Time Series Forecasting		
Praktikan	**Aslab**	
Nama: xxxx Nim: xxxx	Annur Hangga Prihadi	065001800028
	Faiz Kumara	065001800003

PRAKTIKUM 10

DATA SAINS DAN ANALITIK

Topik pertemuan praktikum ke-sepuluh adalah mengolah data produksi beras di suatu provinsi Indonesia berupa data time series untuk mencari hasil prediksi selama beberapa bulan ke depan menggunakan Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average(ARIMA), Holt Winters Exponential Smoothing.

Source Code:

Time Series ARMA&ARIMA:
https://github.com/hanggaa/PrakDSDA/blob/main/Prak_10_ARMA%26ARIMA_Beras.ipynb

Time Series Holt Winters:
https://github.com/hanggaa/PrakDSDA/blob/main/Prak_10_HW_Beras.ipynb

Latihan 1

ARMA dan ARIMA

1. **Install library yang dibutuhkan**

```
In [1]: import sys
        !{sys.executable} -m pip install pmdarima

Requirement already satisfied: pmdarima in c:\users\hangg\anaconda3\lib\site-packages (1.8.4)
Requirement already satisfied: joblib>=0.11 in c:\users\hangg\anaconda3\lib\site-packages (from pmd
```

2. **Memasang library yang dibutuhkan**

```
In [1]: import pandas as pd
        from datetime import datetime
        import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
        sns.set()
        from statsmodels.tsa.statespace.sarimax import SARIMAX
        from sklearn.metrics import mean_squared_error
        from statsmodels.tsa.arima.model import ARIMA
        import warnings
        warnings.filterwarnings("ignore")
```

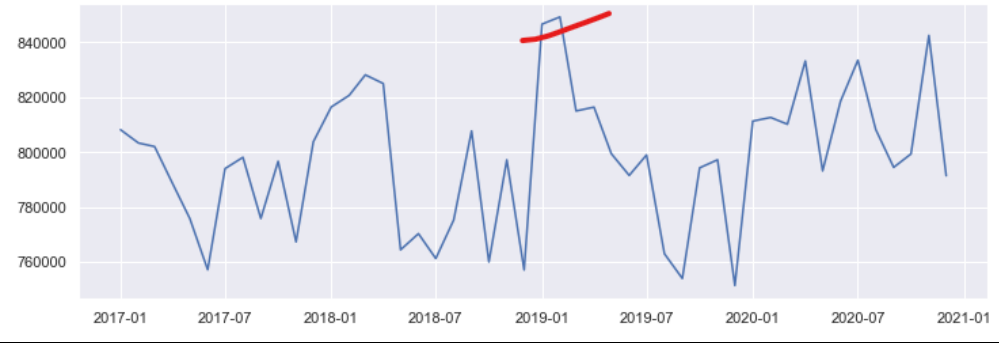
3. **Membaca data**

```
In [2]: beras = pd.read_csv('C:/Users/hangg/Downloads/Random Aslab/DSDA/Material/Supply_Beras.csv',index_col='Periode', parse_
        print(beras.shape)
        print(beras.head())
```

4. Visualisasi data

```
In [3]: plt.figure(figsize=(12,4))
        plt.plot(beras.index, beras['Produksi'])
```

```
Out[3]: [<matplotlib.lines.Line2D at 0x25e3e212610>]
```

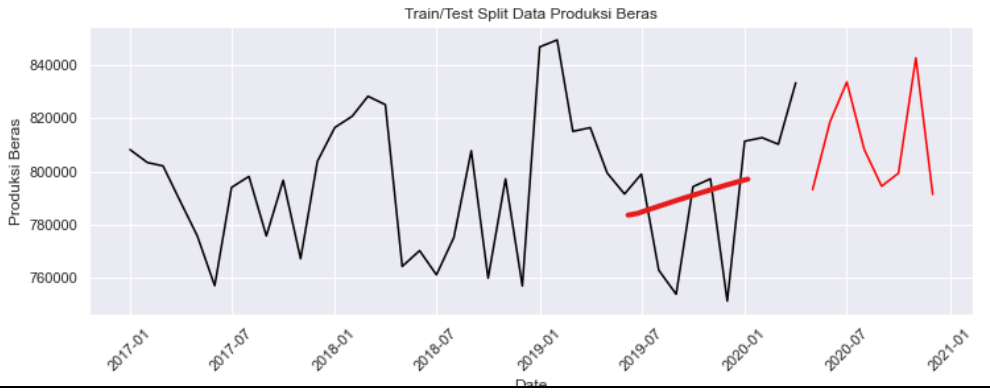


5. Memisah data

```
In [4]: train = beras[:40]
        test = beras[40:]
```

6. Visualisasi data latih dan uji

```
In [5]: plt.figure(figsize=(12,4))
        plt.plot(train, color = "black")
        plt.plot(test, color = "red")
        plt.ylabel('Produksi Beras')
        plt.xlabel('Date')
        plt.xticks(rotation=45)
        plt.title("Train/Test Split Data Produksi Beras")
        plt.show()
```



7. Implementasi metode ARMA

```
In [6]: y = train['Produksi']
        ARMAmodel = SARIMAX(y, order = (1, 0, 1))
        ARMAmodel = ARMAmodel.fit()
        y_pred_ARMA = ARMAmodel.get_forecast(15)
        y_pred_df_ARMA = y_pred_ARMA.conf_int(alpha = 0.05)
        y_pred_df_ARMA["Predictions"] = ARMAmodel.predict(start = y_pred_df_ARMA.index[0], end = y_pred_df_ARMA.index[-1])
        y_pred_out_ARMA = y_pred_df_ARMA["Predictions"]
```

8. Mencetak hasil prediksi metode ARMA

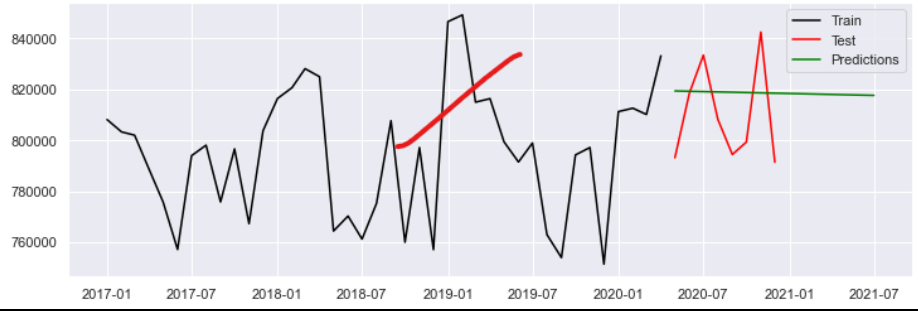
```
In [7]: y_pred_out_ARMA
```

```
Out[7]: 2020-05-01    819426.779361
        2020-06-01    819302.224742
        2020-07-01    819177.680056
        2020-08-01    819053.172300
        2020-09-01    818928.674471
        2020-10-01    818804.195565
        2020-11-01    818679.735581
        2020-12-01    818555.294515
        2021-01-01    818430.872364
        2021-02-01    818306.469125
        2021-03-01    818182.084797
        2021-04-01    818057.719374
```

9. Visualisasi hasil prediksi metode ARMA

```
In [8]: plt.figure(figsize=(12,4))
        plt.plot(train, color = "black", label = "Train")
        plt.plot(test, color = "red", label = "Test")
        plt.plot(y_pred_out_ARMA, color='green', label = 'Predictions')
        plt.legend()
```

```
Out[8]: <matplotlib.legend.Legend at 0x25e3e4262e0>
```



10. RMSE hasil prediksi metode ARMA

```
In [9]: arma_rmse = np.sqrt(mean_squared_error(y_pred_df_ARMA["Predictions"].values, y_pred_df_ARMA["Predictions"]))
        print("RMSE: ",arma_rmse)
```

```
RMSE:  0.0
```

11. Implementasi metode ARIMA

```
In [10]: ARIMAmodel = ARIMA(y, order = (1, 0, 1))
         ARIMAmodel = ARIMAmodel.fit()

         y_pred_ARIMA = ARIMAmodel.get_forecast(15)
         y_pred_df_ARIMA = y_pred_ARIMA.conf_int(alpha = 0.05)
         y_pred_df_ARIMA["Predictions"] = ARIMAmodel.predict(start = y_pred_df_ARIMA.index[0], end = y_pred_df_ARIMA.index[-1])
         y_pred_out_ARIMA = y_pred_df_ARIMA["Predictions"]
```

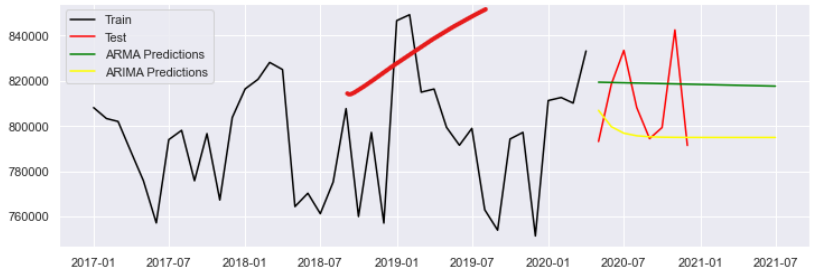
12. Mencetak hasil prediksi metode ARIMA

```
In [11]: y_pred_out_ARIMA
```

```
Out[11]: 2020-05-01    806960.204568
         2020-06-01    799728.509070
         2020-07-01    796858.819686
         2020-08-01    795720.066378
         2020-09-01    795268.185035
         2020-10-01    795088.868981
         2020-11-01    795017.712587
```

13. Visualisasi hasil prediksi metode ARIMA

```
In [12]: plt.figure(figsize=(12,4))
         plt.plot(train, color = "black", label = "Train")
         plt.plot(test, color = "red", label = "Test")
         plt.plot(y_pred_out_ARMA, color='green', label = 'ARMA Predictions')
         plt.plot(y_pred_out_ARIMA, color='Yellow', label = 'ARIMA Predictions')
         plt.legend()

Out[12]: <matplotlib.legend.Legend at 0x25e3e28c520>
```



14. RMSE hasil prediksi metode ARIMA

```
In [13]: arima_rmse = np.sqrt(mean_squared_error(y_pred_df_ARMA["Predictions"].values, y_pred_df_ARIMA["Predictions"]))
         print("RMSE: ",arima_rmse)

         RMSE:  22434.3033838866
```

15. Implementasi metode SARIMA

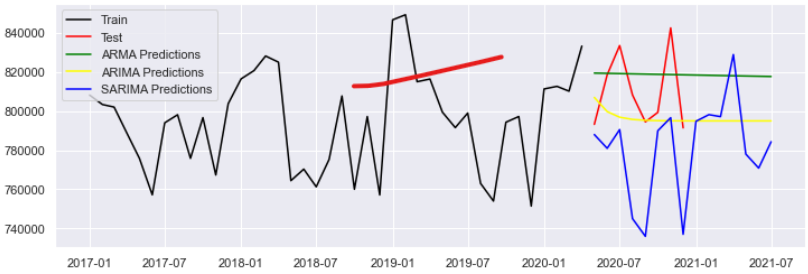
```
In [14]: SARIMAXmodel = SARIMAX(y, order = (1, 0, 1), seasonal_order=(2,2,2,12))
         SARIMAXmodel = SARIMAXmodel.fit()

         y_pred_SARIMA = SARIMAXmodel.get_forecast(15)
         y_pred_df_SARIMA = y_pred_SARIMA.conf_int(alpha = 0.05)
         y_pred_df_SARIMA["Predictions"] = SARIMAXmodel.predict(start = y_pred_df_SARIMA.index[0], end = y_pred_df_SARIMA.index[-1])
         y_pred_out_SARIMA = y_pred_df_SARIMA["Predictions"]
```

16. Mencetak hasil prediksi metode SARIMA

```
In [15]: y_pred_out_SARIMA

Out[15]: 2020-05-01    787982.684464
         2020-06-01    780934.325948
         2020-07-01    790521.997022
         2020-08-01    744968.911394
         2020-09-01    735890.444175
         2020-10-01    789935.441868
         2020-11-01    796623.203484
```

17. Visualisasi hasil prediksi metode SARIMA

```
In [16]: plt.figure(figsize=(12,4))
         plt.plot(train, color = "black", label = "Train")
         plt.plot(test, color = "red", label = "Test")
         plt.plot(y_pred_out_ARMA, color='green', label = 'ARMA Predictions')
         plt.plot(y_pred_out_ARIMA, color='Yellow', label = 'ARIMA Predictions')
         plt.plot(y_pred_out_SARIMA, color='Blue', label = 'SARIMA Predictions')
         plt.legend()

Out[16]: <matplotlib.legend.Legend at 0x25e3e554160>
```



18. RMSE hasil prediksi metode SARIMA

```
In [17]: sarima_rmse = np.sqrt(mean_squared_error(y_pred_df_ARMA["Predictions"].values, y_pred_df_SARIMA["Predictions"]))
         print("SARIMA RMSE: ",sarima_rmse)

SARIMA RMSE:  44787.685206632435
```

Holt Winters

1. Memasang library yang dibutuhkan

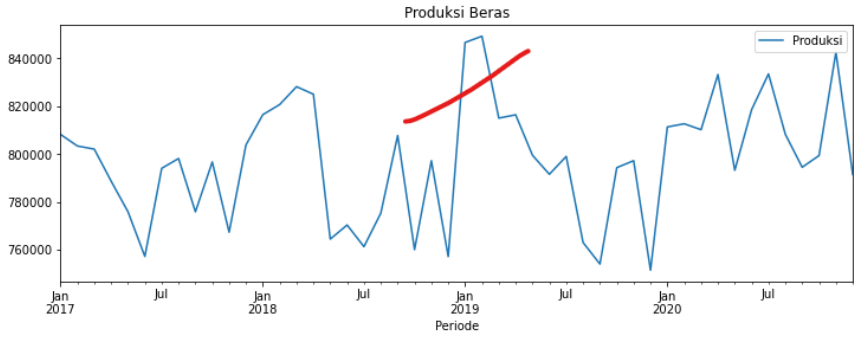
```
In [1]: import pandas as pd
        from matplotlib import pyplot as plt
        from statsmodels.tsa.seasonal import seasonal_decompose
        from statsmodels.tsa.holtwinters import SimpleExpSmoothing
        from statsmodels.tsa.holtwinters import ExponentialSmoothing
        from sklearn.metrics import mean_absolute_error,mean_squared_error
        from datetime import datetime
        import warnings
        warnings.filterwarnings("ignore")
```

2. Membaca data

```
In [2]: beras = pd.read_csv('C:/Users/hangg/Downloads/Random Aslab/DSDA/Material/Supply_Beras.csv',index_col='Periode', parse_
        print(beras.shape)
        print(beras.head())
```

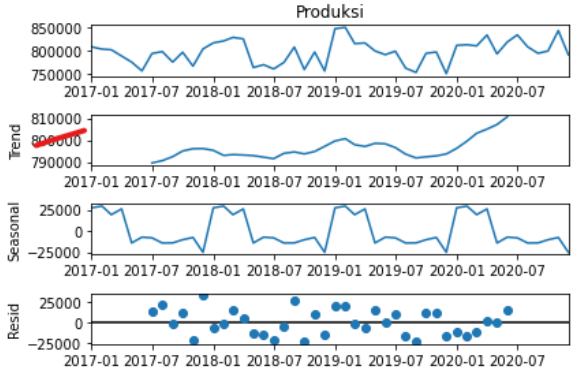
3. Visualisasi data

```
In [3]: beras[['Produksi']].plot(title='Produksi Beras', figsize = (12,4))

Out[3]: <AxesSubplot:title={'center':'Produksi Beras'}, xlabel='Periode'>
```



4. Melihat pola trend dan musiman

```
In [4]: decompose_result = seasonal_decompose(beras['Produksi'],model='additive')
        decompose_result.plot();
```

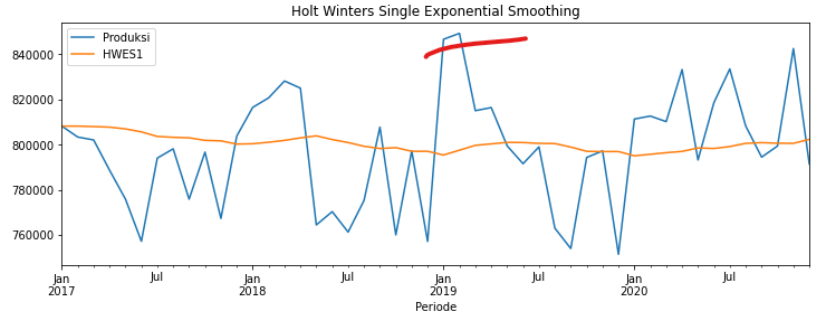


5. Mengatur frekuensi datatime

In [5]:
```
beras.index.freq = 'MS'
m = 12
alpha = 1/(2*m)
```

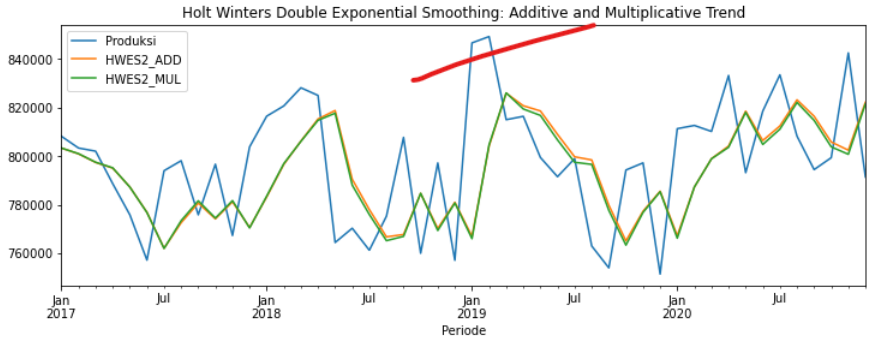
6. Implementasi Single Exponential Smoothing

In [6]:
```
beras['HWES1'] = SimpleExpSmoothing(beras['Produksi']).fit(smoothing_level=alpha,optimized=False,use_brute=True).fittedvalues
beras[['Produksi','HWES1']].plot(title='Holt Winters Single Exponential Smoothing', figsize=(12,4));
```



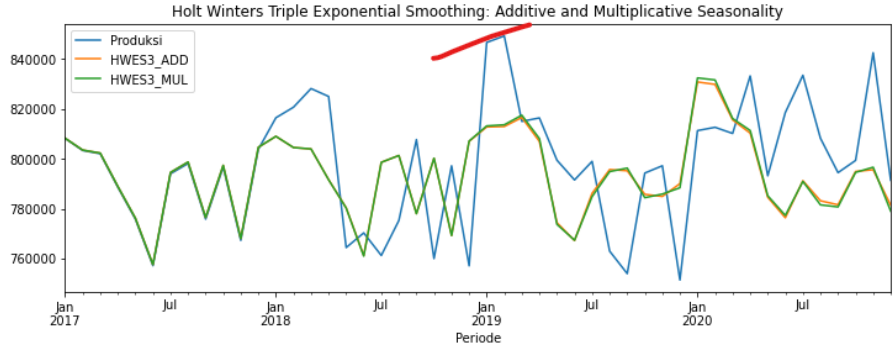
7. Implementasi Double Exponential Smoothing

In [7]:
```
beras['HWES2_ADD'] = ExponentialSmoothing(beras['Produksi'],trend='add').fit().fittedvalues
beras['HWES2_MUL'] = ExponentialSmoothing(beras['Produksi'],trend='mul').fit().fittedvalues
beras[['Produksi','HWES2_ADD','HWES2_MUL']].plot(title='Holt Winters Double Exponential Smoothing: Addi
```



8. Implementasi Triple Exponential Smoothing

In [8]:
```
beras['HWES3_ADD'] = ExponentialSmoothing(beras['Produksi'],trend='add',seasonal='add',seasonal_perio
beras['HWES3_MUL'] = ExponentialSmoothing(beras['Produksi'],trend='mul',seasonal='mul',seasonal_perio
beras[['Produksi','HWES3_ADD','HWES3_MUL']].plot(title='Holt Winters Triple Exponential Smoothing: Ad
```



9. Membaca prediksi data

```
In [9]: f_prod= pd.read_csv('C:/Users/hangg/Downloads/Random Aslab/DSDA/Material/Supply_Beras.csv',i
```

10. Mengatur frekuensi datatime

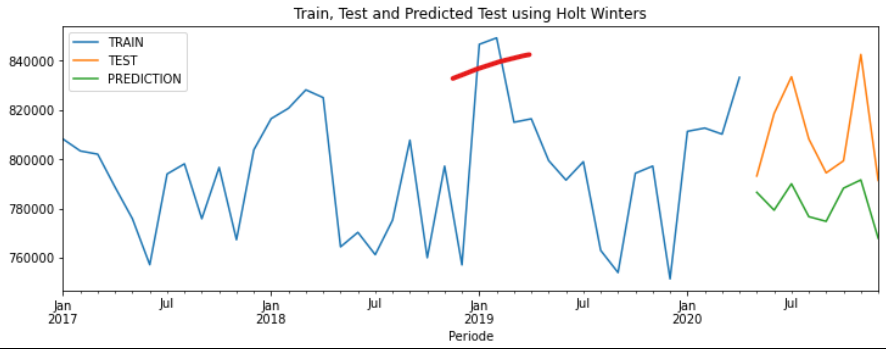
```
In [10]: f_prod.index.freq='MS'
```

11. Mengetahui dimensi prediksi data

```
In [11]: f_prod.shape

Out[11]: (48, 1)
```

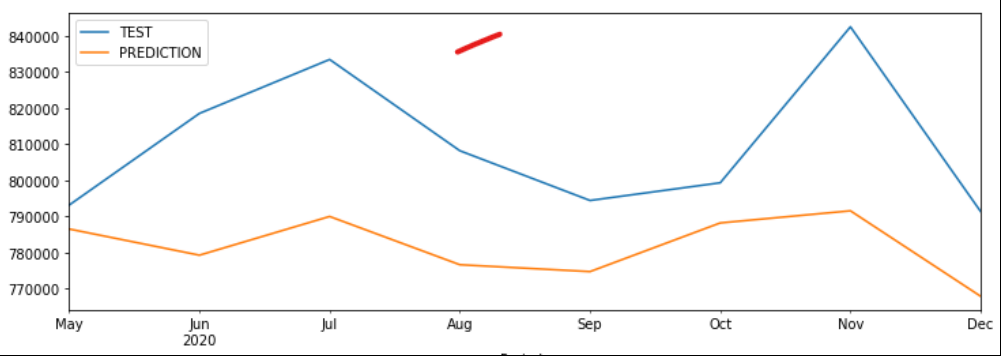
12. Memisah data dan implementasi Triple Exponential Smoothing

```
In [12]: train_prod = f_prod[:40]
         test_prod = f_prod[40:]

In [13]: fitted_model = ExponentialSmoothing(train_prod['Produksi'],trend='mul',seasonal='mul',seasonal_periods=12).fit()
         test_predictions = fitted_model.forecast(len(test_prod))
         train_prod['Produksi'].plot(legend=True,label='TRAIN')
         test_prod['Produksi'].plot(legend=True,label='TEST',figsize=(12,4))
         test_predictions.plot(legend=True,label='PREDICTION')
         plt.title('Train, Test and Predicted Test using Holt Winters')

Out[13]: Text(0.5, 1.0, 'Train, Test and Predicted Test using Holt Winters')
```



13. Visualisasi hasil prediksi

```
In [14]: test_prod['Produksi'].plot(legend=True,label='TEST',figsize=(12,4))
         test_predictions.plot(legend=True,label='PREDICTION');
```



14. MAE dan MSE hasil prediksi

In [15]:
```
print(f'Mean Absolute Error = {mean_absolute_error(test_prod,test_predictions)}')
print(f'Mean Squared Error = {mean_squared_error(test_prod,test_predictions)}')
```

```
Mean Absolute Error = 28278.732970180994
Mean Squared Error = 1016794323.9722463
```

15. Melihat data periode akhir

In [16]:
```
f_prod.tail()
```

Out[16]:

Periode	Produksi
2020-08-01	808237
2020-09-01	794454
2020-10-01	799353
2020-11-01	842555
2020-12-01	791401

16. Olah prediksi data

In [17]:
```
prediksi = fitted_model.predict(start=datetime(2021,1,1), end = datetime(2021,5,1))
```

17. Mencetak hasil prediksi

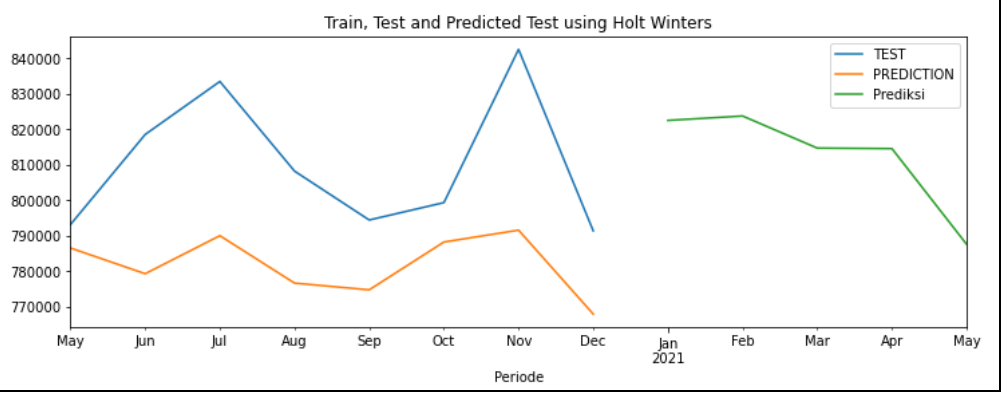
In [18]:
```
print(prediksi)
```

```
2021-01-01    822535.065885
2021-02-01    823778.754679
2021-03-01    814731.265145
2021-04-01    814585.724723
2021-05-01    787600.489025
Freq: MS, dtype: float64
```

18. Visualisasi hasil prediksi

In [19]:
```
test_prod['Produksi'].plot(legend=True,label='TEST',figsize=(12,4))
test_predictions.plot(legend=True,label='PREDICTION');
prediksi.plot(legend=True, label='Prediksi')
plt.title('Train, Test and Predicted Test using Holt Winters')
```

Out[19]:
```
Text(0.5, 1.0, 'Train, Test and Predicted Test using Holt Winters')
```



Latihan 2

1. Menurut anda kapan menggunakan metode Auto Regressive (ARMA, ARIMA, SARIMA) dan Holt Winters?
2. Menurut anda metode mana yang cocok untuk prediksi data produksi beras latihan 1?
3. Menurut anda kasus data time series yang seperti apa yang cocok diolah menggunakan metode Auto Regressive (ARMA,, ARIMA, SARIMA)? Contoh = Data time series harga beras
4. Menurut anda kasus data time series yang seperti apa yang cocok diolah menggunakan metode Holt Winters? Contoh = Data time series harga beras