In [13]:
```
#General Libraries for

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from matplotlib.pyplot import figure
from sklearn.metrics import mean_absolute_error
import statsmodels.tsa.statespace.sarimax

import warnings
warnings.filterwarnings('ignore')
warnings.warn('DelftStack')
warnings.warn('Do not show this message')
print("No Warning Shown")

# Libraries Specifically for Neural Nets:

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from keras.preprocessing.sequence import TimeseriesGenerator
from tensorflow.keras.models import load_model
from tensorflow import keras


```

```
No Warning Shown
```

In [3]:
```
acc_growers = [73177, 91259, 66015, 60706, 399593]
```

In [4]:

```
def sarimax_vs_lstm(RegionID, graph="Go", run=1):

    #Pulling Data

    test_df = pd.read_csv(f'{RegionID}.csv')

    test_df['time'] = pd.to_datetime(test_df['time'])

    test_df.set_index('time', inplace = True)

    test_df.columns = ['Drop', 'RegionName', 'RegionID', 'SizeRank', '
           'Metro', 'CountyName', 'value']

    test_df.drop('Drop', axis = 'columns', inplace = True)

    #Preparing X_train and y_train

    X_train = test_df[['value']].head(len(test_df) - 12)
    y_train = test_df[['value']].tail(12)

    #Scale Values

    scaler = MinMaxScaler()

    scaler.fit(X_train)

    X_train_scaled = scaler.transform(X_train)
    y_train_scaled = scaler.transform(y_train)

    #Time Series Genertor

    n_input = 12
    n_features = 1
    generator = TimeseriesGenerator(X_train_scaled, X_train_scaled, le

    #Building Actual Model

    model = Sequential()
    model.add(LSTM(100, activation='relu', input_shape=(n_input, n_fea
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')

    #Fitting Model

    model.fit(generator, epochs = 32, verbose=0)

    #Getting Predictions

    test_predictions = []

    first_eval_batch = X_train_scaled[-n_input:]
    current_batch = first_eval_batch.reshape((1, n_input, n_features))

    for i in range(len(y_train_scaled)):

        # get the prediction value for the first batch
```

```
57          current_pred = model.predict(current_batch)[0]
58
59          # append the prediction into the array
60          test_predictions.append(current_pred)
61
62          # use the prediction to update the batch and remove the first
63          current_batch = np.append(current_batch[:,1:,:],[[current_pred
64
65      #Inverse Transforming Preds into Timeseries Data:
66
67      preds = scaler.inverse_transform(test_predictions)
68
69      y_train['time'] = y_train.index
70
71      f_steps = y_train[['time']]
72
73      f_steps.index = [0,1,2,3,4,5,6,7,8,9,10,11]
74
75      preds = pd.DataFrame(preds)
76
77      preds = pd.concat([preds, f_steps], axis=1)
78
79      preds['time'] = pd.to_datetime(preds['time'])
80
81      preds.set_index('time', inplace = True)
82
83      preds.columns = ['value']
84
85      #Retrieving Sarimax Preds from Database
86
87      sarimax_preds = pd.read_csv(f'{RegionID}_preds.csv')
88
89      sarimax_preds.columns=['time', 'value']
90
91      sarimax_preds['time'] = pd.to_datetime(sarimax_preds['time'])
92
93      sarimax_preds.set_index('time', inplace = True)
94
95      sarimax_preds
96
97      # Showing Results
98
99      y_train_iso = y_train[['value']]
100
101     nn_mae = mean_absolute_error(preds, y_train_iso)
102
103     if graph=="Go":
104
105         plt.plot(preds, color='green')
106         plt.plot(y_train_iso, color='blue')
107         plt.plot(sarimax_preds, color ='purple')
108         plt.title(f'{RegionID} Comparative Analysis')
109
110         print(f'Mean Absolute Error for {RegionID} Neural Net: {mean_al
111         print(f'mean Absolute Error for {RegionID} SARIMAX: {mean_abso
112
113     else:
```

```
114
115         model.save(f'{RegionID}_model_{run}.h5')  # creates a HDF5 fil
116     #         del model  # deletes the existing model
117
118         return nn_mae
119
120
```

In [5]:
```
1 sarimax_vs_lstm(73177, graph="Stop", run=1)
```

```
2021-10-25 16:44:15.770473: I tensorflow/core/platform/cpu_feature_guard.
cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Netwo
rk Library (oneDNN) to use the following CPU instructions in performance-
critical operations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropria
te compiler flags.
2021-10-25 16:44:15.903719: I tensorflow/compiler/mlir/mlir_graph_optimiz
ation_pass.cc:185] None of the MLIR Optimization Passes are enabled (regi
stered 2)
```

Out[5]:
```
1117.8625139097373
```

In [12]:
```
1 # reconstructed_model = keras.models.load_model("73177_model_1.h5")
```

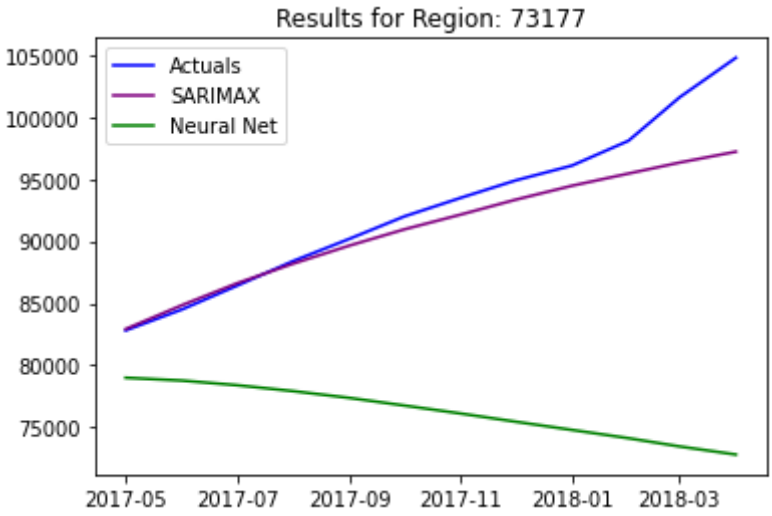
In [14]:

```
#Pulling Data

def reconstructed_test(RegionID, run):

    reconstructed_model = keras.models.load_model(f"{RegionID}_model_{

    test_df = pd.read_csv(f'{RegionID}.csv')

    test_df['time'] = pd.to_datetime(test_df['time'])

    test_df.set_index('time', inplace = True)

    test_df.columns = ['Drop', 'RegionName', 'RegionID', 'SizeRank', '
           'Metro', 'CountyName', 'value']

    test_df.drop('Drop', axis = 'columns', inplace = True)

    #Preparing X_train and y_train

    X_train = test_df[['value']].head(len(test_df) - 12)
    y_train = test_df[['value']].tail(12)

    #Scale Values

    scaler = MinMaxScaler()

    scaler.fit(X_train)

    X_train_scaled = scaler.transform(X_train)
    y_train_scaled = scaler.transform(y_train)

    #Time Series Genertor

    n_input = 12
    n_features = 1
    generator = TimeseriesGenerator(X_train_scaled, X_train_scaled, le

    #Getting Predictions

    test_predictions = []

    first_eval_batch = X_train_scaled[-n_input:]
    current_batch = first_eval_batch.reshape((1, n_input, n_features))

    for i in range(len(y_train_scaled)):

        # get the prediction value for the first batch
        current_pred = reconstructed_model.predict(current_batch)[0]

        # append the prediction into the array
        test_predictions.append(current_pred)

        # use the prediction to update the batch and remove the first
        current_batch = np.append(current_batch[:,1:,:],[[current_pred

    #Inverse Transforming Preds into Timeseries Data:
```

```
57
58      preds = scaler.inverse_transform(test_predictions)
59
60      y_train['time'] = y_train.index
61
62      f_steps = y_train[['time']]
63
64      f_steps.index = [0,1,2,3,4,5,6,7,8,9,10,11]
65
66      preds = pd.DataFrame(preds)
67
68      preds = pd.concat([preds, f_steps], axis=1)
69
70      preds['time'] = pd.to_datetime(preds['time'])
71
72      preds.set_index('time', inplace = True)
73
74      preds.columns = ['value']
75
76      #Retrieving Sarimax Preds from Database
77
78      sarimax_preds = pd.read_csv(f'73177_preds.csv')
79
80      sarimax_preds.columns=['time', 'value']
81
82      sarimax_preds['time'] = pd.to_datetime(sarimax_preds['time'])
83
84      sarimax_preds.set_index('time', inplace = True)
85
86      sarimax_preds
87
88      # Showing Results
89
90      y_train_iso = y_train[['value']]
91
92      nn_mae = mean_absolute_error(preds, y_train_iso)
93
94      plt.plot(y_train_iso, color='blue', label='Actuals')
95      plt.plot(sarimax_preds, color='purple', label='SARIMAX')
96      plt.plot(preds, color='green', label='Neural Net')
97      plt.title(f'Results for Region: {RegionID}')
98      plt.legend()
99
100     print(f'Mean Absolute Error of {nn_mae}')
```

In [15]:
```
1 reconstructed_test(73177, 2)
```

```
Mean Absolute Error of 16544.22244131565
```



In [12]:
```
1  # results = []
2
3  # for region in acc_growers:
4  #     for item in [1,2,3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,1
5  #         nn_mae = sarimax_vs_lstm(region, graph="Stop", run=item)
6  #         results.append([region, item, nn_mae])
7  #         print([region, nn_mae, item])
8
```

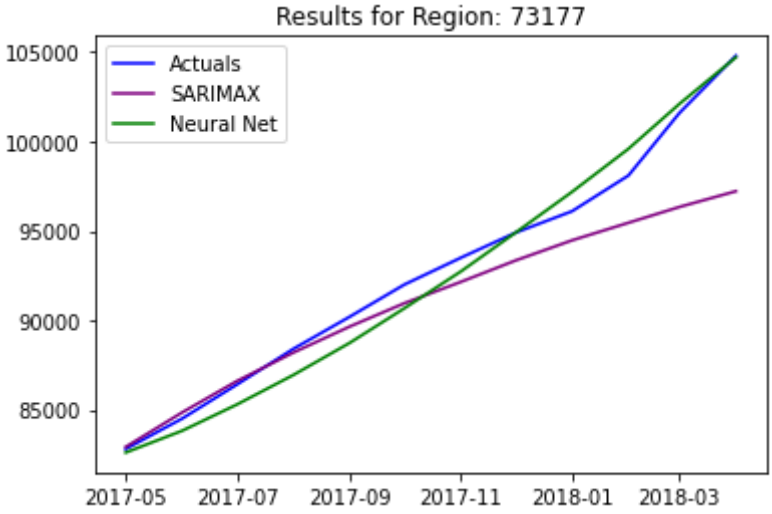
In [28]:
```
# best_models = []

# for item in acc_growers:
#     df = print_best_model(item)
#     best_models.append([round(df.iloc[0]['RegionID']), round(df.iloc[

# best_models = pd.DataFrame(best_models)

# best_models.columns = ['RegionID', 'Run', 'MAE']

# best_models

# best_models_stored = []

best_nn_models = pd.read_excel('Best NN Models.xlsx').iloc[:,1:]

best_nn_models
```

Out[28]:

	RegionID	Run	MAE
0	73177	18	848.61
1	91259	10	1173.71
2	66015	3	746.66
3	60706	9	7711.63
4	399593	5	4343.45

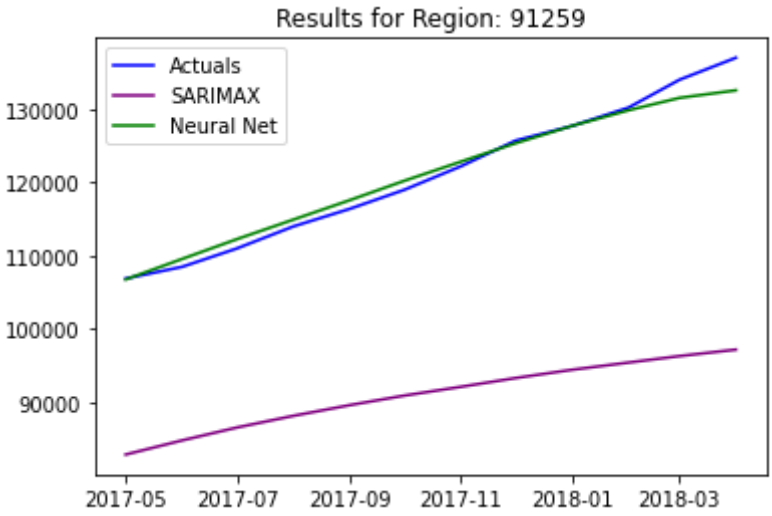
In [17]:
```
reconstructed_test(73177, 18)
```

```
Mean Absolute Error of 848.6076898872852
```



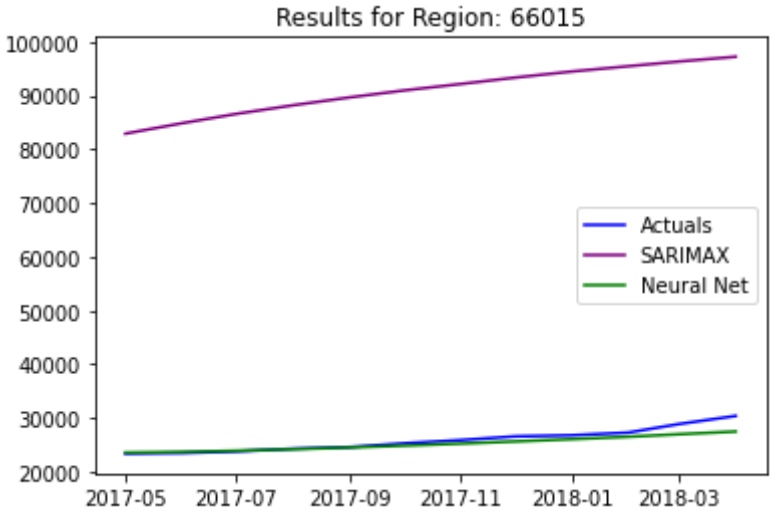
In [18]:
```
1 reconstructed_test(91259, 10)
```

```
Mean Absolute Error of 1173.708751797676
```



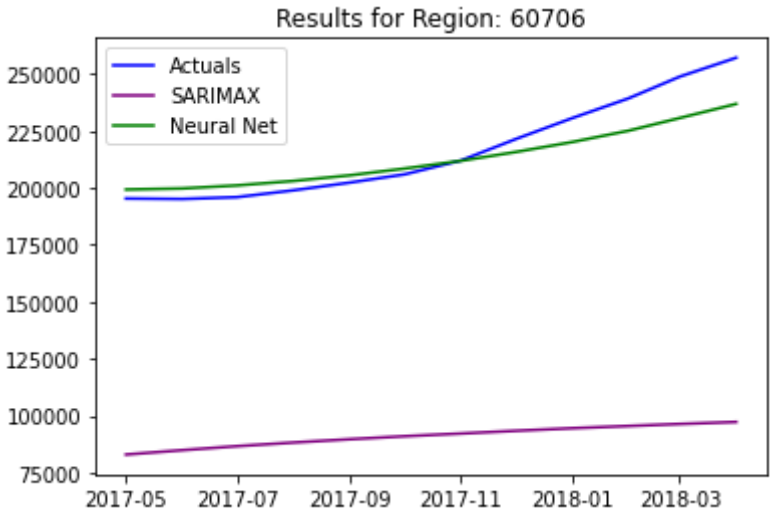
In [19]:
```
1 reconstructed_test(66015, 3)
```

```
Mean Absolute Error of 746.6633679966131
```



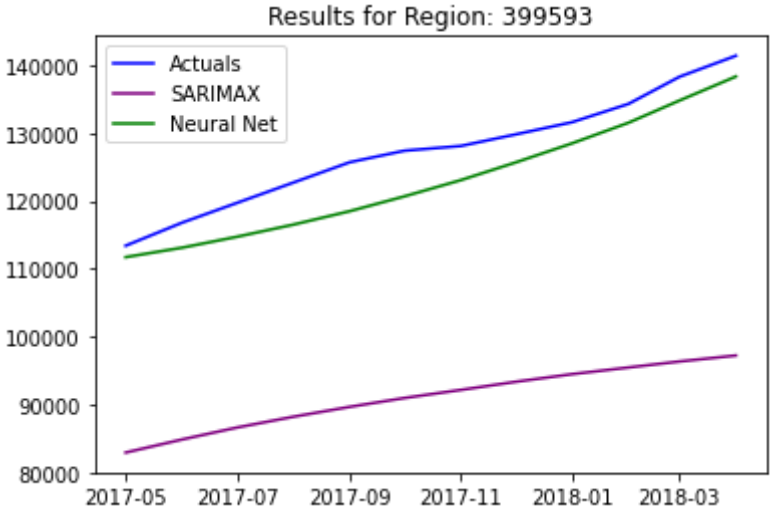
In [20]:
```
1 reconstructed_test(60706, 9)
```

```
Mean Absolute Error of 7711.633028214176
```



In [21]:
```
1 reconstructed_test(399593, 5)
```

```
Mean Absolute Error of 4343.453592558687
```



In [31]:
```
acc_growers_df
```

Out[31]:

	RegionID	City	State	Metro	CountyName	MAE	Comp Error
0	73177	Holiday	FL	Tampa	Pasco	1872.961447	0.02
1	91259	Fort Worth	TX	Dallas-Fort Worth	Tarrant	4541.077453	0.04
2	66015	Reading	PA	Reading	Berks	1921.952964	0.08
3	60706	Paterson	NJ	New York	Passaic	16572.053604	0.08
4	399593	Southfield	MI	Detroit	Oakland	10998.247600	0.09

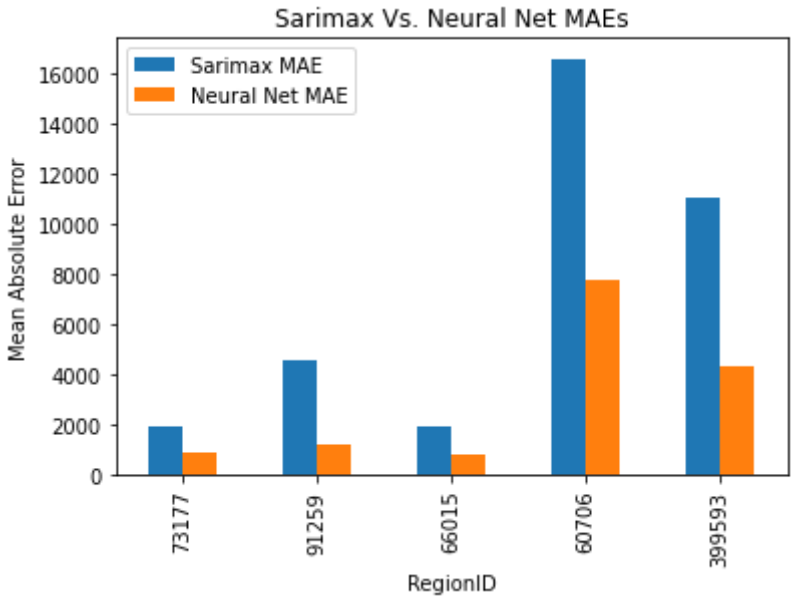
In [39]:
```
#Comparisson of Models

acc_growers_df = pd.read_excel('acc_growers.xlsx').iloc[:,1:]

#Joining Models

sarimax_vs_nn = acc_growers_df.merge(best_nn_models, how='inner', left_on=

sarimax_vs_nn.columns = ['RegionID', 'City', 'State', 'Metro', 'CountyName

sarimax_vs_nn[['RegionID', 'Sarimax MAE', 'Neural Net MAE']]
```

Out[39]:

	RegionID	Sarimax MAE	Neural Net MAE
0	73177	1872.961447	848.61
1	91259	4541.077453	1173.71
2	66015	1921.952964	746.66
3	60706	16572.053604	7711.63
4	399593	10998.247600	4343.45

In [51]:
```
1 sarimax_vs_nn.plot(x="RegionID", y=["Sarimax MAE", "Neural Net MAE"], k
2 plt.title('Sarimax Vs. Neural Net MAEs')
3 plt.ylabel('Mean Absolute Error')
```

Out[51]: Text(0, 0.5, 'Mean Absolute Error')



In [ ]:
```
1
```