Shauna Smith

Bellevue University -DSC630

Professor: Andrew Hua

Week 8 "Time Series Modeling"

Objective

Plot the data with proper labeling and make some observations on the graph.

Split this data into a training and test set. Use the last year of data (July 2020 – June 2021) of data as your test set and the rest as your training set.

Use the training set to build a predictive model for the monthly retail sales.

Use the model to predict the monthly retail sales on the last year of data.

Report the RMSE of the model predictions on the test set.

In [1]:
```
#importing the dataset
import pandas as pd
```

In [249…
```
df=pd.read_excel(r"C:\Users\Shaun\OneDrive\Documents\DSC630\us_retail_sales.xlsx")
```

In [149…
```
#Checking df
df.tail()
```

Out[149]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
25	2017	416081	415503	414620	416889	414540	416505	416744.0	417179.0	426501.0	426933.0	431158.0	433282.0
26	2018	432148	434106	433232	435610	439996	438191	440703.0	439278.0	438985.0	444038.0	445242.0	434803.0
27	2019	440751	439996	447167	448709	449552	450927	454012.0	456500.0	452849.0	455486.0	457658.0	458055.0
28	2020	460586	459610	434281	379892	444631	476343	481627.0	483716.0	493327.0	493991.0	488652.0	484782.0
29	2021	520162	504458	559871	562269	548987	550782	NaN	NaN	NaN	NaN	NaN	NaN

In [150…
```
#familiarize
df.shape
```

Out[150]:
```
(30, 13)
```

In [151…
```
df.describe()
```

Out[151]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	29.000000	
mean	2006.500000	304803.833333	305200.900000	307533.566667	306719.600000	309205.633333	311406.966667	304375.448276	3
std	8.803408	97687.399232	96682.043053	100002.422696	98207.161171	99541.010078	101057.212178	92471.103673	
min	1992.000000	146925.000000	147223.000000	146805.000000	148032.000000	149010.000000	149800.000000	150761.000000	1
25%	1999.250000	228856.750000	231470.750000	233019.000000	233235.500000	234976.500000	235967.250000	233948.000000	2
50%	2006.500000	303486.000000	304592.500000	308655.500000	311233.500000	308690.000000	312957.000000	313520.000000	3
75%	2013.750000	371527.000000	377008.500000	379221.000000	376797.500000	382698.250000	383839.750000	373554.000000	3
max	2021.000000	520162.000000	504458.000000	559871.000000	562269.000000	548987.000000	550782.000000	481627.000000	4

In [152…

```
#Plot the data with proper labeling and make some observations on the graph.
```

In [153…

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [154…

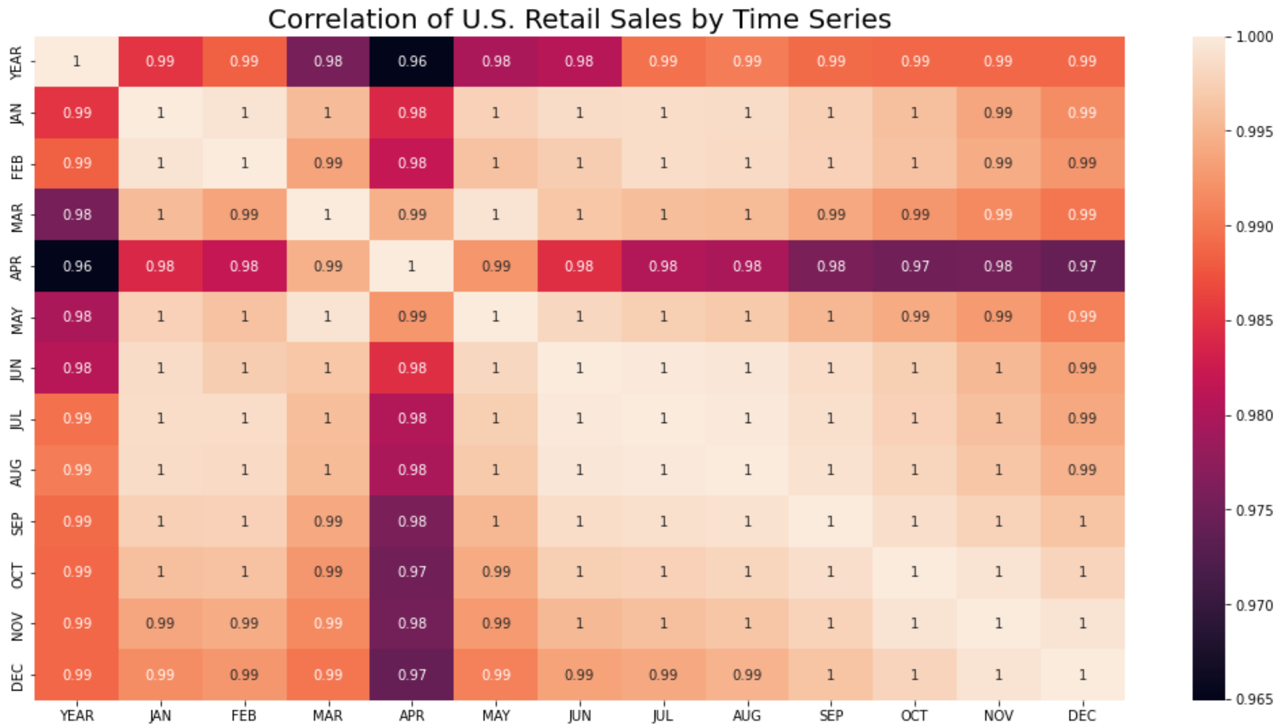
```
#correlation review
correl=df.corr()
```

In [39]:

```
#create heatmap
plt.figure(figsize=(18,9))
sns.heatmap(correl, annot=True)
plt.title("Correlation of U.S. Retail Sales by Time Series", fontsize=20)
```

Out[39]:

```
Text(0.5, 1.0, 'Correlation of U.S. Retail Sales by Time Series')
```



	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
YEAR	1	0.99	0.99	0.98	0.96	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99
JAN	0.99	1	1	1	0.98	1	1	1	1	1	1	0.99	0.99
FEB	0.99	1	1	0.99	0.98	1	1	1	1	1	1	0.99	0.99
MAR	0.98	1	0.99	1	0.99	1	1	1	1	0.99	0.99	0.99	0.99
APR	0.96	0.98	0.98	0.99	1	0.99	0.98	0.98	0.98	0.98	0.97	0.98	0.97
MAY	0.98	1	1	1	0.99	1	1	1	1	1	0.99	0.99	0.99
JUN	0.98	1	1	1	0.98	1	1	1	1	1	1	1	0.99
JUL	0.99	1	1	1	0.98	1	1	1	1	1	1	1	0.99
AUG	0.99	1	1	1	0.98	1	1	1	1	1	1	1	0.99
SEP	0.99	1	1	0.99	0.98	1	1	1	1	1	1	1	1
OCT	0.99	1	1	0.99	0.97	0.99	1	1	1	1	1	1	1
NOV	0.99	0.99	0.99	0.99	0.98	0.99	1	1	1	1	1	1	1
DEC	0.99	0.99	0.99	0.99	0.97	0.99	0.99	0.99	0.99	1	1	1	1

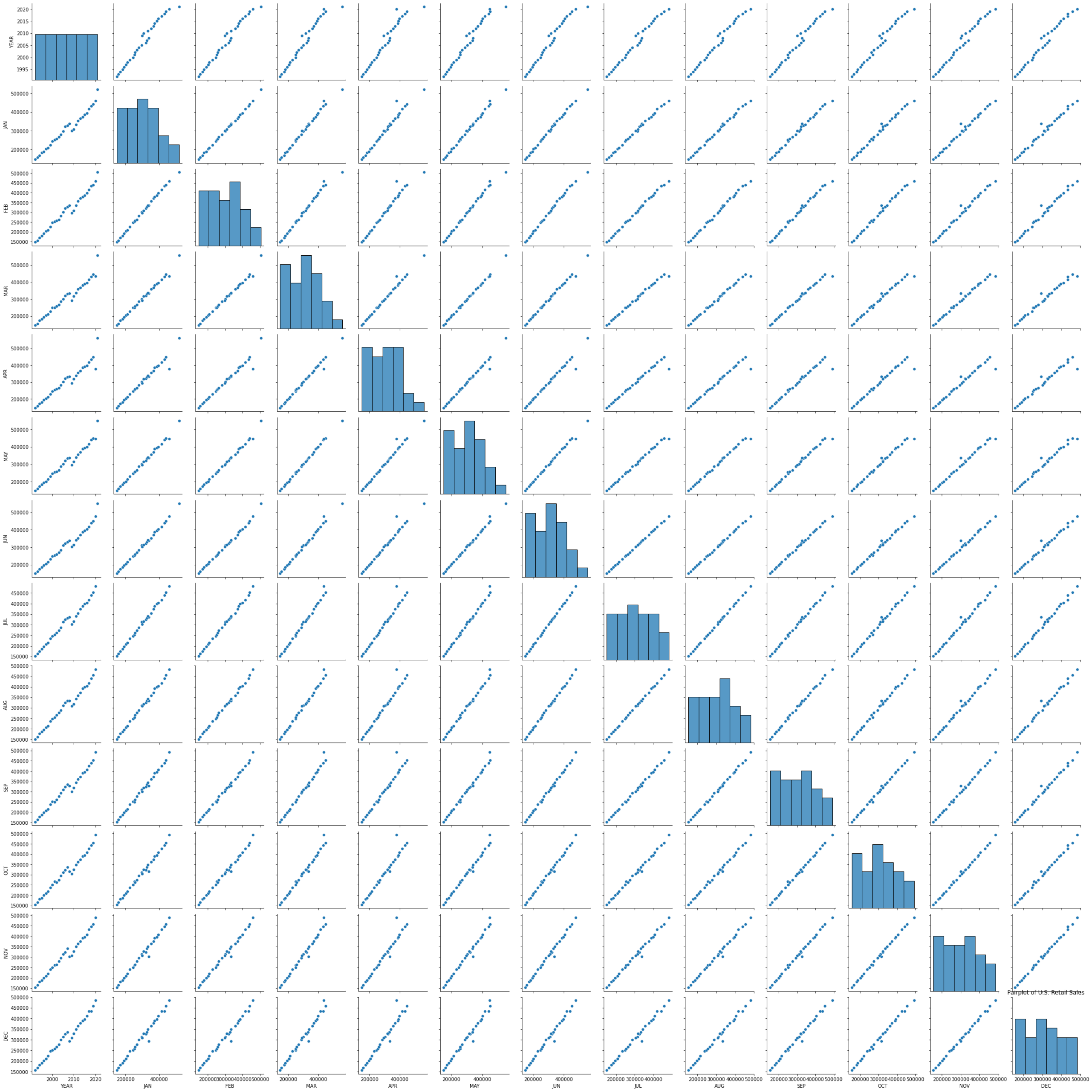
In [40]:

```
#create pairplot
plt.figure(figsize=(18,9))
sns.pairplot(df)
plt.title("Pairplot of U.S. Retail Sales")
```

Out[40]:

```
Text(0.5, 1.0, 'Pairplot of U.S. Retail Sales')

<Figure size 1296x648 with 0 Axes>
```

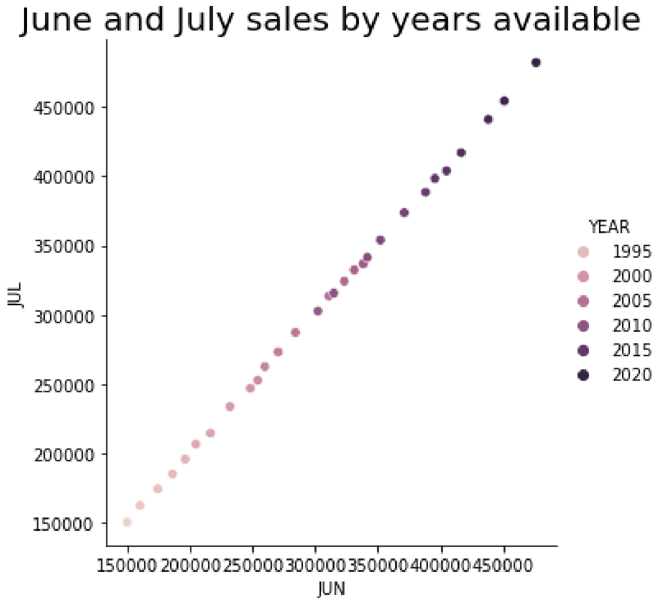


In [155…

```
#Review of available JUNE and JULY by YEAR
sns.relplot(x='JUN', y='JUL', hue='YEAR', data=df)
plt.title("June and July sales by years available", fontsize=20)
```

Out[155]:

```
Text(0.5, 1.0, 'June and July sales by years available')
```



In [50]:
```
#Begin the process to split the data for model.
import numpy as np
```

In [182…
```
df.tail(3)
```

Out[182]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
27	2019	440751	439996	447167	448709	449552	450927	454012.0	456500.0	452849.0	455486.0	457658.0	458055.0
28	2020	460586	459610	434281	379892	444631	476343	481627.0	483716.0	493327.0	493991.0	488652.0	484782.0
29	2021	520162	504458	559871	562269	548987	550782	NaN	NaN	NaN	NaN	NaN	NaN

In [158…
```
#seperate values of interest for proposed test set
vOI20=df.loc[28,['YEAR','JUL','AUG','SEP','OCT','NOV','DEC']]
vOI21=df.loc[29,['YEAR','JAN','FEB','MAR','APR','MAY','JUN']]
```

In [159…
```
#Checking values 2020
df20=pd.DataFrame(vOI20)
df20=df20.T
```

In [160…
```
#checking values 2021
df21=pd.DataFrame(vOI21)
df21=df21.T
```

In [166…
```
#combining into 1 set of complete values for a full picture of predictive test set range.
hmm=pd.merge(df20, df21, on = "YEAR", how = "outer")
hmm
```

Out[166]:

	YEAR	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN
0	2020.0	481627.0	483716.0	493327.0	493991.0	488652.0	484782.0	NaN	NaN	NaN	NaN	NaN	NaN
1	2021.0	NaN	NaN	NaN	NaN	NaN	NaN	520162.0	504458.0	559871.0	562269.0	548987.0	550782.0

In [250…
```
#configure the df values to a new 'Sales' column and index by 'Date'
df.set_index('YEAR', inplace=True)
```

In [251…
```
df.tail(3)
```

Out[251]:

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
YEAR												
2019	440751	439996	447167	448709	449552	450927	454012.0	456500.0	452849.0	455486.0	457658.0	458055.0
2020	460586	459610	434281	379892	444631	476343	481627.0	483716.0	493327.0	493991.0	488652.0	484782.0
2021	520162	504458	559871	562269	548987	550782	NaN	NaN	NaN	NaN	NaN	NaN

In [252…

```
# Prepping the dataframe for SARIMA
tink=df.stack
```

In [253…

```
tinker=pd.DataFrame({'Sales':df.stack()})
tinker
```

Out[253]:

		Sales
YEAR		
1992	**JAN**	146925.0
	FEB	147223.0
	MAR	146805.0
	APR	148032.0
	MAY	149010.0
...	**...**	...
2021	**FEB**	504458.0
	MAR	559871.0
	APR	562269.0
	MAY	548987.0
	JUN	550782.0

354 rows × 1 columns

In [254…

```
tinkere=tinker.reset_index()
```

In [255…

```
tinkere.rename(columns={'level_1':'Month'}, inplace=True)
```

In [256…

```
tinkere.head(3)
```

Out[256]:

	YEAR	Month	Sales
0	1992	JAN	146925.0
1	1992	FEB	147223.0
2	1992	MAR	146805.0

In [257…

```
#Converting to datetime
tinkere['Date'] = pd.to_datetime(tinkere.YEAR.astype(str) + '/' + tinkere.Month.astype(str) + '/01')
```

In [258…

```
tinkere.head(2)
```

Out[258]:

	YEAR	Month	Sales	Date
0	1992	JAN	146925.0	1992-01-01
1	1992	FEB	147223.0	1992-02-01

In [265…

```
tinkeri=tinkere.drop(['YEAR','Month'], axis=1)
tinkeri.tail()
```

Out[265]:

	Sales	Date
349	504458.0	2021-02-01
350	559871.0	2021-03-01
351	562269.0	2021-04-01
352	548987.0	2021-05-01
353	550782.0	2021-06-01

In [266…

```
tinkeri.set_index('Date', inplace=True)
```

In [271…

```
#Final working DF for SARIMA
tinkered=tinkeri
#Showing the test set value ranges for iloc
tinkered.tail(12)
```

Out[271]:

	Sales
Date	
2020-07-01	481627.0
2020-08-01	483716.0
2020-09-01	493327.0
2020-10-01	493991.0
2020-11-01	488652.0
2020-12-01	484782.0
2021-01-01	520162.0
2021-02-01	504458.0
2021-03-01	559871.0
2021-04-01	562269.0
2021-05-01	548987.0
2021-06-01	550782.0

In [272…

```
#Split this data into a training and test set.
#Use the last year of data (July 2020 - June 2021) as your test set and the rest as your training set
training=tinkered.iloc[:-12,:]
test=tinkered.iloc[-12:,:]
```

In [273…

```
training.shape,test.shape
```

Out[273]:

```
((342, 1), (12, 1))
```

In [274…

```
#Confirmed ranges
test
```

Out[274]:

	Sales
Date	
2020-07-01	481627.0
2020-08-01	483716.0
2020-09-01	493327.0
2020-10-01	493991.0
2020-11-01	488652.0
2020-12-01	484782.0
2021-01-01	520162.0
2021-02-01	504458.0
2021-03-01	559871.0
2021-04-01	562269.0
2021-05-01	548987.0
2021-06-01	550782.0

In [275…

```
#Use the training set to build a predictive model for the monthly retail sales.
!pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.3-cp39-cp39-win_amd64.whl (572 kB)
Requirement already satisfied: scipy>=1.3.2 in c:\users\shaun\anaconda3\lib\site-packages (from pmdarima)
(1.7.3)
Requirement already satisfied: statsmodels>=0.13.2 in c:\users\shaun\anaconda3\lib\site-packages (from pmdar
ima) (0.13.2)
Requirement already satisfied: joblib>=0.11 in c:\users\shaun\anaconda3\lib\site-packages (from pmdarima)
(1.1.0)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\shaun\anaconda3\lib\site-packages (fr
om pmdarima) (61.2.0)
Requirement already satisfied: urllib3 in c:\users\shaun\anaconda3\lib\site-packages (from pmdarima) (1.26.
9)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\shaun\anaconda3\lib\site-packages (from pmdari
ma) (1.0.2)
Requirement already satisfied: pandas>=0.19 in c:\users\shaun\anaconda3\lib\site-packages (from pmdarima)
(1.4.2)
Requirement already satisfied: numpy>=1.21.2 in c:\users\shaun\anaconda3\lib\site-packages (from pmdarima)
(1.21.5)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in c:\users\shaun\anaconda3\lib\site-package
s (from pmdarima) (0.29.28)
Requirement already satisfied: pytz>=2020.1 in c:\users\shaun\anaconda3\lib\site-packages (from pandas>=0.19
->pmdarima) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\shaun\anaconda3\lib\site-packages (from pa
ndas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\shaun\anaconda3\lib\site-packages (from python-dateutil>
=2.8.1->pandas>=0.19->pmdarima) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\shaun\anaconda3\lib\site-packages (from scik
it-learn>=0.22->pmdarima) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\shaun\anaconda3\lib\site-packages (from statsmodels>
=0.13.2->pmdarima) (0.5.2)
Requirement already satisfied: packaging>=21.3 in c:\users\shaun\anaconda3\lib\site-packages (from statsmode
ls>=0.13.2->pmdarima) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shaun\anaconda3\lib\site-packages (from
packaging>=21.3->statsmodels>=0.13.2->pmdarima) (3.0.4)
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.3
```

In [276…

```
from pmdarima import auto_arima
```

In [277…

```
#SARIMA Model
model=auto_arima(y=training.Sales, m=7)
```

In [280…

```
#Predictions
predictions=pd.Series(model.predict(n_periods=len(test)))
```

```
predictions.index=test.index
```

In [281…

```
predictions
```

Out[281]:

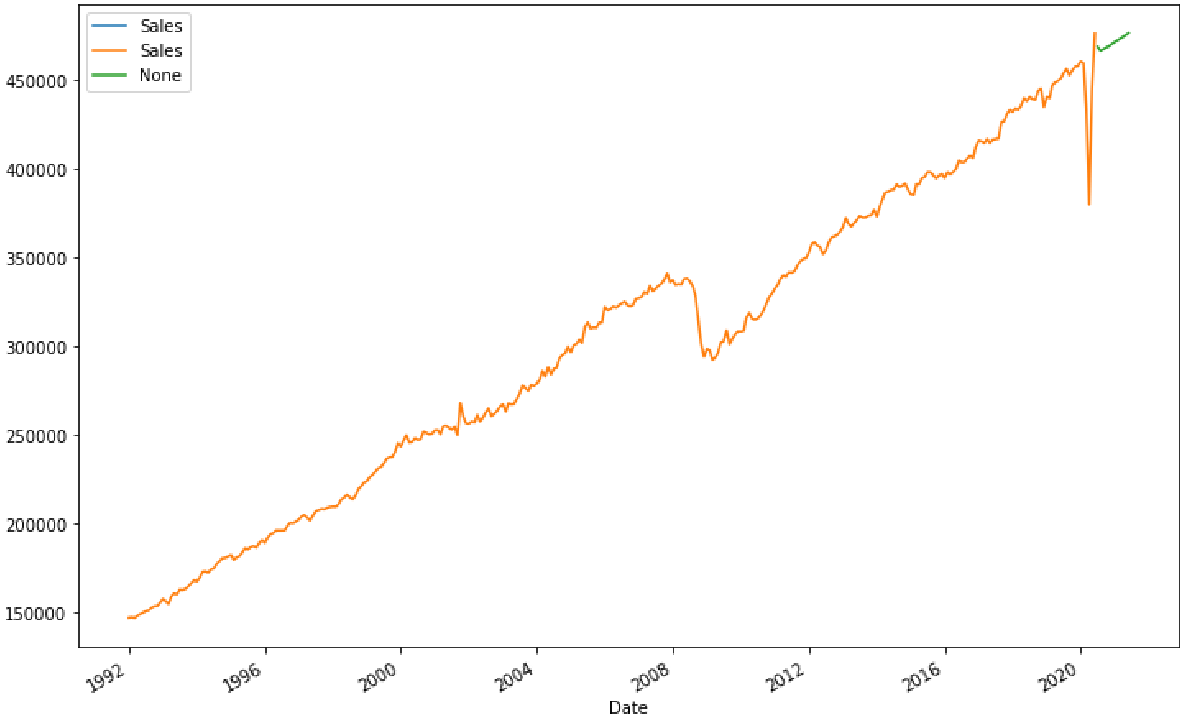
```
Date
2020-07-01    468850.619119
2020-08-01    466542.141878
2020-09-01    467507.165180
2020-10-01    468417.777849
2020-11-01    469234.406228
2020-12-01    470439.331724
2021-01-01    471531.868987
2021-02-01    472492.534026
2021-03-01    473468.272628
2021-04-01    474474.955092
2021-05-01    475514.137938
2021-06-01    476609.459177
dtype: float64
```

In [308…

```
#Visualize
training['Sales']['2020-07-01':].plot(figsize=(12,8),legend=True)
training['Sales'].plot(legend=True)
predictions.plot(legend=True)
```

Out[308]:

```
<AxesSubplot:xlabel='Date'>
```

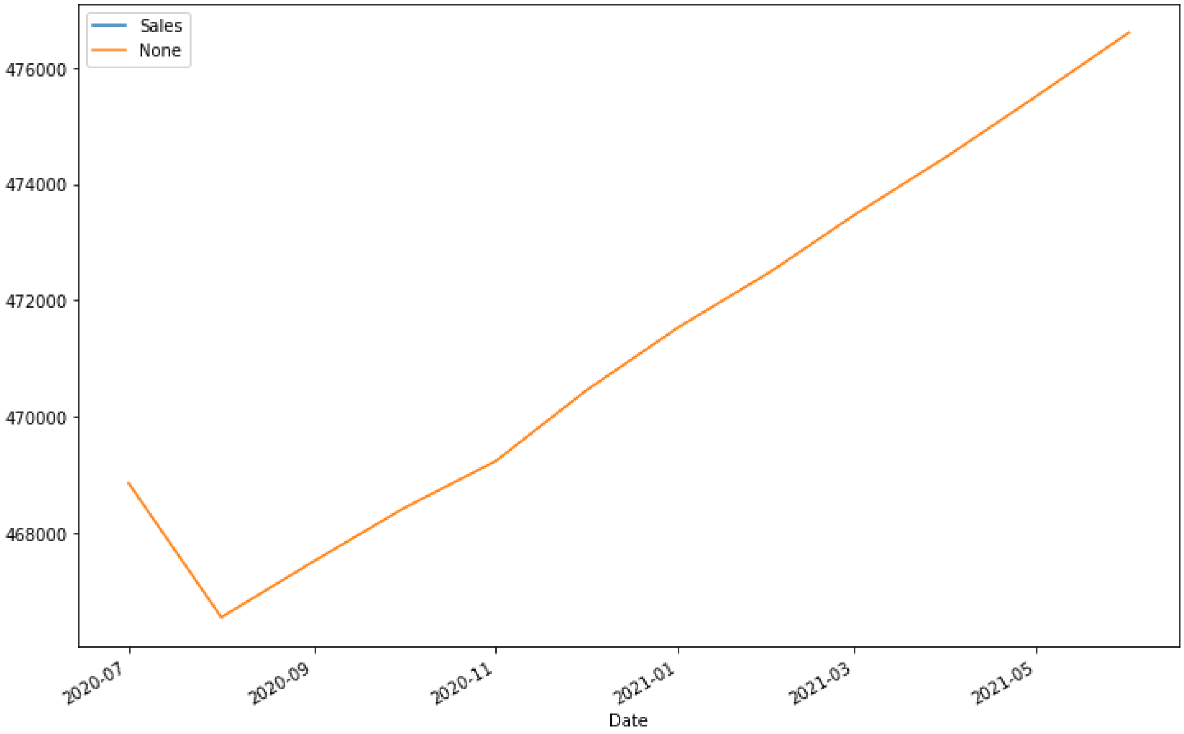


In [311…

```
#Closer Just Predictions View for 7/20'-6/21' "Retail Sales"
training['Sales']['2020-07-01':].plot(figsize=(12,8),legend=True)
predictions.plot(legend=True)
```

Out[311]:

```
<AxesSubplot:xlabel='Date'>
```

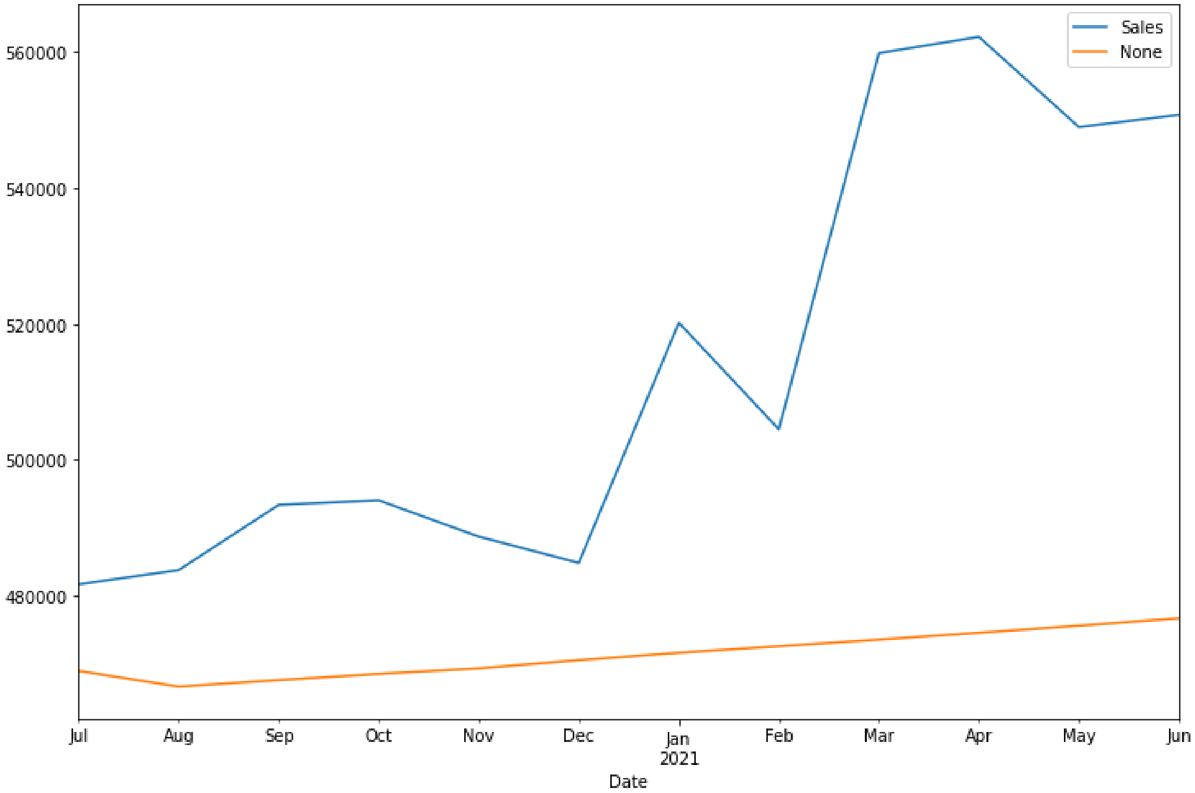


In [312…

```
#Actual test Sales with predictions in same time series together, showing predictions much lower than expecte
test.plot(figsize=(12,8),legend=True)
predictions.plot(legend=True)
```

Out[312]:

```
<AxesSubplot:xlabel='Date'>
```



In [302…

```
#Report the RMSE of the model predictions on the test set.
from sklearn.metrics import mean_squared_error
```

In [299…

```
rmse= np.sqrt(mean_squared_error(test['Sales'], predictions))

print('The RMSE for the model test is: ', rmse)
```

```
The RMSE for the model test is:  51495.403922120684
```

```
In [ ]:
```