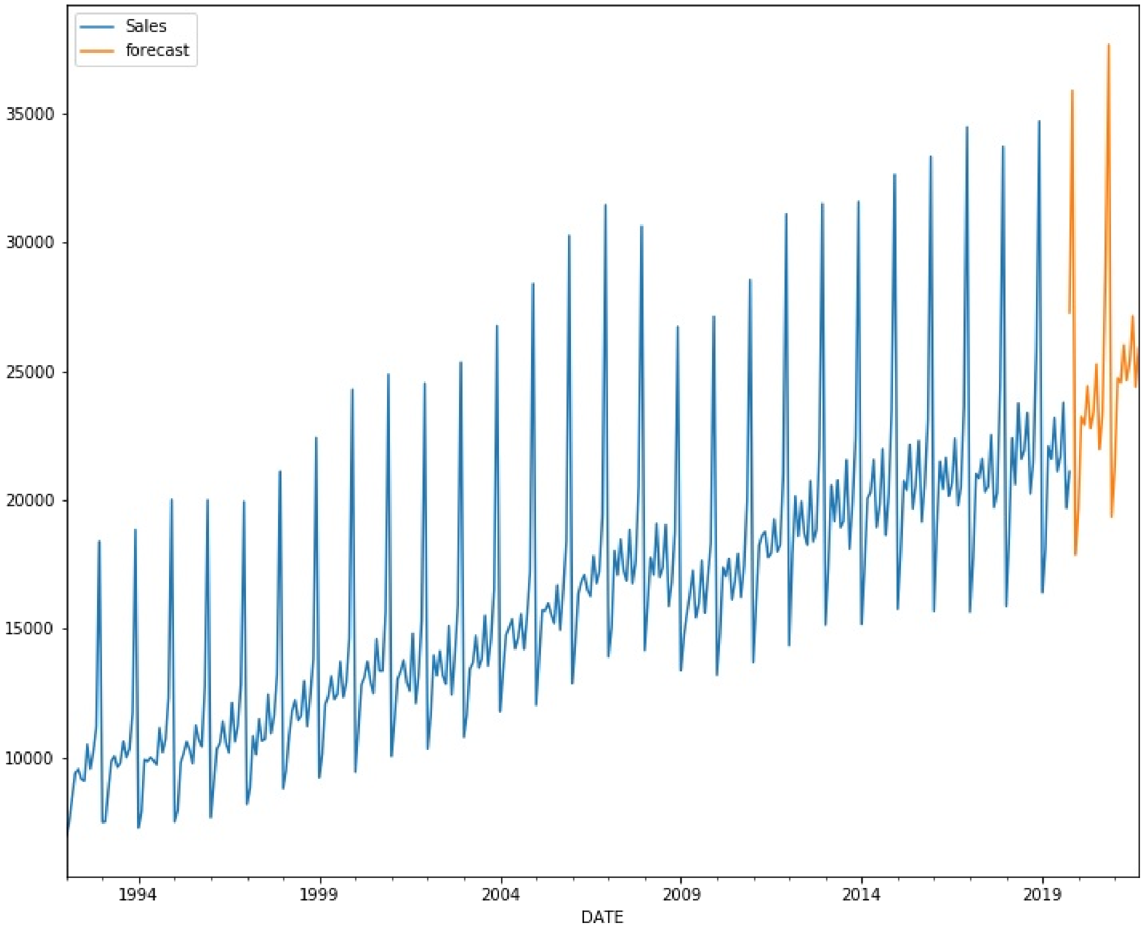
```
<matplotlib.axes._subplots.AxesSubplot at 0x28a3c563708>
```





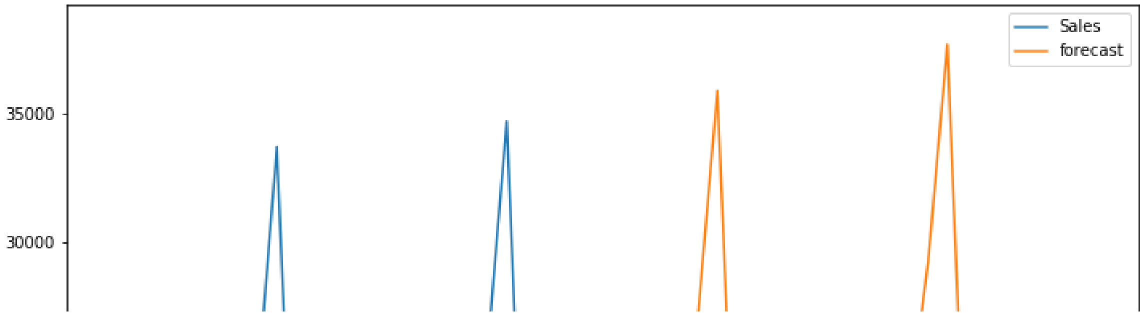
In [97]:

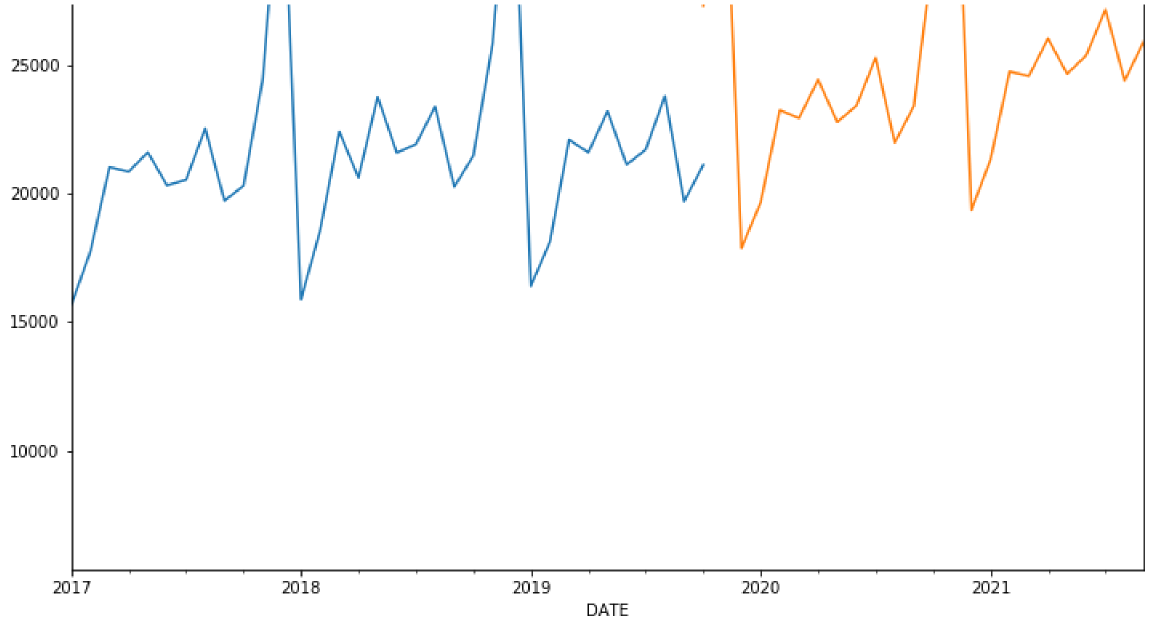
```
ax=df.plot(figsize=(12,10))
forecast_df.plot(ax=ax,figsize=(12,10));
```



In [99]:

```
ax=df.plot(figsize=(12,10))
forecast_df.plot(ax=ax,figsize=(12,10))
plt.xlim('2017-01-01','2021-09-01');
```





```
In [ ]:
```