In [1]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.set_option('display.max_rows', 5000)
pd.set_option('display.max_columns', 5000)
pd.set_option('display.width', 1000)
pd.options.display.max_colwidth = 1000
sns.set(style = 'darkgrid')
```

In [2]:
```
df = pd.read_csv("/Users/senth/Downloads/train_1.csv")
```

In [3]:
```
df.shape
```

Out[3]:
```
(145063, 551)
```

In [4]:
```
Exog_Campaign_eng = pd.read_csv("/Users/senth/Downloads/Exog_Campaign_eng")
```

In [5]:
```
Exog_Campaign_eng.shape
```

Out[5]:
```
(550, 1)
```

In [6]:
```
df.Page.sample(20)
```

Out[6]:
```
57160                                       藤岡麻美_ja.wikipedia.org_mobile-web_all-agents
128531                                   Le_Caravage_fr.wikipedia.org_all-access_spider
134374                                          木村佳乃_ja.wikipedia.org_all-access_spider
59178                         ねじ巻き精霊戦記_天鏡のアルデラミン_ja.wikipedia.org_mobile-web_all-agents
30959                                        黃心穎_zh.wikipedia.org_all-access_all-agents
116610                                Frankreich_de.wikipedia.org_mobile-web_all-agents
110590                     Diathrausta_minutalis_en.wikipedia.org_all-access_all-agents
44256         Category:Videos_of_animal_sex_commons.wikimedia.org_all-access_all-agents
49221                                 Kray-Zwillinge_de.wikipedia.org_all-access_spider
45341     Category:Vintage_nude_photographs_commons.wikimedia.org_all-access_all-agents
57489                                       旋毛虫症_ja.wikipedia.org_mobile-web_all-agents
40855                              Vince_McMahon_en.wikipedia.org_all-access_all-agents
74081                                    Lucifer_en.wikipedia.org_mobile-web_all-agents
60159                                      桜田ひより_ja.wikipedia.org_mobile-web_all-agents
72233                            Compuesto_orgánico_es.wikipedia.org_desktop_all-agents
7537                             Bernard-Henri_Lévy_fr.wikipedia.org_desktop_all-agents
84580                                Special:MyPage_www.mediawiki.org_all-access_spider
63690                                          陳亭妃_zh.wikipedia.org_desktop_all-agents
132831                                          日高里菜_ja.wikipedia.org_all-access_spider
120873                                   ライトセーバー_ja.wikipedia.org_all-access_all-agents
Name: Page, dtype: object
```

In [7]:
```
df.Page.str.split("_").apply(lambda x:x[3]).head(20)
```

Out[7]:
```
0               spider
1               spider
2               spider
3               spider
4                 Love
5               spider
6               spider
7               spider
8               spider
9               spider
10              spider
11    zh.wikipedia.org
12                 are
13              spider
14              spider
15              spider
16              spider
17          all-access
18          all-access
19              spider
Name: Page, dtype: object
```

In [8]:
```
data = df.copy()
```

In [9]:
```
data.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
data.dtypes.sample(10)
```

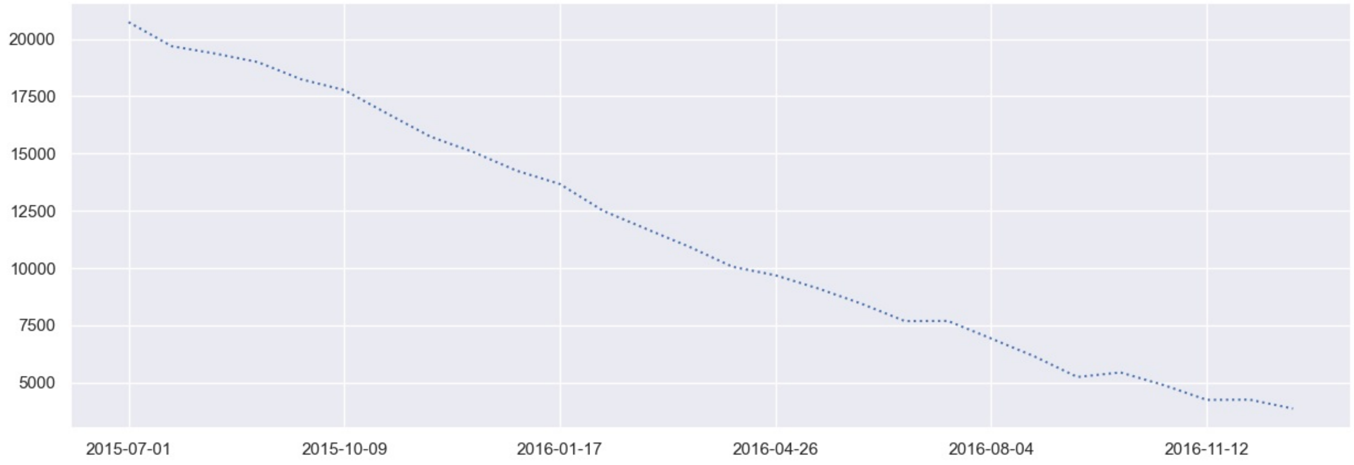
```
Out[10]: 2016-03-23    float64
         2016-05-14    float64
         2015-07-27    float64
         2016-01-18    float64
         2016-04-16    float64
         2015-11-07    float64
         2016-05-13    float64
         2015-12-21    float64
         2015-08-09    float64
         2015-11-25    float64
         dtype: object
```

```
In [11]: indexes = data.head(2).columns[1:][range(0,549,20)].values
         indexes
```

```
Out[11]: array(['2015-07-01', '2015-07-21', '2015-08-10', '2015-08-30',
                '2015-09-19', '2015-10-09', '2015-10-29', '2015-11-18',
                '2015-12-08', '2015-12-28', '2016-01-17', '2016-02-06',
                '2016-02-26', '2016-03-17', '2016-04-06', '2016-04-26',
                '2016-05-16', '2016-06-05', '2016-06-25', '2016-07-15',
                '2016-08-04', '2016-08-24', '2016-09-13', '2016-10-03',
                '2016-10-23', '2016-11-12', '2016-12-02', '2016-12-22'],
               dtype=object)
```

```
In [12]: plt.figure(figsize=(15, 5))

         data.isna().sum()[indexes].plot(linestyle='dotted')
```

```
Out[12]: <Axes: >
```



from above plot , we can observe that with time , null values are decreasing.

recent dates have lesser null values

that means newer pages will have no data of prior to that page hosting date.

```
In [13]: data.fillna(0,inplace =True)
```

```
In [14]: data.isnull().sum()[indexes]
```

Out[14]:

```
2015-07-01    0
2015-07-21    0
2015-08-10    0
2015-08-30    0
2015-09-19    0
2015-10-09    0
2015-10-29    0
2015-11-18    0
2015-12-08    0
2015-12-28    0
2016-01-17    0
2016-02-06    0
2016-02-26    0
2016-03-17    0
2016-04-06    0
2016-04-26    0
2016-05-16    0
2016-06-05    0
2016-06-25    0
2016-07-15    0
2016-08-04    0
2016-08-24    0
2016-09-13    0
2016-10-03    0
2016-10-23    0
2016-11-12    0
2016-12-02    0
2016-12-22    0
dtype: int64
```

Exploratory Data Analysis

Extracting Language

In [15]:

```
data.Page[0]
```

Out[15]:

```
'2NE1_zh.wikipedia.org_all-access_spider'
```

In [16]:

```
import re
re.findall(r'_(.{2}).wikipedia.org_', "2NE1_zh.wikipedia.org_all-access_spider")
```

Out[16]:

```
['zh']
```

In [17]:

```
data.Page.str.findall(pat="_(.{2}).wikipedia.org_").sample(10)
```

Out[17]:

```
53468     [fr]
96021     [es]
10788     [en]
27884     [fr]
101940    [ru]
122097    [ja]
124205    [ru]
143094    [es]
66008     [de]
41022     [en]
Name: Page, dtype: object
```

In [18]:

```
# extracting language
def Extract_Language(name):
  if len(re.findall(r'_(.{2}).wikipedia.org_', name)) == 1 :
    return re.findall(r'_(.{2}).wikipedia.org_', name)[0]
  else:
    return 'Unknown'
```

In [19]:

```
data["Language"] = data["Page"].map(Extract_Language)
```

In [20]:

```
data["Language"].unique()
```

Out[20]:

```
array(['zh', 'fr', 'en', 'Unknown', 'ru', 'de', 'ja', 'es'], dtype=object)
```

In [21]:

```
dict_ ={'de':'German',
        'en':'English',
        'es': 'Spanish',
        'fr': 'French',
        'ja': 'Japenese' ,
        'ru': 'Russian',
        'zh': 'Chinese',
        'Unknown': 'Unknown_Language'}
```

```
data["Language"] = data["Language"].map(dict_)
```

In [22]:

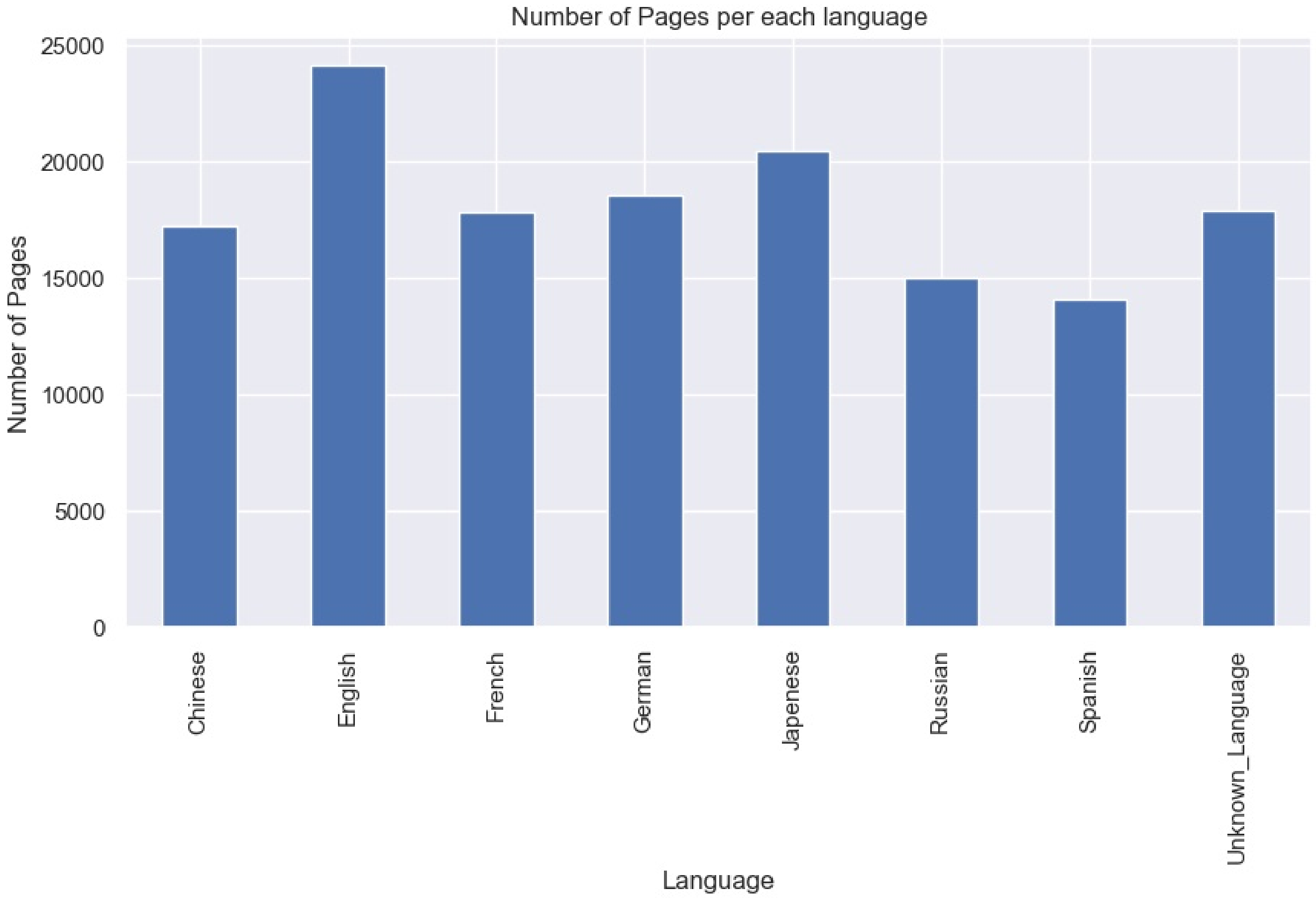
```
data.head()
```

Out[22]:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	2015-07-10	2015-07-11	2015-07-12	2015-07-13
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	24.0	19.0	10.0	14.0
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	4.0	41.0	65.0	57.0
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	4.0	1.0	1.0	1.0
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	16.0	16.0	11.0	23.0
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [23]:

```
plt.figure(figsize=(10, 5))

data.groupby("Language")["Page"].count().plot(kind="bar")
plt.xlabel("Language")
plt.ylabel("Number of Pages")
plt.title("Number of Pages per each language")
plt.show()
```



In [24]:

```
from locale import normalize
data["Language"].value_counts(normalize=True) * 100
```

Out[24]:

```
English             16.618986
Japenese            14.084225
German              12.785479
Unknown_Language    12.308445
French              12.271909
Chinese             11.876909
Russian             10.355501
Spanish              9.698545
Name: Language, dtype: float64
```

Extracting ACCESS TYPE

In [25]:

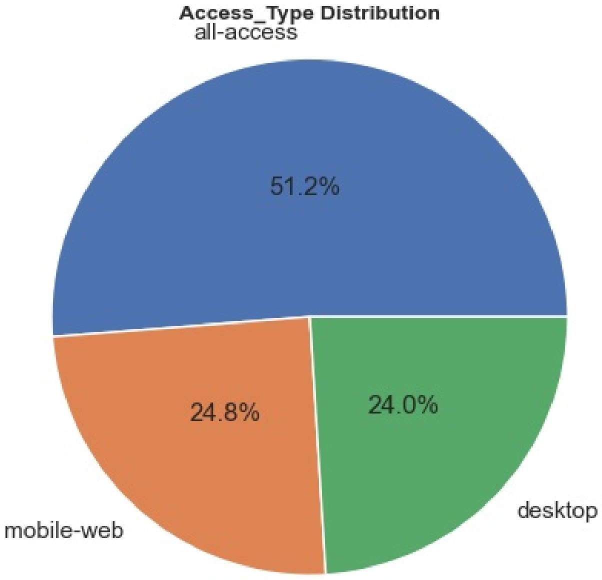
```
data["Access_Type"] = data.Page.str.findall(r'all-access|mobile-web|desktop').apply(lambda x:x[0])
```

In [26]:

```
data["Access_Type"].value_counts(dropna=False, normalize=True)
```

```
Out[26]: all-access    0.512295
         mobile-web    0.247748
         desktop       0.239958
         Name: Access_Type, dtype: float64
```

```
In [27]: x = (data["Access_Type"].value_counts(dropna= False, normalize=True) * 100).values
         y = (data["Access_Type"].value_counts(dropna= False, normalize=True) * 100).index

         plt.pie(x,labels= y,radius=1.5,  autopct='%1.1f%%', pctdistance=0.5 )
         plt.title(f'Access_Type Distribution', fontsize = 10, fontweight = 'bold')
         plt.axis('equal')
         plt.show()
```



Exrtacting ACCESS ORIGIN

```
In [28]: data.Page.sample(20)
```

```
Out[28]: 1507                                                    陸貞傳奇_zh.wikipedia.org_all-access_spider
         32551                                       Battle_of_Inchon_en.wikipedia.org_all-access_spider
         15086               File:Liverpool_FC_1892-1896_kit.jpg_commons.wikimedia.org_all-access_spider
         28168                                               100毛_zh.wikipedia.org_all-access_all-agents
         38092                                 Hamilton_(musical)_en.wikipedia.org_all-access_all-agents
         67313                                          Billy_Chapin_de.wikipedia.org_desktop_all-agents
         134463                                                  金田勝年_ja.wikipedia.org_all-access_spider
         69244           Relegation_zur_deutschen_Fußball-Bundesliga_de.wikipedia.org_desktop_all-agents
         45918     Commons:Wiki_Loves_Monuments_2016_in_Peru_commons.wikimedia.org_all-access_all-agents
         124507                         Заворотнюк,_Анастасия_Юрьевна_ru.wikipedia.org_all-access_spider
         66189                                          Mark_Forster_de.wikipedia.org_desktop_all-agents
         23815                                      Charlie_Hebdo_fr.wikipedia.org_all-access_all-agents
         37303                                    Abraham_Lincoln_en.wikipedia.org_all-access_all-agents
         143883                        Copa_Mundial_de_Fútbol_de_2014_es.wikipedia.org_all-access_spider
         19981                              How_to_contribute/ml_www.mediawiki.org_all-access_all-agents
         129157                           Royale_Entente_Bertrigeoise_fr.wikipedia.org_all-access_spider
         58728                                               桑田佳祐_ja.wikipedia.org_mobile-web_all-agents
         12031                                  Thor:_The_Dark_World_en.wikipedia.org_desktop_all-agents
         95692                                     Mahatma_Gandhi_es.wikipedia.org_mobile-web_all-agents
         142483                                            Marcos_Ana_es.wikipedia.org_all-access_spider
         Name: Page, dtype: object
```

```
In [29]: data.Page.str.findall(r'spider|agents').apply(lambda x:x[0]).isna().sum()
```

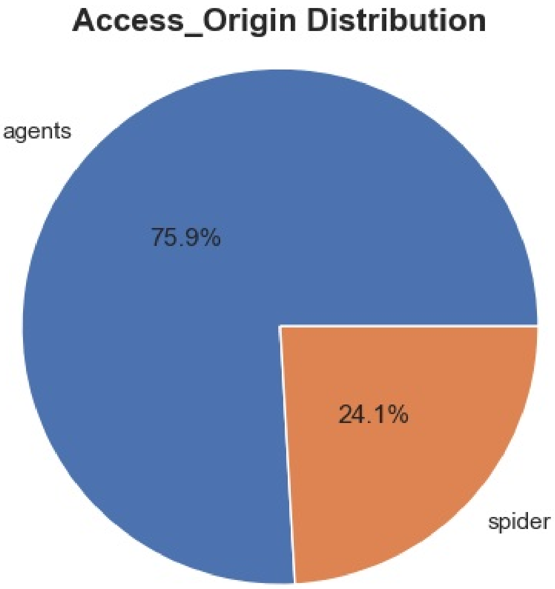
```
Out[29]: 0
```

```
In [30]: data["Access_Origin"] =  data.Page.str.findall(r'spider|agents').apply(lambda x:x[0])
```

```
In [31]: data["Access_Origin"].value_counts(dropna= False, normalize=True) * 100
```

```
Out[31]: agents    75.932526
         spider    24.067474
         Name: Access_Origin, dtype: float64
```

```
In [32]: x = (data["Access_Origin"].value_counts(dropna= False, normalize=True) * 100).values
         y = (data["Access_Origin"].value_counts(dropna= False, normalize=True) * 100).index

         plt.pie(x,labels= y,radius=1.5,  autopct='%1.1f%%', pctdistance=0.5 )
         plt.title(f'Access_Origin Distribution', fontsize = 15, fontweight = 'bold')
         plt.axis('equal')
plt.show()
```



In [33]:

```
data
```

Out[33]:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
145058	Underworld_(serie_de_películas)_es.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145059	Resident_Evil:_Capítulo_Final_es.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145060	Enamorándome_de_Ramón_es.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145061	Hasta_el_último_hombre_es.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145062	Francisco_el_matemático_(serie_de_televisión_de_2017)_es.wikipedia.org_all-access_spider	0.0	0.0	0.0	0.0	0.0	0.0	0.0

145063 rows × 554 columns

In [34]:

```
data.groupby("Language").mean()
```

Out[34]:

	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08
Language								
Chinese	240.582042	240.941958	239.344071	241.653491	257.779674	259.114864	258.832260	265.589529
English	3513.862203	3502.511407	3325.357889	3462.054256	3575.520035	3849.736021	3643.523063	3437.871080
French	475.150994	478.202000	459.837659	491.508932	482.557746	502.741209	485.945399	476.998820
German	714.968405	705.229741	676.877231	621.145145	722.076185	794.832480	770.814256	782.077641
Japenese	580.647056	666.672801	602.289805	756.509177	725.720914	632.399148	615.184181	611.462337
Russian	629.999601	640.902876	594.026295	558.728132	595.029157	640.986287	626.293436	623.360205
Spanish	1085.972919	1037.814557	954.412680	896.050750	974.508210	1110.637145	1082.568342	1050.669557
Unknown_Language	83.479922	87.471857	82.680538	70.572557	78.214562	89.720190	94.939457	99.096724

In [35]:

```
pd.set_option('display.max_rows', 500)
```

In [36]:

```
aggregated_data = data.groupby("Language").mean().T.drop("Unknown_Language",axis = 1).reset_index()
```

In [37]:

```
aggregated_data["index"] = pd.to_datetime(aggregated_data["index"])
aggregated_data = aggregated_data.set_index("index")
```

In [38]:

```
aggregated_data
```

Out[38]:

Language	Chinese	English	French	German	Japenese	Russian	Spanish
index							
2015-07-01	240.582042	3513.862203	475.150994	714.968405	580.647056	629.999601	1085.972919
2015-07-02	240.941958	3502.511407	478.202000	705.229741	666.672801	640.902876	1037.814557
2015-07-03	239.344071	3325.357889	459.837659	676.877231	602.289805	594.026295	954.412680
2015-07-04	241.653491	3462.054256	491.508932	621.145145	756.509177	558.728132	896.050750
2015-07-05	257.779674	3575.520035	482.557746	722.076185	725.720914	595.029157	974.508210
...	...	...	...	...	...	...	...
2016-12-27	376.019618	6040.680728	858.413100	1085.095379	789.158680	1001.209426	1133.367901
2016-12-28	378.048639	5860.227559	774.155769	1032.640804	790.500465	931.987685	1178.290923
2016-12-29	350.719427	6245.127510	752.712954	994.657141	865.483236	897.282452	1112.171085
2016-12-30	354.704452	5201.783018	700.543422	949.265649	952.018354	803.271868	821.671405
2016-12-31	365.579256	5127.916418	646.258342	893.013425	1197.239440	880.244508	787.399531

550 rows × 7 columns

In [39]:

```
aggregated_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 550 entries, 2015-07-01 to 2016-12-31
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Chinese   550 non-null    float64
 1   English   550 non-null    float64
 2   French    550 non-null    float64
 3   German    550 non-null    float64
 4   Japenese  550 non-null    float64
 5   Russian   550 non-null    float64
 6   Spanish   550 non-null    float64
dtypes: float64(7)
memory usage: 34.4 KB
```

In [40]:

```
aggregated_data.index
```

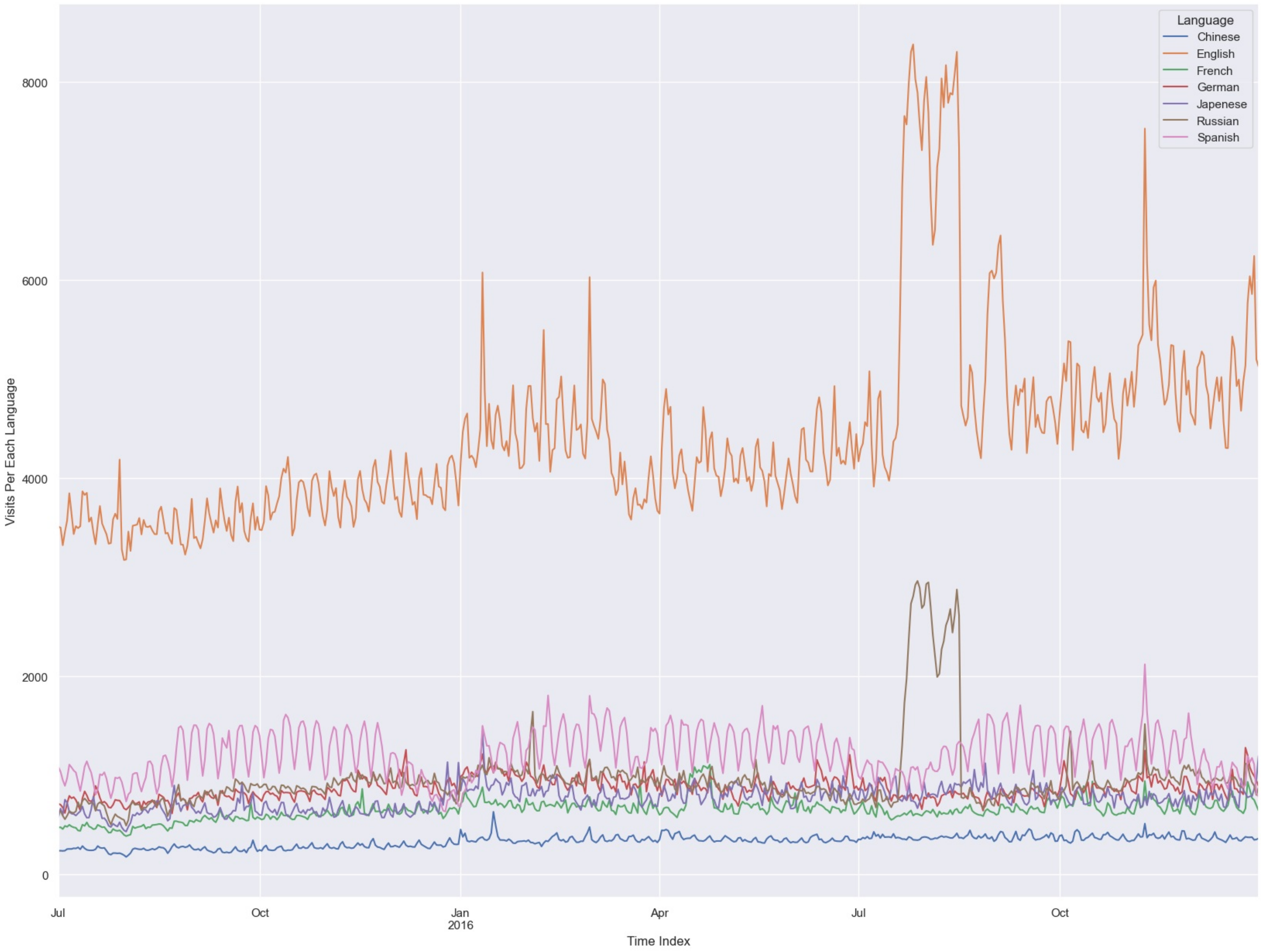
Out[40]:

```
DatetimeIndex(['2015-07-01', '2015-07-02', '2015-07-03', '2015-07-04', '2015-07-05', '2015-07-06', '2015-07-07'
, '2015-07-08', '2015-07-09', '2015-07-10',
               ...
               '2016-12-22', '2016-12-23', '2016-12-24', '2016-12-25', '2016-12-26', '2016-12-27', '2016-12-28'
, '2016-12-29', '2016-12-30', '2016-12-31'], dtype='datetime64[ns]', name='index', length=550, freq=None)
```

Visualising Time Series for each languages

In [41]:

```
plt.rcParams['figure.figsize'] = (20, 15)

aggregated_data.plot()

plt.xlabel("Time Index")
plt.ylabel("Visits Per Each Language")
plt.show()
```



Hypothesis Testing : if Time Series is Stationary or Trending

Null Hypothesis: The series is Non-Stationary

Alternative Hypothesis: The series is Stationary

significant value : 0.05 (alpha)

if p-value > 0.05 : we failed to reject Null hypothesis:

That means the series is Non-Stationart if p-value <= 0.05: we reject Null Hypothesis

that means the time series in Stationary

In [42]:

```
import statsmodels.api as sm
```

In [43]:

```
def Dickey_Fuller_test(ts,significances_level = 0.05):
    p_value = sm.tsa.stattools.adfuller(ts)[1]
    if p_value <= significances_level:
        print("Time Series is Stationary")
    else:
        print("Time Series is NOT Stationary")
    print("P_value is: ", p_value)
```

In [44]:

```
for Language in aggregated_data.columns:
  print(Language)
  print(Dickey_Fuller_test(aggregated_data[Language],significances_level = 0.05))
  print()
  print()
```

```
Chinese
Time Series is NOT Stationary
P_value is:  0.44744579229311354
None


English
Time Series is NOT Stationary
P_value is:  0.18953359279992427
None


French
Time Series is NOT Stationary
P_value is:  0.05149502195245779
None


German
Time Series is NOT Stationary
P_value is:  0.14097382319729113
None


Japenese
Time Series is NOT Stationary
P_value is:  0.10257133898557619
None


Russian
Time Series is Stationary
P_value is:  0.0018649376536617962
None


Spanish
Time Series is Stationary
P_value is:  0.033588590844791315
None
```

Based on DickeyFuller test of Stationarity , we can observe Spanish and Russian languages Pages visits Time series are stationary.

Chinese, English , German , Japanese and French are not stationary.

In [45]:

```
TS_English = aggregated_data.English
```

In [46]:

```
def adf_test(timeseries):
    print ('Results of Dickey-Fuller Test:')

    dftest = sm.tsa.stattools.adfuller(timeseries, autolag='AIC')
    df_output = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations U
    for key, value in dftest[4].items():
       df_output['Critical Value (%s)' %key] = value
    print (df_output)
```

In [47]:

```
adf_test(TS_English)
```

```
Results of Dickey-Fuller Test:
Test Statistic                  -2.247284
p-value                          0.189534
#Lags Used                      14.000000
Number of Observations Used    535.000000
Critical Value (1%)             -3.442632
Critical Value (5%)             -2.866957
Critical Value (10%)            -2.569655
dtype: float64
```

In [48]:

```
Dickey_Fuller_test(TS_English)
```

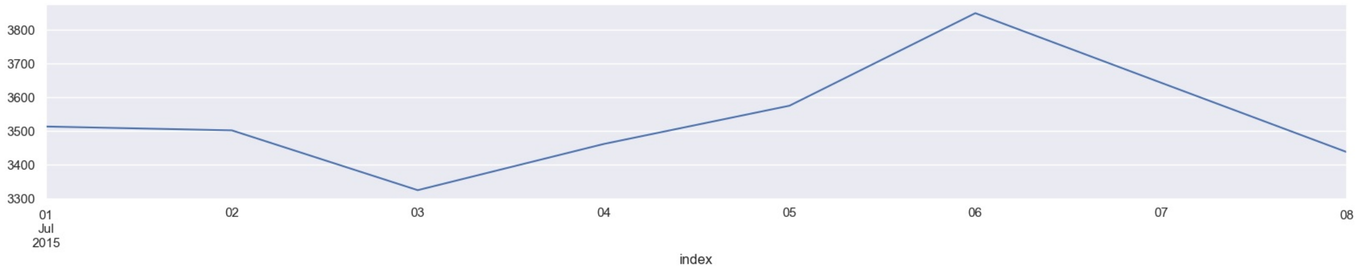
```
Time Series is NOT Stationary
P_value is:  0.18953359279992427
```

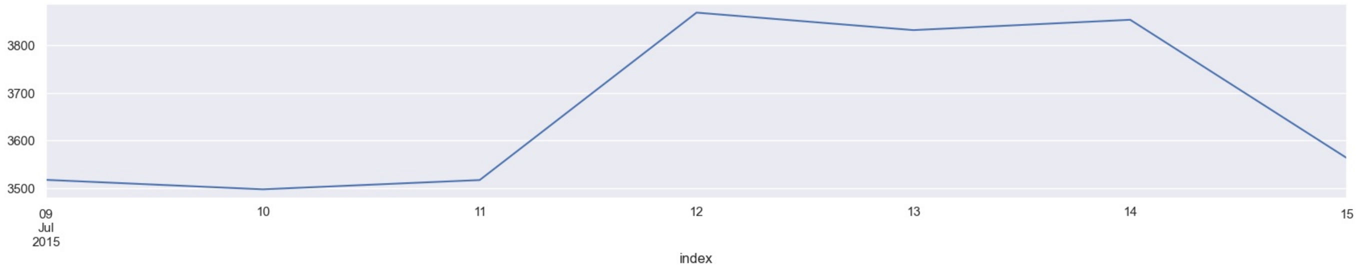
Visualising English-Language Page Visits Time Series manually to identify seasonality and period

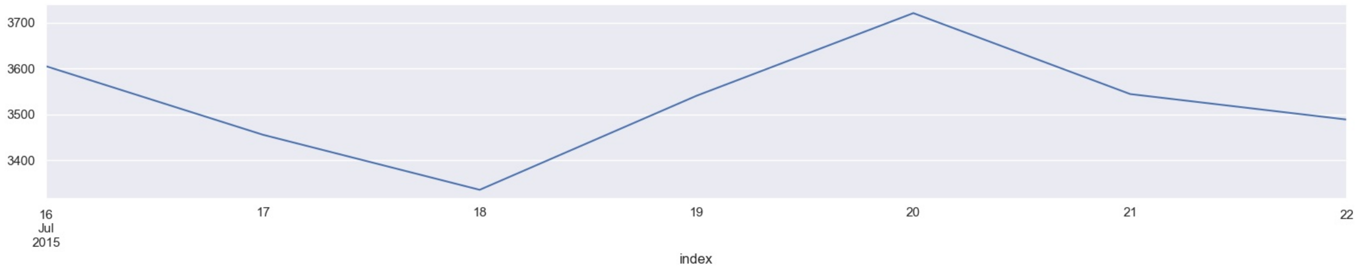
In [49]:

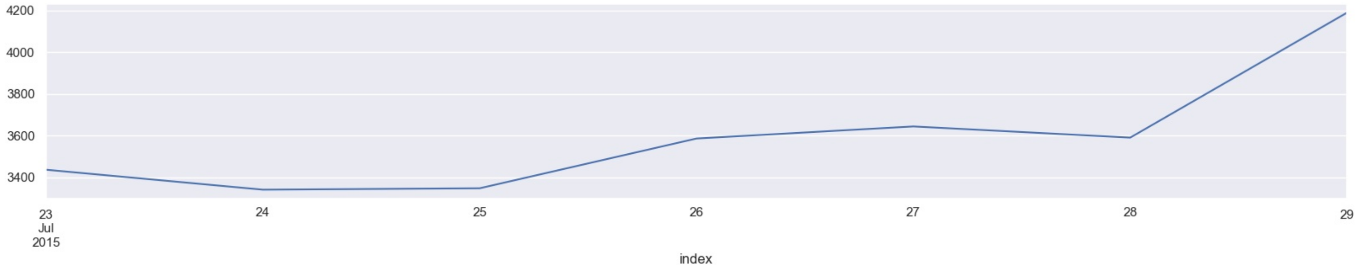
```
plt.rcParams['figure.figsize'] = (20, 3)

TS_English[:8].plot()
```

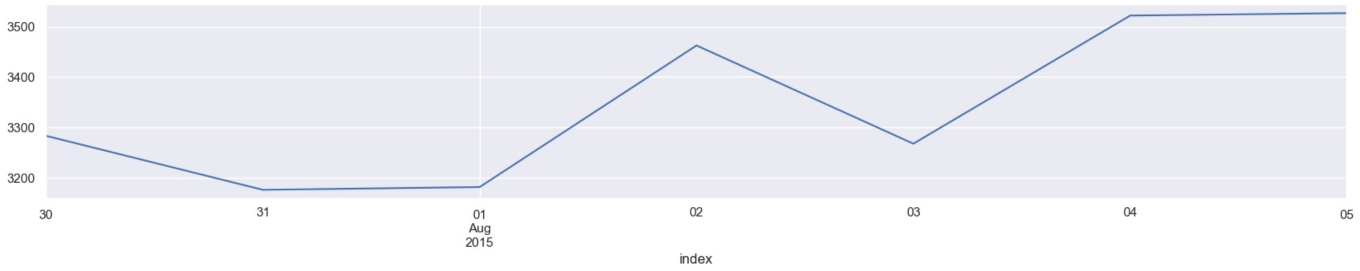
```
plt.show()
TS_English[8:15].plot()
plt.show()
TS_English[15:22].plot()
plt.show()
TS_English[22:29].plot()
plt.show()
TS_English[29:36].plot()
plt.show()

TS_English[36:44].plot()
plt.show()
```

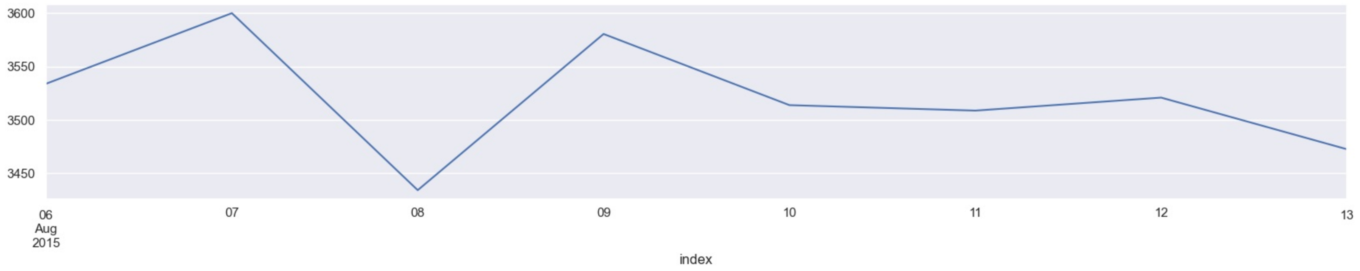












In [50]:

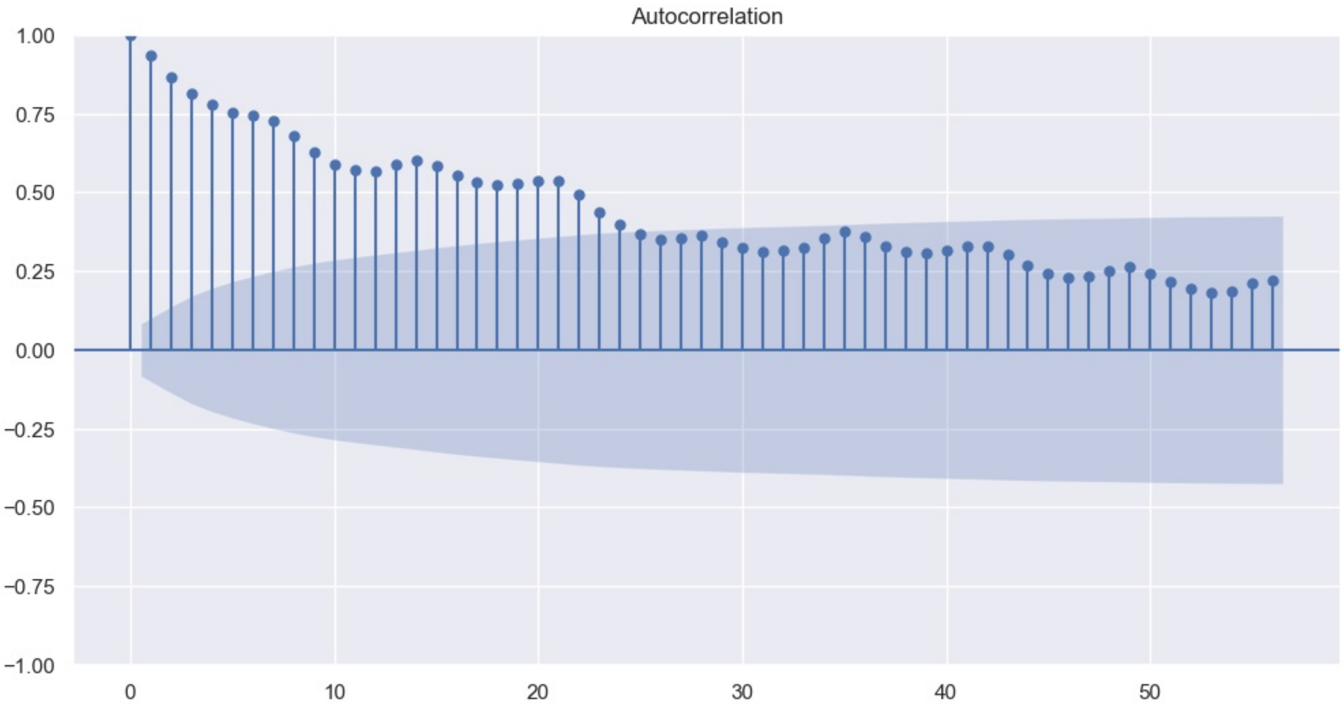
```
correlations = []
for lag in range(1,30):
    present = TS_English[:-lag]
    past = TS_English.shift(-lag)[:-lag]
    corrs = np.corrcoef(present,past)[0][-1]
    print(lag,corrs)
    correlations.append(corrs)
```

```
1 0.9363434527458436
2 0.8682966716039893
3 0.8185418037184543
4 0.7846718829500339
5 0.7612561076942569
6 0.7542260641783564
7 0.7386829287516696
8 0.6912638018189879
9 0.6370978014300408
10 0.6015277501876304
11 0.5825450402423569
12 0.5812931934793543
13 0.600726646281779
14 0.6142525351445116
15 0.5971084554755529
16 0.5693834937428246
17 0.5488401467532629
18 0.5377431132136109
19 0.54308167434112
20 0.5552694244923041
21 0.5540623423718064
22 0.5092655604869362
23 0.45373695576813594
24 0.41123362976203237
25 0.3816286061625173
26 0.36519963166994807
27 0.37236036273026013
28 0.37818226683160044
29 0.35939242667328164
```

Time Series Decomposition

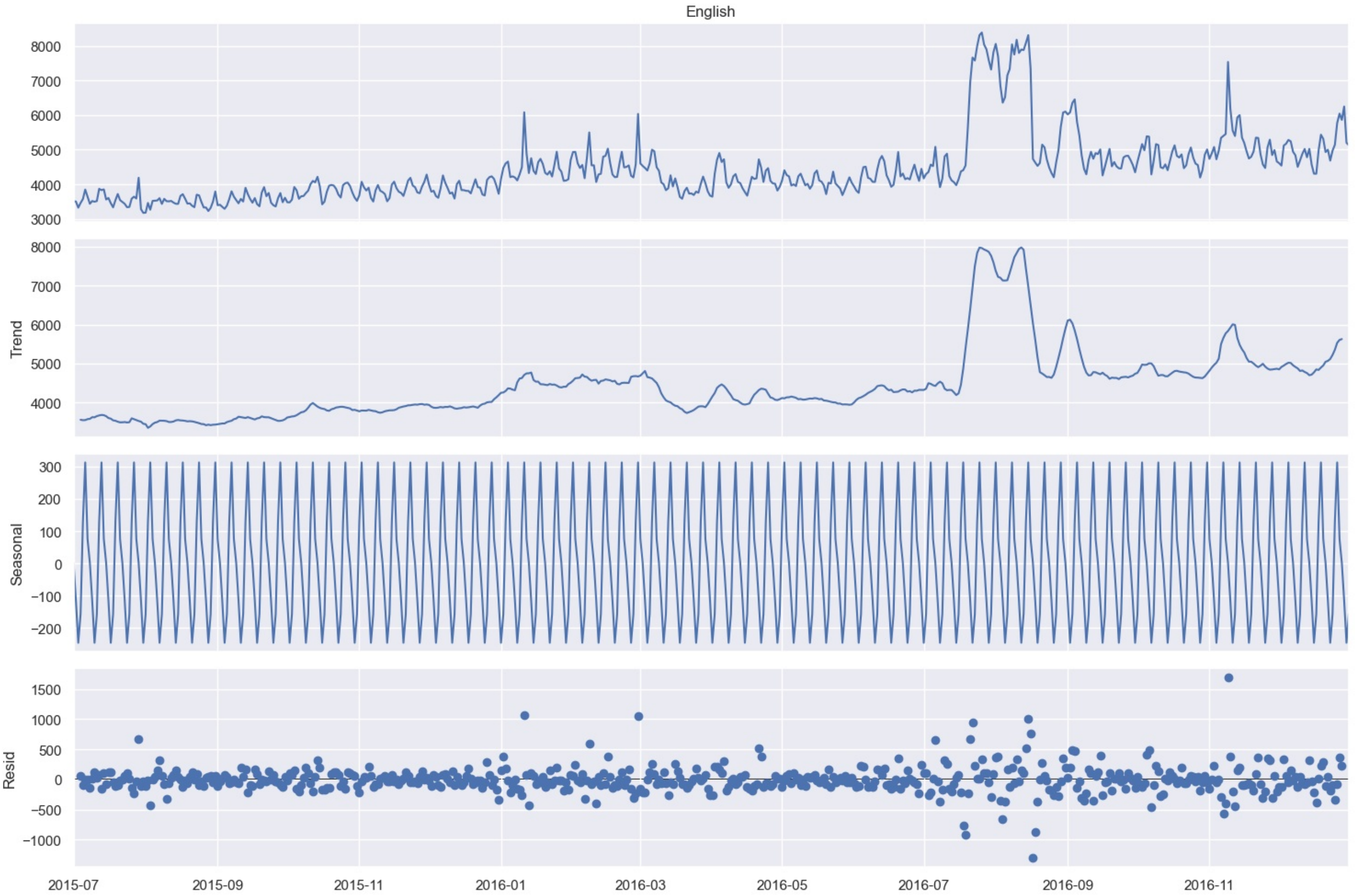
In [51]:

```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

plt.rcParams['figure.figsize'] = (12, 6)
plot_acf(TS_English,lags=56);
```



In [52]:

```
plt.rcParams['figure.figsize'] = (15, 10)

Decomposition_model = sm.tsa.seasonal_decompose(TS_English, model='additive',period=7)
Decomposition_model.plot();
```

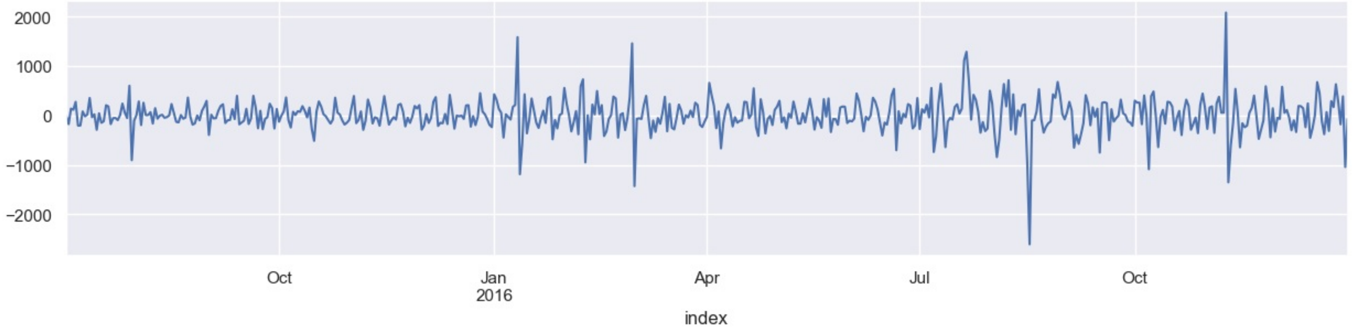


```
In [53]: Dickey_Fuller_test(pd.Series(Decomposition_model.resid).fillna(0))
```

```
Time Series is Stationary
P_value is:  3.727526947813056e-21
```

```
In [54]: plt.rcParams['figure.figsize'] = (15, 3)

         TS_English.diff(1).dropna().plot()
```

```
Out[54]: <Axes: xlabel='index'>
```



```
In [55]: Dickey_Fuller_test(TS_English.diff(1).dropna())
```

```
Time Series is Stationary
P_value is:  5.292474635436038e-13
```

```
In [56]: from sklearn.metrics import (
             mean_squared_error as mse,
             mean_absolute_error as mae,
             mean_absolute_percentage_error as mape
         )

         # Creating a function to print values of all these metrics.
         def performance(actual, predicted):
             print('MAE :', round(mae(actual, predicted), 3))
             print('RMSE :', round(mse(actual, predicted)**0.5, 3))
             print('MAPE:', round(mape(actual, predicted), 3))
```

Forecasting

Exponential Smoothing Method

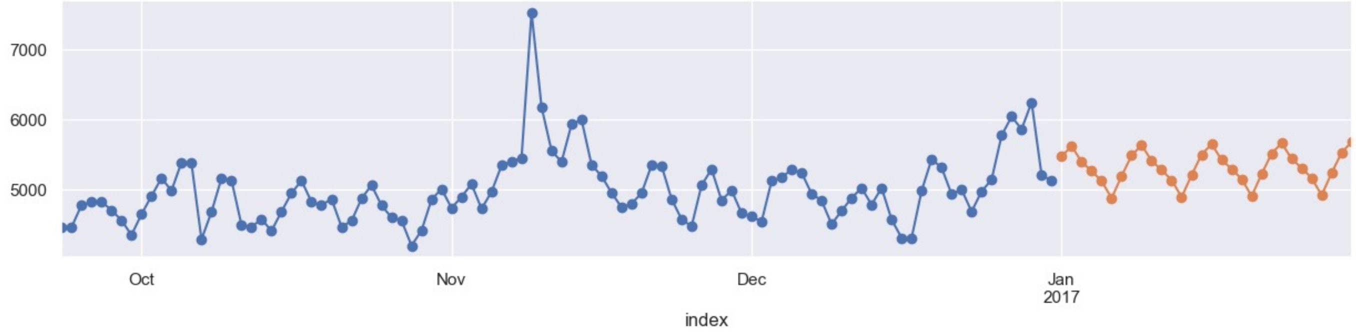
In [57]:

```
model = sm.tsa.ExponentialSmoothing(TS_English, seasonal='add',trend="add")
model = model.fit()


TS_English.tail(100).plot(style='-o', label='actual')
model.forecast(30).plot(style='-o', label='predicted')
```

```
C:\Users\senth\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency inf
ormation was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\senth\anaconda3\Lib\site-packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning: Optimiz
ation failed to converge. Check mle_retvals.
  warnings.warn(
```

Out[57]:

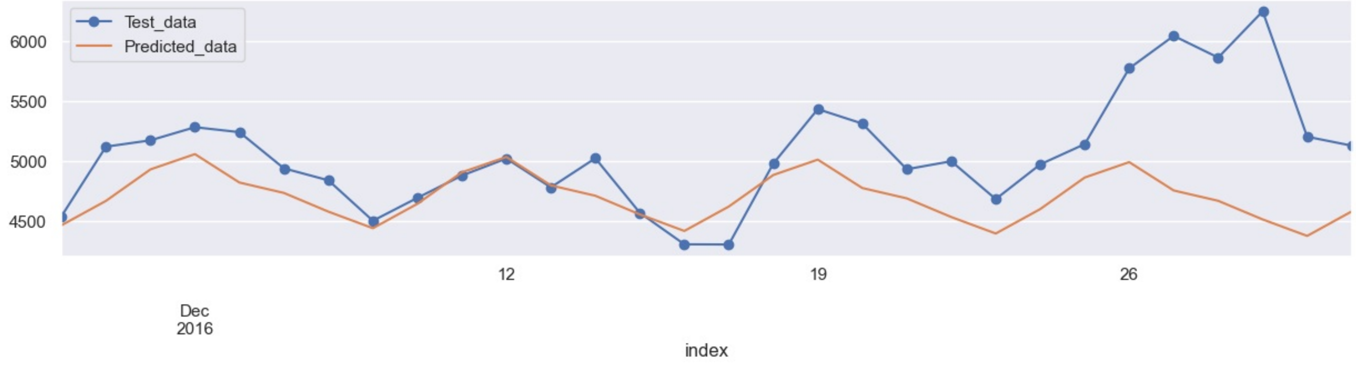
```
<Axes: xlabel='index'>
```



In [58]:

```
X_train = TS_English.loc[TS_English.index < TS_English.index[-30] ].copy()
X_test = TS_English.loc[TS_English.index >= TS_English.index[-30] ].copy()

import warnings # supress warnings
warnings.filterwarnings('ignore')


model = sm.tsa.ExponentialSmoothing(X_train,
                                    trend="add",
                                    damped_trend="add",
                                    seasonal="add")
model = model.fit(smoothing_level=None,     # alpha
            smoothing_trend=None,          # beta
            smoothing_seasonal=None)       # gama)

# X_test.plot()
Pred = model.forecast(steps=30)
performance(X_test,Pred)

X_test.plot(style="-o",label ="Test_data")
Pred.plot(label="Predicted_data")
plt.legend()
plt.show()
```

```
MAE : 394.978
RMSE : 563.352
MAPE: 0.073
```



ARIMA

In [59]:

```
from statsmodels.tsa.arima.model import ARIMA
```

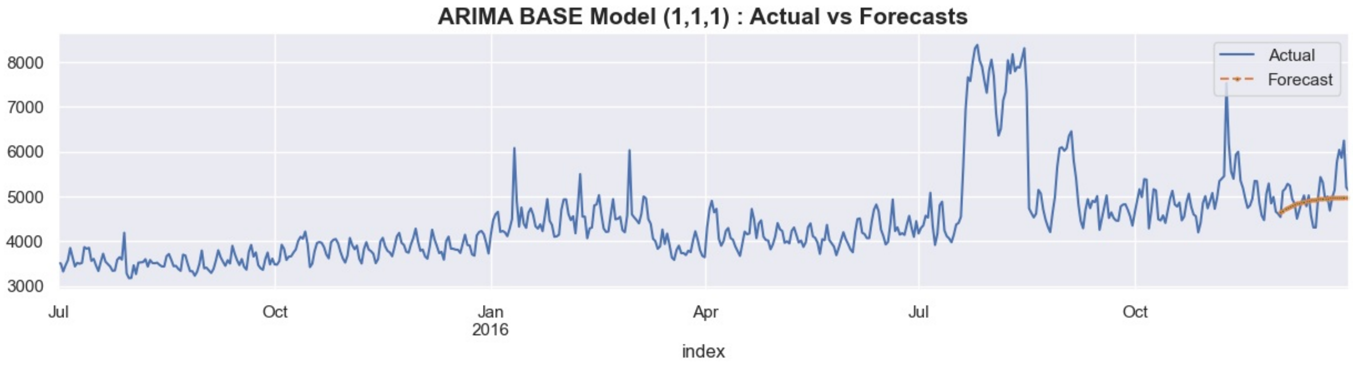
In [60]:

```
TS = TS_English.copy(deep=True)
```

In [61]:

```
n_forecast = 30
```

```
model = ARIMA(TS[:-n_forecast],
              order = (1,1,1))
model = model.fit()

predicted = model.forecast(steps= n_forecast, alpha = 0.05)


TS.plot(label = 'Actual')
predicted.plot(label = 'Forecast', linestyle='dashed', marker='o',markerfacecolor='green', markersize=2)
plt.legend(loc="upper right")
plt.title('ARIMA BASE Model (1,1,1) : Actual vs Forecasts', fontsize = 15, fontweight = 'bold')
plt.show()


#Calculating MAPE & RMSE
actuals = TS.values[-n_forecast:]
errors = TS.values[-n_forecast:] - predicted.values

mape = np.mean(np.abs(errors)/ np.abs(actuals))
rmse = np.sqrt(np.mean(errors**2))

print()
print(f'MAPE of Model : {np.round(mape,5)}')

print(f'RMSE of Model : {np.round(rmse,3)}')
```



```
MAPE of Model : 0.06585
RMSE of Model : 472.186
```

SARIMAX model

In [62]:

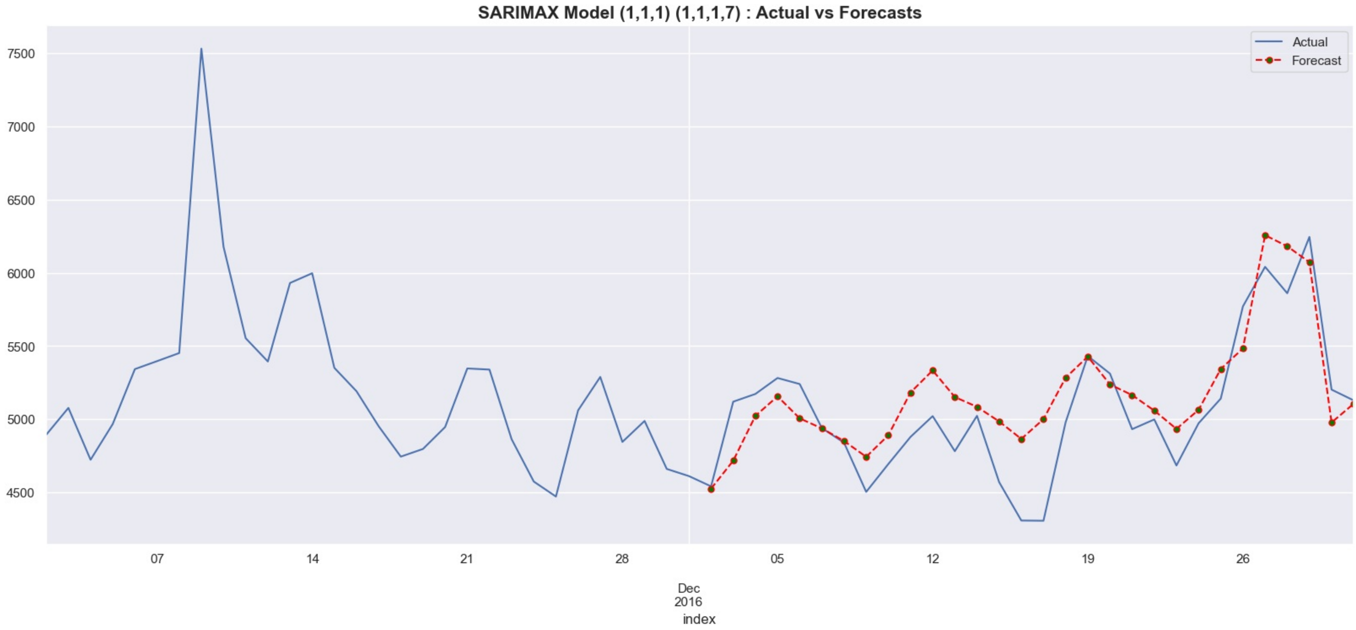
```
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

In [63]:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

def sarimax_model(time_series, n, p=0, d=0, q=0, P=0, D=0, Q=0, s=0, exog = []):

    #Creating SARIMAX Model with order(p,d,q) & seasonal_order=(P, D, Q, s)
    model = SARIMAX(time_series[:-n], \
                    order =(p,d,q),
                    seasonal_order=(P, D, Q, s),
                    exog = exog[:-n],
                    initialization='approximate_diffuse')
    model_fit = model.fit()

    #Creating forecast for last n-values
    model_forecast = model_fit.forecast(n, dynamic = True, exog = pd.DataFrame(exog[-n:]))

    #plotting Actual & Forecasted values

    plt.figure(figsize = (20,8))
    time_series[-60:].plot(label = 'Actual')
    model_forecast[-60:].plot(label = 'Forecast', color = 'red',
                              linestyle='dashed', marker='o',markerfacecolor='green', markersize=5)
    plt.legend(loc="upper right")
    plt.title(f'SARIMAX Model ({p},{d},{q}) ({P},{D},{Q},{s}) : Actual vs Forecasts', fontsize = 15, fontweight
    plt.show()

    #Calculating MAPE & RMSE
    actuals = time_series.values[-n:]
    errors = time_series.values[-n:] - model_forecast.values

    mape = np.mean(np.abs(errors)/ np.abs(actuals))
    rmse = np.sqrt(np.mean(errors**2))

    print()
    print(f'MAPE of Model : {np.round(mape,5)}')
    print(f'RMSE of Model : {np.round(rmse,3)}')
```

In [64]:

```
exog = Exog_Campaign_eng['Exog'].to_numpy()
time_series = aggregated_data.English
test_size= 0.1
p,d,q, P,D,Q,s = 1,1,1,1,1,1,7
n = 30
sarimax_model(time_series, n, p=p, d=d, q=q, P=P, D=D, Q=Q, s=s, exog = exog)
```



```
MAPE of Model : 0.04449
RMSE of Model : 272.497
```

Hyperparameter tuning for SARIMAX model

In [65]:

```
def SARIMAX_grid_search(time_series, n, param, d_param, s_param, exog = []):
    counter = 0
    #creating df for storing results summary
    param_df = pd.DataFrame(columns = ['serial','pdq', 'PDQs', 'mape', 'rmse'])

    #Creating loop for every paramater to fit SARIMAX model
    for p in param:
        for d in d_param:
            for q in param:
                for P in param:
                    for D in d_param:
                        for Q in param:
                            for s in s_param:
                                #Creating Model
                                model = SARIMAX(time_series[:-n],
                                                order=(p,d,q),
                                                seasonal_order=(P, D, Q, s),
                                                exog = exog[:-n],
                                                initialization='approximate_diffuse')
                                model_fit = model.fit()

                                #Creating forecast from Model
                                model_forecast = model_fit.forecast(n, dynamic = True, exog = pd.DataFrame(exog

                                #Calculating errors for results
                                actuals = time_series.values[-n:]
                                errors = time_series.values[-n:] - model_forecast.values

                                #Calculating MAPE & RMSE
                                mape = np.mean(np.abs(errors)/ np.abs(actuals))
                                rmse = np.sqrt(np.mean(errors**2))
                                mape = np.round(mape,5)
                                rmse = np.round(rmse,3)

                                #Storing the results in param_df
                                counter += 1
                                list_row = [counter, (p,d,q), (P,D,Q,s), mape, rmse]
                                param_df.loc[len(param_df)] = list_row

                        #print statement to check progress of Loop
                        print(f'Possible Combination: {counter} out of { (len(param)**4)*len(s_param)*(len(d_param)**2)}

    return param_df
```

In [66]:

```
exog = Exog_Campaign_eng['Exog'].to_numpy()
time_series = aggregated_data.English
n = 30
param = [0,1,2]
d_param = [0,1]
s_param = [7]

english_params  = SARIMAX_grid_search(time_series, n, param, d_param,s_param, exog)
```

```
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
```

In [67]:

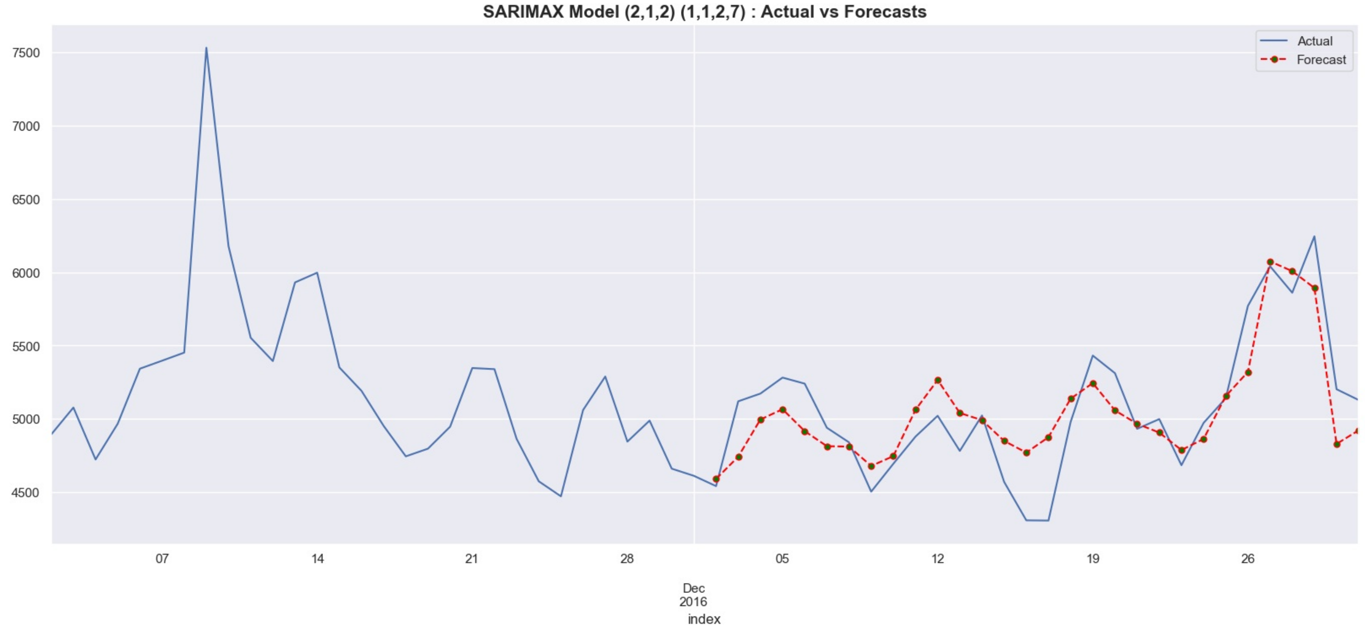
```
english_params.sort_values(['mape', 'rmse']).head()
```

Out[67]:

	serial	pdq	PDQs	mape	rmse
209	210	(1, 1, 2)	(1, 1, 2, 7)	0.04014	242.824
317	318	(2, 1, 2)	(1, 1, 2, 7)	0.04045	247.862
323	324	(2, 1, 2)	(2, 1, 2, 7)	0.04127	252.235
40	41	(0, 0, 2)	(0, 1, 1, 7)	0.04199	276.311
41	42	(0, 0, 2)	(0, 1, 2, 7)	0.04206	271.577

In [68]:

```
exog = Exog_Campaign_eng['Exog'].to_numpy()
time_series = aggregated_data.English
test_size= 0.1
p,d,q, P,D,Q,s = 2,1,2,1,1,2,7
n = 30
sarimax_model(time_series, n, p=p, d=d, q=q, P=P, D=D, Q=Q, s=s, exog = exog)
```



```
MAPE of Model : 0.04045
RMSE of Model : 247.862
```

Hyperparameter tuning for all other languages

In [69]:

```
def pipeline_sarimax_grid_search_without_exog(languages, data, n, param, d_param, s_param):

    best_param_df  = pd.DataFrame(columns = ['language','p','d', 'q', 'P','D','Q','s','mape'])
    for lang in languages:
        print('')
        print('')
        print(f'--------------------------------------------------------------')
        print(f'          Finding best parameters for {lang}                   ')
        print(f'--------------------------------------------------------------')
        counter = 0
        time_series = data[lang]
        best_mape = 100

        #Creating loop for every paramater to fit SARIMAX model
        for p in param:
            for d in d_param:
                for q in param:
                    for P in param:
                        for D in d_param:
                            for Q in param:
                                for s in s_param:
                                    #Creating Model
                                    model = SARIMAX(time_series[:-n],
                                                    order=(p,d,q),
                                                    seasonal_order=(P, D, Q, s),
                                                    initialization='approximate_diffuse')
                                    model_fit = model.fit()

                                    #Creating forecast from Model
                                    model_forecast = model_fit.forecast(n, dynamic = True)

                                    #Calculating errors for results
                                    actuals = time_series.values[-n:]
                                    errors = time_series.values[-n:] - model_forecast.values

                                    #Calculating MAPE & RMSE
                                    mape = np.mean(np.abs(errors)/ np.abs(actuals))

                                    counter += 1

                                    if (mape < best_mape):
                                        best_mape = mape
                                        best_p = p
                                        best_d = d
                                        best_q = q
                                        best_P = P
                                        best_D = D
                                        best_Q = Q
                                        best_s = s
                                    else: pass

                    #print statement to check progress of Loop
                    print(f'Possible Combination: {counter} out of {(len(param)**4)*len(s_param)*(len(d_param)**

        best_mape = np.round(best_mape, 5)
        print(f'--------------------------------------------------------------')
        print(f'Minimum MAPE for {lang} = {best_mape}')
        print(f'Corresponding Best Parameters are {best_p , best_d, best_q, best_P, best_D, best_Q, best_s}')
        print(f'--------------------------------------------------------------')

        best_param_row = [lang, best_p, best_d, best_q, best_P, best_D, best_Q, best_s, best_mape]
        best_param_df.loc[len(best_param_df)] = best_param_row

    return best_param_df
```

In [70]:

```
languages = aggregated_data.columns
n = 30
param = [0,1,2]
d_param = [0,1]
s_param = [7]


best_param_df = pipeline_sarimax_grid_search_without_exog(languages, aggregated_data, n, param, d_param, s_para
```

```
--------------------------------------------------------------
          Finding best parameters for Chinese
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for Chinese = 0.03074
Corresponding Best Parameters are (0, 1, 0, 1, 0, 2, 7)
--------------------------------------------------------------


--------------------------------------------------------------
          Finding best parameters for English
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for English = 0.05264
Corresponding Best Parameters are (2, 0, 1, 0, 1, 2, 7)
--------------------------------------------------------------


--------------------------------------------------------------
          Finding best parameters for French
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for French = 0.06362
Corresponding Best Parameters are (0, 0, 2, 2, 1, 2, 7)
--------------------------------------------------------------


--------------------------------------------------------------
          Finding best parameters for German
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for German = 0.06578
Corresponding Best Parameters are (0, 1, 1, 1, 0, 1, 7)
--------------------------------------------------------------


--------------------------------------------------------------
          Finding best parameters for Japenese
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for Japenese = 0.07122
Corresponding Best Parameters are (0, 1, 2, 2, 1, 0, 7)
--------------------------------------------------------------


--------------------------------------------------------------
          Finding best parameters for Russian
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated
Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for Russian = 0.0458
Corresponding Best Parameters are (0, 0, 2, 1, 0, 2, 7)
--------------------------------------------------------------


--------------------------------------------------------------
          Finding best parameters for Spanish
--------------------------------------------------------------
Possible Combination: 18 out of 324 calculated
Possible Combination: 36 out of 324 calculated
Possible Combination: 54 out of 324 calculated
Possible Combination: 72 out of 324 calculated
Possible Combination: 90 out of 324 calculated
Possible Combination: 108 out of 324 calculated
Possible Combination: 126 out of 324 calculated
Possible Combination: 144 out of 324 calculated
Possible Combination: 162 out of 324 calculated
Possible Combination: 180 out of 324 calculated
Possible Combination: 198 out of 324 calculated
Possible Combination: 216 out of 324 calculated
Possible Combination: 234 out of 324 calculated
Possible Combination: 252 out of 324 calculated

Possible Combination: 270 out of 324 calculated
Possible Combination: 288 out of 324 calculated
Possible Combination: 306 out of 324 calculated
Possible Combination: 324 out of 324 calculated
--------------------------------------------------------------
Minimum MAPE for Spanish = 0.08561
Corresponding Best Parameters are (0, 1, 0, 2, 1, 0, 7)
--------------------------------------------------------------
```

In [71]:

```
best_param_df.sort_values(['mape'], inplace = True)
best_param_df
```

Out[71]:

	language	p	d	q	P	D	Q	s	mape
0	Chinese	0	1	0	1	0	2	7	0.03074
5	Russian	0	0	2	1	0	2	7	0.04580
1	English	2	0	1	0	1	2	7	0.05264
2	French	0	0	2	2	1	2	7	0.06362
3	German	0	1	1	1	0	1	7	0.06578
4	Japenese	0	1	2	2	1	0	7	0.07122
6	Spanish	0	1	0	2	1	0	7	0.08561

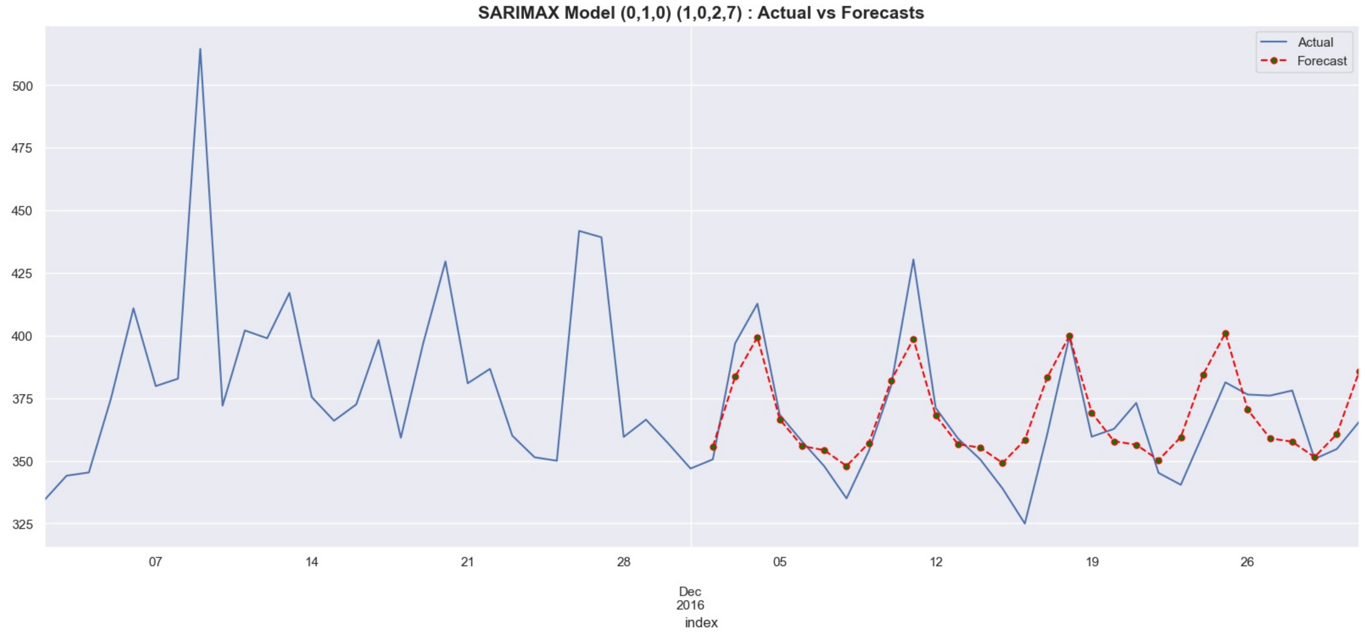
In [72]:

```
def plot_best_SARIMAX_model(languages, data, n, best_param_df):

    for lang in languages:
        #fetching respective best parameters for that language
        p = best_param_df.loc[best_param_df['language'] == lang, ['p']].values[0][0]
        d = best_param_df.loc[best_param_df['language'] == lang, ['d']].values[0][0]
        q = best_param_df.loc[best_param_df['language'] == lang, ['q']].values[0][0]
        P = best_param_df.loc[best_param_df['language'] == lang, ['P']].values[0][0]
        D = best_param_df.loc[best_param_df['language'] == lang, ['D']].values[0][0]
        Q = best_param_df.loc[best_param_df['language'] == lang, ['Q']].values[0][0]
        s = best_param_df.loc[best_param_df['language'] == lang, ['s']].values[0][0]

        #Creating language time-series
        time_series = data[lang]

        #Creating SARIMAX Model with order(p,d,q) & seasonal_order=(P, D, Q, s)
        model = SARIMAX(time_series[:-n],
                        order =(p,d,q),
                        seasonal_order=(P, D, Q, s),
                        initialization='approximate_diffuse')
        model_fit = model.fit()

        #Creating forecast for last n-values
        model_forecast = model_fit.forecast(n, dynamic = True)

        #Calculating MAPE & RMSE
        actuals = time_series.values[-n:]
        errors = time_series.values[-n:] - model_forecast.values

        mape = np.mean(np.abs(errors)/ np.abs(actuals))
        rmse = np.sqrt(np.mean(errors**2))

        print('')
        print('')
        print(f'------------------------------------------------------------------------------------------')
        print(f'        SARIMAX model for {lang} Time Series                                              ')
        print(f'        Parameters of Model : ({p},{d},{q}) ({P},{D},{Q},{s})                            ')
        print(f'        MAPE of Model       : {np.round(mape,5)}                                          ')
        print(f'        RMSE of Model       : {np.round(rmse,3)}                                          ')
        print(f'------------------------------------------------------------------------------------------')

        #plotting Actual & Forecasted values
        time_series.index = time_series.index.astype('datetime64[ns]')
        model_forecast.index = model_forecast.index.astype('datetime64[ns]')
        plt.figure(figsize = (20,8))
        time_series[-60:].plot(label = 'Actual')
        model_forecast[-60:].plot(label = 'Forecast', color = 'red',
                                  linestyle='dashed', marker='o',markerfacecolor='green', markersize=5)
        plt.legend(loc="upper right")
        plt.title(f'SARIMAX Model ({p},{d},{q}) ({P},{D},{Q},{s}) : Actual vs Forecasts', fontsize = 15, fontwe
        plt.show()
    return 0
```

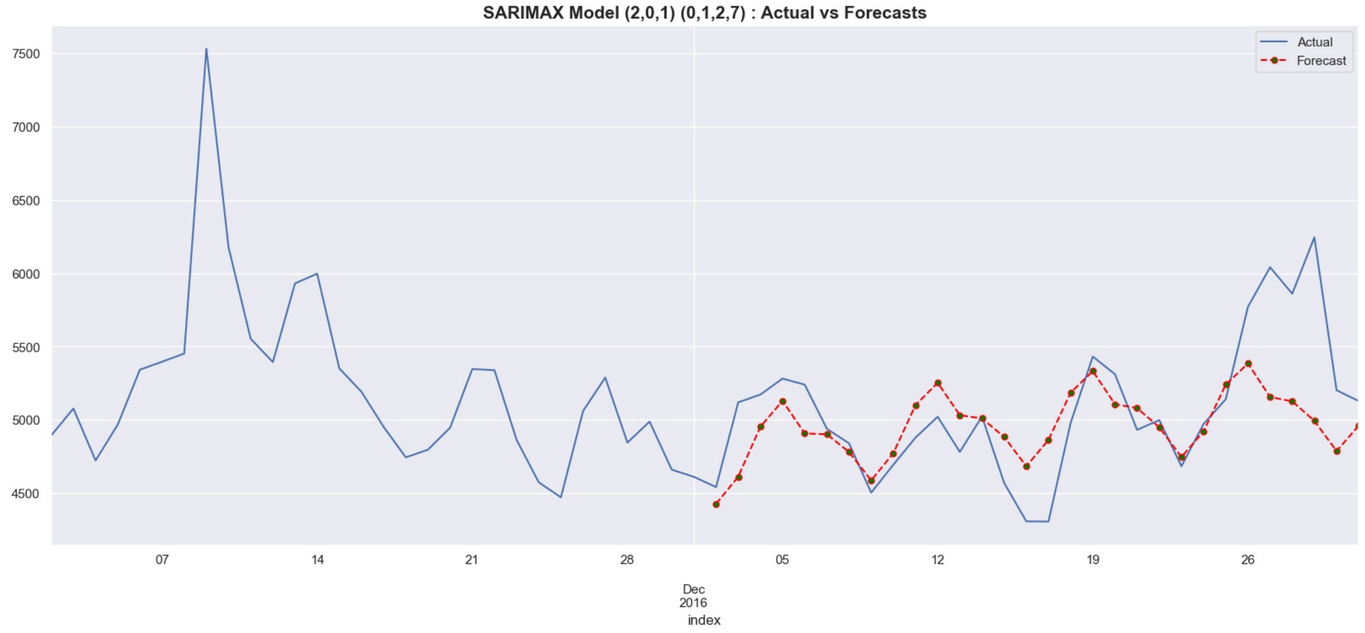
In [73]:

```
#Plotting SARIMAX model for each Language Time Series
languages = aggregated_data.columns
n = 30
plot_best_SARIMAX_model(languages, aggregated_data, n, best_param_df)
```

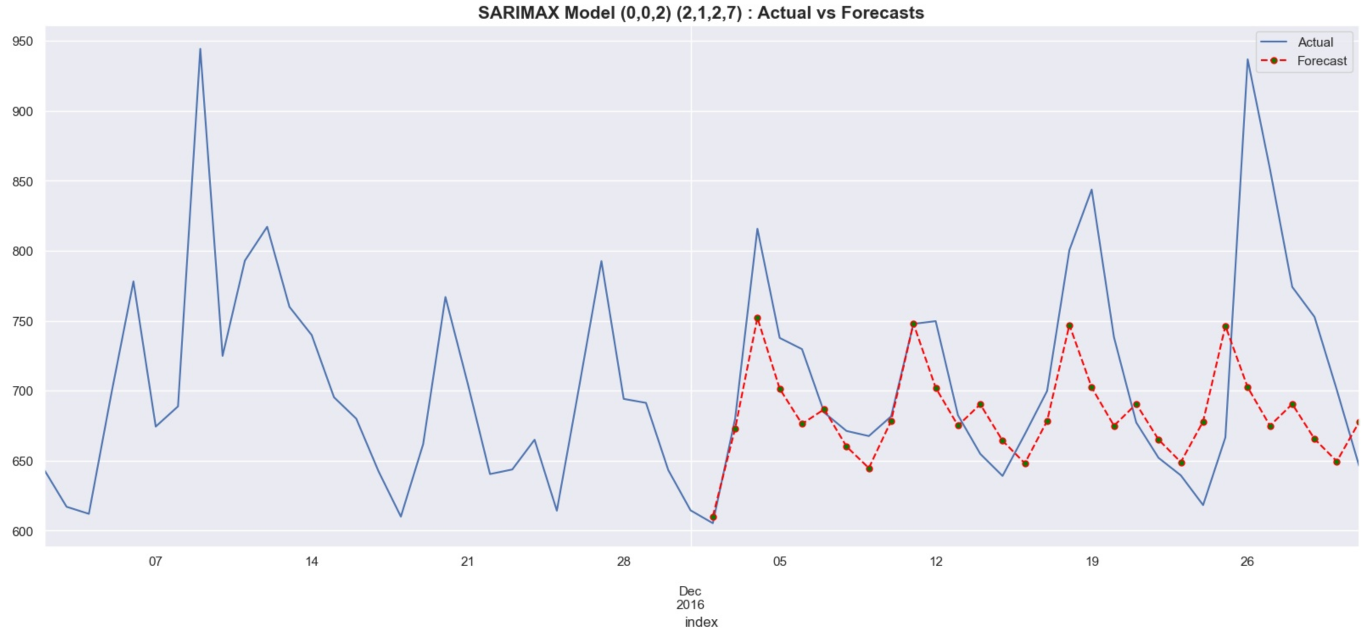
```
--------------------------------------------------------------------------------
        SARIMAX model for Chinese Time Series
        Parameters of Model : (0,1,0) (1,0,2,7)
        MAPE of Model       : 0.03074
        RMSE of Model       : 14.487
--------------------------------------------------------------------------------
```


SARIMAX Model (0,1,0) (1,0,2,7) : Actual vs Forecasts

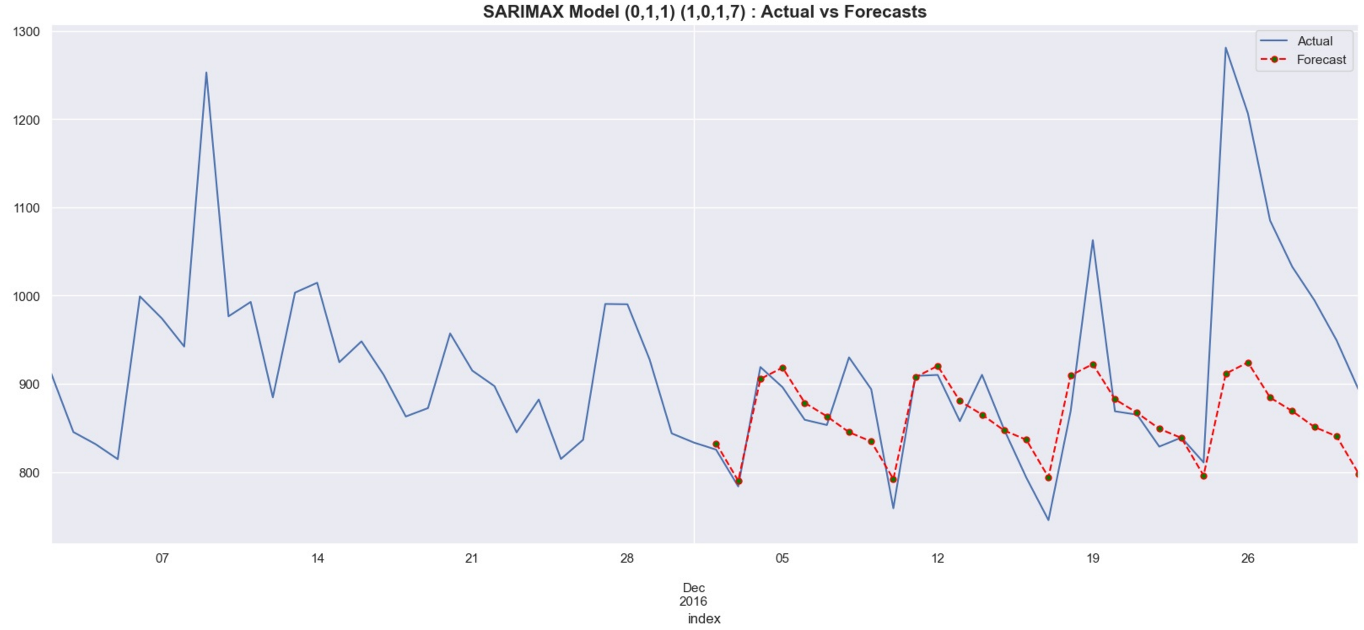
```
--------------------------------------------------------------------------------
        SARIMAX model for English Time Series
        Parameters of Model : (2,0,1) (0,1,2,7)
        MAPE of Model       : 0.05264
        RMSE of Model       : 390.016
--------------------------------------------------------------------------------
```


SARIMAX Model (2,0,1) (0,1,2,7) : Actual vs Forecasts

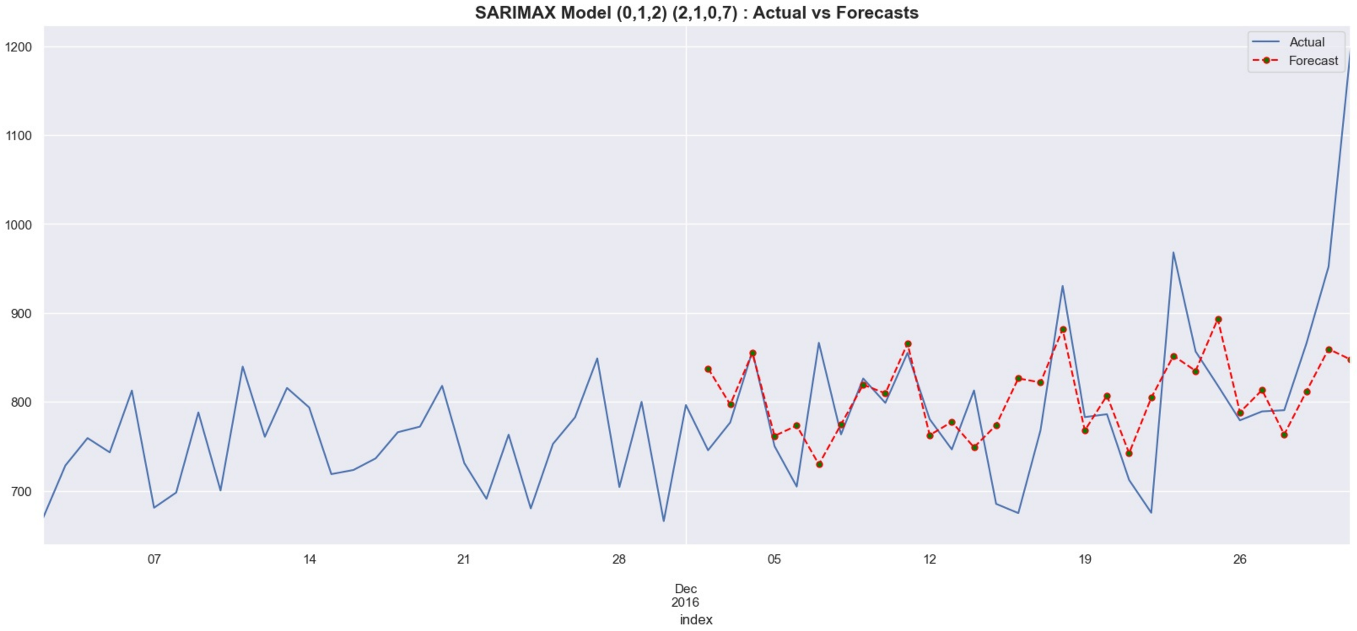
```
--------------------------------------------------------------------------------
        SARIMAX model for French Time Series
        Parameters of Model : (0,0,2) (2,1,2,7)
        MAPE of Model       : 0.06362
        RMSE of Model       : 72.605
--------------------------------------------------------------------------------
```



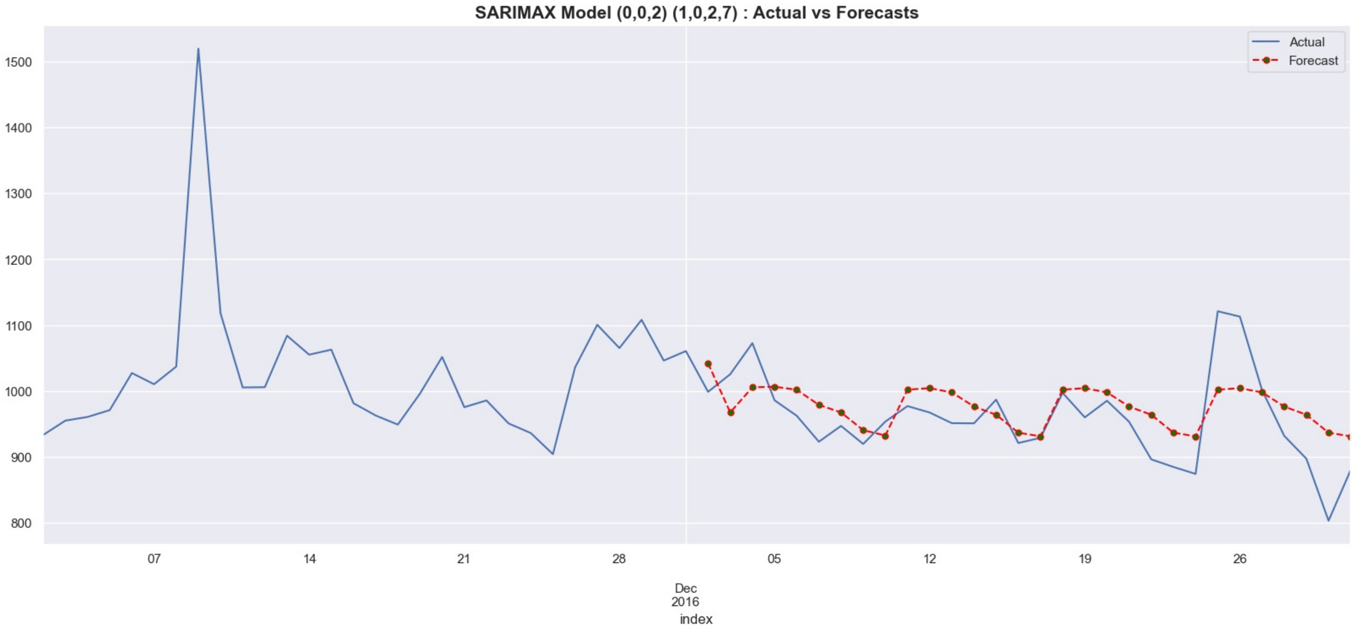
```
-------------------------------------------------------------------------------------
       SARIMAX model for German Time Series
       Parameters of Model : (0,1,1) (1,0,1,7)
       MAPE of Model       : 0.06578
       RMSE of Model       : 110.617
-------------------------------------------------------------------------------------
```



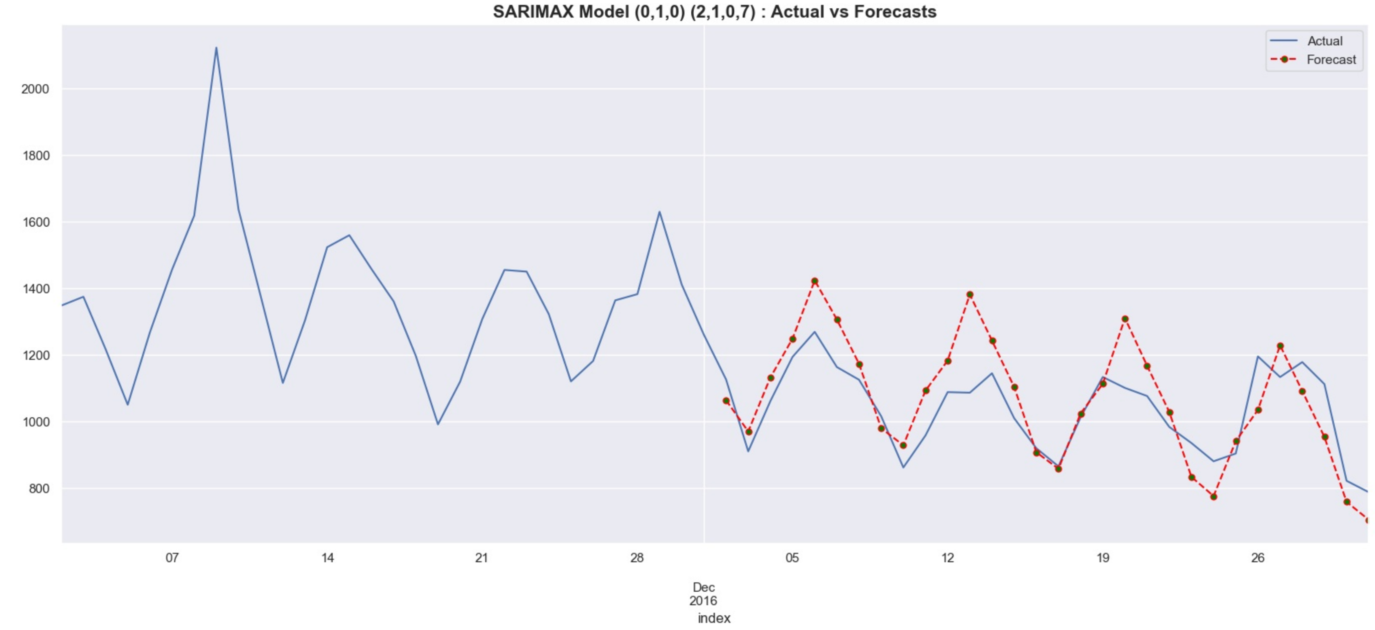
```
-------------------------------------------------------------------------------------
       SARIMAX model for Japenese Time Series
       Parameters of Model : (0,1,2) (2,1,0,7)
       MAPE of Model       : 0.07122
       RMSE of Model       : 90.833
-------------------------------------------------------------------------------------
```



```
----------------------------------------------------------------------------------------
        SARIMAX model for Russian Time Series
        Parameters of Model : (0,0,2) (1,0,2,7)
        MAPE of Model       : 0.0458
        RMSE of Model       : 54.07
----------------------------------------------------------------------------------------
```



```
----------------------------------------------------------------------------------------
        SARIMAX model for Spanish Time Series
        Parameters of Model : (0,1,0) (2,1,0,7)
        MAPE of Model       : 0.08561
        RMSE of Model       : 109.03
----------------------------------------------------------------------------------------
```



Out[73]: 0

Forecasting using Facebook Prophet

In [74]:

```
!pip install numpy==1.22.4
```

```
Collecting numpy==1.22.4
  Using cached numpy-1.22.4.zip (11.5 MB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Building wheels for collected packages: numpy
  Building wheel for numpy (pyproject.toml): started
  Building wheel for numpy (pyproject.toml): finished with status 'error'
Failed to build numpy
```

```
  error: subprocess-exited-with-error

  Building wheel for numpy (pyproject.toml) did not run successfully.
  exit code: 1

  [227 lines of output]
  setup.py:66: RuntimeWarning: NumPy 1.22.4 may not yet support Python 3.11.
    warnings.warn(
  Running from numpy source directory.
  Processing numpy/random\_bounded_integers.pxd.in
  Processing numpy/random\bit_generator.pyx
  Processing numpy/random\mtrand.pyx
  Processing numpy/random\_bounded_integers.pyx.in
  Processing numpy/random\_common.pyx
  Processing numpy/random\_generator.pyx
  Processing numpy/random\_mt19937.pyx
  Processing numpy/random\_pcg64.pyx
  Processing numpy/random\_philox.pyx
  Processing numpy/random\_sfc64.pyx
  Cythonizing sources
  INFO: blas_opt_info:
  INFO: blas_armpl_info:
  INFO: No module named 'numpy.distutils._msvccompiler' in numpy.distutils; trying from distutils
  INFO: customize MSVCCompiler
  INFO:   libraries armpl_lp64_mp not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\a
naconda3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: blas_mkl_info:
  INFO:   libraries mkl_rt not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda
3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: blis_info:
  INFO:   libraries blis not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda3\
\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: openblas_info:
  INFO:   libraries openblas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anacon
da3\\libs']
  INFO: get_default_fcompiler: matching types: '['gnu', 'intelv', 'absoft', 'compaqv', 'intelev', 'gnu95', 'g95'
, 'intelvem', 'intelem', 'flang']'
  INFO: customize GnuFCompiler
  WARN: Could not locate executable g77
  WARN: Could not locate executable f77
  INFO: customize IntelVisualFCompiler
  WARN: Could not locate executable ifort
  WARN: Could not locate executable ifl
  INFO: customize AbsoftFCompiler
  WARN: Could not locate executable f90
  INFO: customize CompaqVisualFCompiler
  INFO: Found executable C:\Users\senth\anaconda3\Library\usr\bin\DF.exe
  INFO: customize IntelItaniumVisualFCompiler
  WARN: Could not locate executable efl
  INFO: customize Gnu95FCompiler
  INFO: Found executable C:\Users\senth\anaconda3\Library\mingw-w64\bin\gfortran.exe
  Using built-in specs.
  COLLECT_GCC=C:\Users\senth\anaconda3\Library\mingw-w64\bin\gfortran.exe
  COLLECT_LTO_WRAPPER=C:/Users/senth/anaconda3/Library/mingw-w64/bin/../lib/gcc/x86_64-w64-mingw32/5.3.0/lto-wra
pper.exe
  Target: x86_64-w64-mingw32
  Configured with: ../gcc-5.3.0/configure --prefix=/mingw64 --with-local-prefix=/mingw64/local --build=x86_64-w6
4-mingw32 --host=x86_64-w64-mingw32 --target=x86_64-w64-mingw32 --with-native-system-header-dir=/mingw64/x86_64-
w64-mingw32/include --libexecdir=/mingw64/lib --with-gxx-include-dir=/mingw64/include/c++/5.3.0 --enable-bootstr
ap --with-arch=x86-64 --with-tune=generic --enable-languages=c,lto,c++,objc,obj-c++,fortran,ada --enable-shared
--enable-static --enable-libatomic --enable-threads=posix --enable-graphite --enable-fully-dynamic-string --enab
le-libstdcxx-time=yes --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-version-specific-runtime-libs -
-disable-isl-version-check --enable-lto --enable-libgomp --disable-multilib --enable-checking=release --disable-
rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-libiconv --with-system-zl
ib --with-gmp=/mingw64 --with-mpfr=/mingw64 --with-mpc=/mingw64 --with-isl=/mingw64 --with-pkgversion='Rev5, Bui
lt by MSYS2 project' --with-bugurl=https://sourceforge.net/projects/msys2 --with-gnu-as --with-gnu-ld
  Thread model: posix
  gcc version 5.3.0 (Rev5, Built by MSYS2 project)
  INFO:   NOT AVAILABLE
  INFO:
  INFO: accelerate_info:
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_3_10_blas_threads_info:
  INFO: Setting PTATLAS=ATLAS
  INFO:   libraries tatlas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda
3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_3_10_blas_info:
  INFO:   libraries satlas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda
3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_blas_threads_info:
  INFO: Setting PTATLAS=ATLAS
  INFO:   libraries ptf77blas,ptcblas,atlas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users
\\senth\\anaconda3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_blas_info:
  INFO:   libraries f77blas,cblas,atlas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\se
nth\\anaconda3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  C:\Users\senth\AppData\Local\Temp\pip-install-89qbc1u6\numpy_0e0e205ce6734dd1ada168290df6f5e0\numpy\distutils\
system_info.py:2077: UserWarning:
      Optimized (vendor) Blas libraries are not found.
      Falls back to netlib Blas library which has worse performance.
      A better performance should be easily gained by switching
      Blas library.
    if self._calc_info(blas):
  INFO: blas_info:
  INFO:   libraries blas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda3\
\libs']
  INFO:   NOT AVAILABLE
  INFO:
  C:\Users\senth\AppData\Local\Temp\pip-install-89qbc1u6\numpy_0e0e205ce6734dd1ada168290df6f5e0\numpy\distutils\
system_info.py:2077: UserWarning:
      Blas (http://www.netlib.org/blas/) libraries not found.
      Directories to search for the libraries can be specified in the
      numpy/distutils/site.cfg file (section [blas]) or by setting
      the BLAS environment variable.
    if self._calc_info(blas):
  INFO: blas_src_info:
  INFO:   NOT AVAILABLE
  INFO:
  C:\Users\senth\AppData\Local\Temp\pip-install-89qbc1u6\numpy_0e0e205ce6734dd1ada168290df6f5e0\numpy\distutils\
system_info.py:2077: UserWarning:
      Blas (http://www.netlib.org/blas/) sources not found.
      Directories to search for the sources can be specified in the
      numpy/distutils/site.cfg file (section [blas_src]) or by setting
      the BLAS_SRC environment variable.
    if self._calc_info(blas):
  INFO:   NOT AVAILABLE
  INFO:
  non-existing path in 'numpy\\distutils': 'site.cfg'
  INFO: lapack_opt_info:
  INFO: lapack_armpl_info:
  INFO:   libraries armpl_lp64_mp not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\a
naconda3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: lapack_mkl_info:
  INFO:   libraries mkl_rt not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda
3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: openblas_lapack_info:
  INFO:   libraries openblas not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anacon
da3\\libs']
  INFO: get_default_fcompiler: matching types: '['gnu', 'intelv', 'absoft', 'compaqv', 'intelev', 'gnu95', 'g95'
, 'intelvem', 'intelem', 'flang']'
  INFO: customize GnuFCompiler
  INFO: customize IntelVisualFCompiler
  INFO: customize AbsoftFCompiler
  INFO: customize CompaqVisualFCompiler
  INFO: customize IntelItaniumVisualFCompiler
  INFO: customize Gnu95FCompiler
  Using built-in specs.
  COLLECT_GCC=C:\Users\senth\anaconda3\Library\mingw-w64\bin\gfortran.exe
  COLLECT_LTO_WRAPPER=C:/Users/senth/anaconda3/Library/mingw-w64/bin/../lib/gcc/x86_64-w64-mingw32/5.3.0/lto-wra
pper.exe
  Target: x86_64-w64-mingw32
  Configured with: ../gcc-5.3.0/configure --prefix=/mingw64 --with-local-prefix=/mingw64/local --build=x86_64-w6
4-mingw32 --host=x86_64-w64-mingw32 --target=x86_64-w64-mingw32 --with-native-system-header-dir=/mingw64/x86_64-
w64-mingw32/include --libexecdir=/mingw64/lib --with-gxx-include-dir=/mingw64/include/c++/5.3.0 --enable-bootstr
ap --with-arch=x86-64 --with-tune=generic --enable-languages=c,lto,c++,objc,obj-c++,fortran,ada --enable-shared
--enable-static --enable-libatomic --enable-threads=posix --enable-graphite --enable-fully-dynamic-string --enab
le-libstdcxx-time=yes --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-version-specific-runtime-libs -
-disable-isl-version-check --enable-lto --enable-libgomp --disable-multilib --enable-checking=release --disable-
rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-libiconv --with-system-zl
ib --with-gmp=/mingw64 --with-mpfr=/mingw64 --with-mpc=/mingw64 --with-isl=/mingw64 --with-pkgversion='Rev5, Bui
lt by MSYS2 project' --with-bugurl=https://sourceforge.net/projects/msys2 --with-gnu-as --with-gnu-ld
  Thread model: posix
  gcc version 5.3.0 (Rev5, Built by MSYS2 project)
  INFO:   NOT AVAILABLE
  INFO:
  INFO: openblas_clapack_info:
  INFO:   libraries openblas,lapack not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\
\anaconda3\\libs']
  INFO: get_default_fcompiler: matching types: '['gnu', 'intelv', 'absoft', 'compaqv', 'intelev', 'gnu95', 'g95'
, 'intelvem', 'intelem', 'flang']'
  INFO: customize GnuFCompiler
  INFO: customize IntelVisualFCompiler
  INFO: customize AbsoftFCompiler
  INFO: customize CompaqVisualFCompiler
  INFO: customize IntelItaniumVisualFCompiler
  INFO: customize Gnu95FCompiler
  Using built-in specs.
  COLLECT_GCC=C:\Users\senth\anaconda3\Library\mingw-w64\bin\gfortran.exe
  COLLECT_LTO_WRAPPER=C:/Users/senth/anaconda3/Library/mingw-w64/bin/../lib/gcc/x86_64-w64-mingw32/5.3.0/lto-wra
pper.exe
  Target: x86_64-w64-mingw32
  Configured with: ../gcc-5.3.0/configure --prefix=/mingw64 --with-local-prefix=/mingw64/local --build=x86_64-w6
4-mingw32 --host=x86_64-w64-mingw32 --target=x86_64-w64-mingw32 --with-native-system-header-dir=/mingw64/x86_64-
w64-mingw32/include --libexecdir=/mingw64/lib --with-gxx-include-dir=/mingw64/include/c++/5.3.0 --enable-bootstr
ap --with-arch=x86-64 --with-tune=generic --enable-languages=c,lto,c++,objc,obj-c++,fortran,ada --enable-shared
--enable-static --enable-libatomic --enable-threads=posix --enable-graphite --enable-fully-dynamic-string --enab
le-libstdcxx-time=yes --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-version-specific-runtime-libs -
-disable-isl-version-check --enable-lto --enable-libgomp --disable-multilib --enable-checking=release --disable-
rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-libiconv --with-system-zl
```

```
ib --with-gmp=/mingw64 --with-mpfr=/mingw64 --with-mpc=/mingw64 --with-isl=/mingw64 --with-pkgversion='Rev5, Bui
lt by MSYS2 project' --with-bugurl=https://sourceforge.net/projects/msys2 --with-gnu-as --with-gnu-ld
  Thread model: posix
  gcc version 5.3.0 (Rev5, Built by MSYS2 project)
  INFO:   NOT AVAILABLE
  INFO:
  INFO: flame_info:
  INFO:   libraries flame not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda3
\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_3_10_threads_info:
  INFO: Setting PTATLAS=ATLAS
  INFO:   libraries tatlas,tatlas not found in C:\Users\senth\anaconda3\lib
  INFO:   libraries tatlas,tatlas not found in C:\
  INFO:   libraries tatlas,tatlas not found in C:\Users\senth\anaconda3\libs
  INFO: <class 'numpy.distutils.system_info.atlas_3_10_threads_info'>
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_3_10_info:
  INFO:   libraries satlas,satlas not found in C:\Users\senth\anaconda3\lib
  INFO:   libraries satlas,satlas not found in C:\
  INFO:   libraries satlas,satlas not found in C:\Users\senth\anaconda3\libs
  INFO: <class 'numpy.distutils.system_info.atlas_3_10_info'>
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_threads_info:
  INFO: Setting PTATLAS=ATLAS
  INFO:   libraries ptf77blas,ptcblas,atlas not found in C:\Users\senth\anaconda3\lib
  INFO:   libraries ptf77blas,ptcblas,atlas not found in C:\
  INFO:   libraries ptf77blas,ptcblas,atlas not found in C:\Users\senth\anaconda3\libs
  INFO: <class 'numpy.distutils.system_info.atlas_threads_info'>
  INFO:   NOT AVAILABLE
  INFO:
  INFO: atlas_info:
  INFO:   libraries f77blas,cblas,atlas not found in C:\Users\senth\anaconda3\lib
  INFO:   libraries f77blas,cblas,atlas not found in C:\
  INFO:   libraries f77blas,cblas,atlas not found in C:\Users\senth\anaconda3\libs
  INFO: <class 'numpy.distutils.system_info.atlas_info'>
  INFO:   NOT AVAILABLE
  INFO:
  INFO: lapack_info:
  INFO:   libraries lapack not found in ['C:\\Users\\senth\\anaconda3\\lib', 'C:\\', 'C:\\Users\\senth\\anaconda
3\\libs']
  INFO:   NOT AVAILABLE
  INFO:
  C:\Users\senth\AppData\Local\Temp\pip-install-89qbc1u6\numpy_0e0e205ce6734dd1ada168290df6f5e0\numpy\distutils\
system_info.py:1902: UserWarning:
      Lapack (http://www.netlib.org/lapack/) libraries not found.
      Directories to search for the libraries can be specified in the
      numpy/distutils/site.cfg file (section [lapack]) or by setting
      the LAPACK environment variable.
    return getattr(self, '_calc_info_{}'.format(name))()
  INFO: lapack_src_info:
  INFO:   NOT AVAILABLE
  INFO:
  C:\Users\senth\AppData\Local\Temp\pip-install-89qbc1u6\numpy_0e0e205ce6734dd1ada168290df6f5e0\numpy\distutils\
system_info.py:1902: UserWarning:
      Lapack (http://www.netlib.org/lapack/) sources not found.
      Directories to search for the sources can be specified in the
      numpy/distutils/site.cfg file (section [lapack_src]) or by setting
      the LAPACK_SRC environment variable.
    return getattr(self, '_calc_info_{}'.format(name))()
  INFO:   NOT AVAILABLE
  INFO:
  INFO: numpy_linalg_lapack_lite:
  INFO:   FOUND:
  INFO:     language = c
  INFO:     define_macros = [('HAVE_BLAS_ILP64', None), ('BLAS_SYMBOL_SUFFIX', '64_')]
  INFO:
  Warning: attempted relative import with no known parent package
  C:\Users\senth\AppData\Local\Temp\pip-build-env-zxn39aru\overlay\Lib\site-packages\setuptools\_distutils\dist.
py:275: UserWarning: Unknown distribution option: 'define_macros'
    warnings.warn(msg)
  running bdist_wheel
  running build
  running config_cc
  INFO: unifing config_cc, config, build_clib, build_ext, build commands --compiler options
  running config_fc
  INFO: unifing config_fc, config, build_clib, build_ext, build commands --fcompiler options
  running build_src
  INFO: build_src
  INFO: building py_modules sources
  creating build
  creating build\src.win-amd64-3.11
  creating build\src.win-amd64-3.11\numpy
  creating build\src.win-amd64-3.11\numpy\distutils
  INFO: building library "npymath" sources
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visu
alstudio.microsoft.com/visual-cpp-build-tools/
  [end of output]

  note: This error originates from a subprocess, and is likely not a problem with pip.
  ERROR: Failed building wheel for numpy
ERROR: Could not build wheels for numpy, which is required to install pyproject.toml-based projects
```

In [75]:
```
!pip install prophet
```

```
Requirement already satisfied: prophet in c:\users\senth\anaconda3\lib\site-packages (1.1.6)
Requirement already satisfied: cmdstanpy>=1.0.4 in c:\users\senth\anaconda3\lib\site-packages (from prophet) (1.
2.5)
Requirement already satisfied: numpy>=1.15.4 in c:\users\senth\anaconda3\lib\site-packages (from prophet) (1.24.
3)
Requirement already satisfied: matplotlib>=2.0.0 in c:\users\senth\anaconda3\lib\site-packages (from prophet) (3
.7.1)
Requirement already satisfied: pandas>=1.0.4 in c:\users\senth\anaconda3\lib\site-packages (from prophet) (1.5.3
)
Requirement already satisfied: holidays<1,>=0.25 in c:\users\senth\anaconda3\lib\site-packages (from prophet) (0
.62)
Requirement already satisfied: tqdm>=4.36.1 in c:\users\senth\anaconda3\lib\site-packages (from prophet) (4.65.0
)
Requirement already satisfied: importlib-resources in c:\users\senth\anaconda3\lib\site-packages (from prophet)
(6.4.5)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in c:\users\senth\anaconda3\lib\site-packages (from cmdstanp
y>=1.0.4->prophet) (0.5.1)
Requirement already satisfied: python-dateutil in c:\users\senth\anaconda3\lib\site-packages (from holidays<1,>=
0.25->prophet) (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>=
2.0.0->prophet) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>=2.0.
0->prophet) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>
=2.0.0->prophet) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>
=2.0.0->prophet) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>=2
.0.0->prophet) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>=2.0
.0->prophet) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\senth\anaconda3\lib\site-packages (from matplotlib>=
2.0.0->prophet) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\users\senth\anaconda3\lib\site-packages (from pandas>=1.0.4->p
rophet) (2022.7)
Requirement already satisfied: colorama in c:\users\senth\anaconda3\lib\site-packages (from tqdm>=4.36.1->prophe
t) (0.4.6)
Requirement already satisfied: six>=1.5 in c:\users\senth\anaconda3\lib\site-packages (from python-dateutil->hol
idays<1,>=0.25->prophet) (1.16.0)
```

In [76]:
```
time_series = aggregated_data
time_series = time_series.reset_index()
time_series = time_series[['index', 'English']]
time_series.columns = ['ds', 'y']
exog = Exog_Campaign_eng.copy(deep = True)
time_series['exog'] = exog.values
```

In [77]:
```
time_series
```

Out[77]:

	ds	y	exog
0	2015-07-01	3513.862203	0
1	2015-07-02	3502.511407	0
2	2015-07-03	3325.357889	0
3	2015-07-04	3462.054256	0
4	2015-07-05	3575.520035	0
...	...	...	...
545	2016-12-27	6040.680728	1
546	2016-12-28	5860.227559	1
547	2016-12-29	6245.127510	1
548	2016-12-30	5201.783018	0
549	2016-12-31	5127.916418	0

550 rows × 3 columns

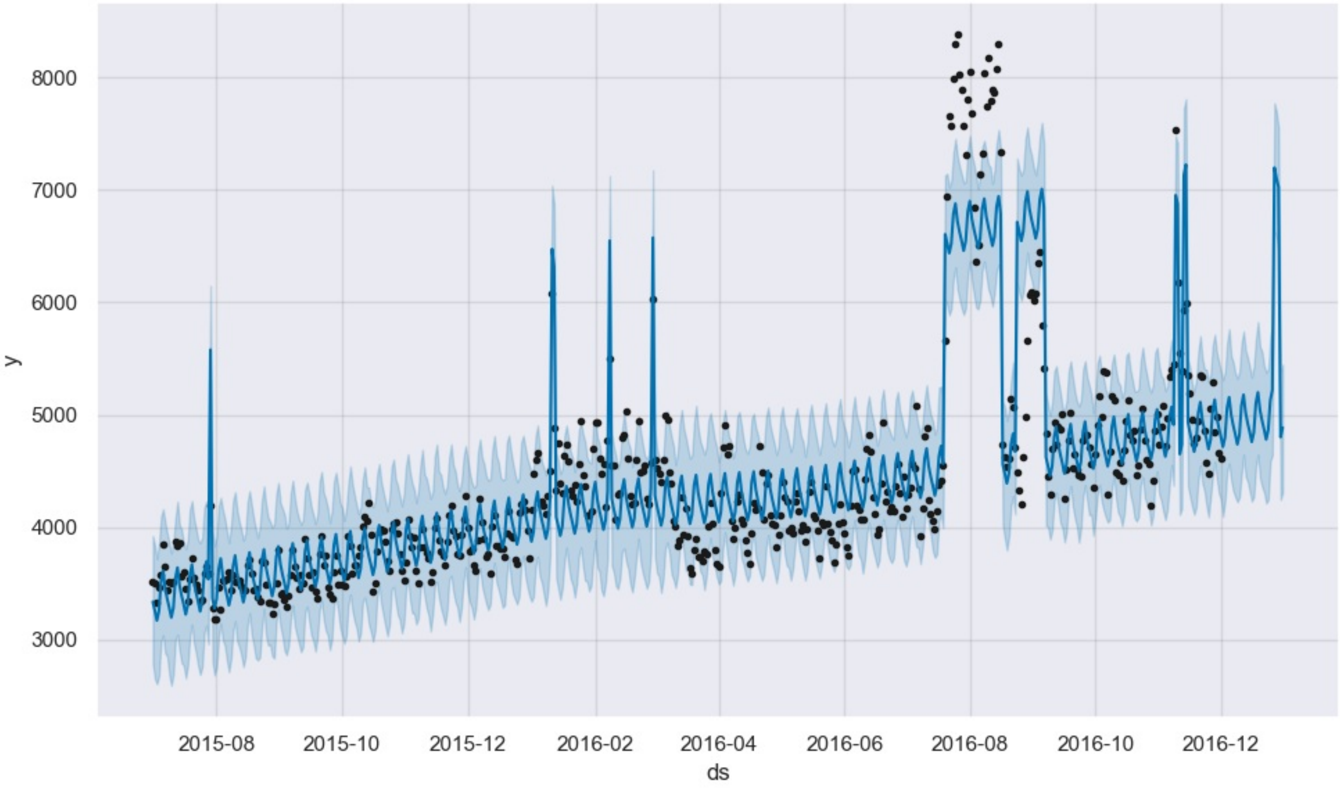
In [86]:

```
from prophet import Prophet
```

In [89]:

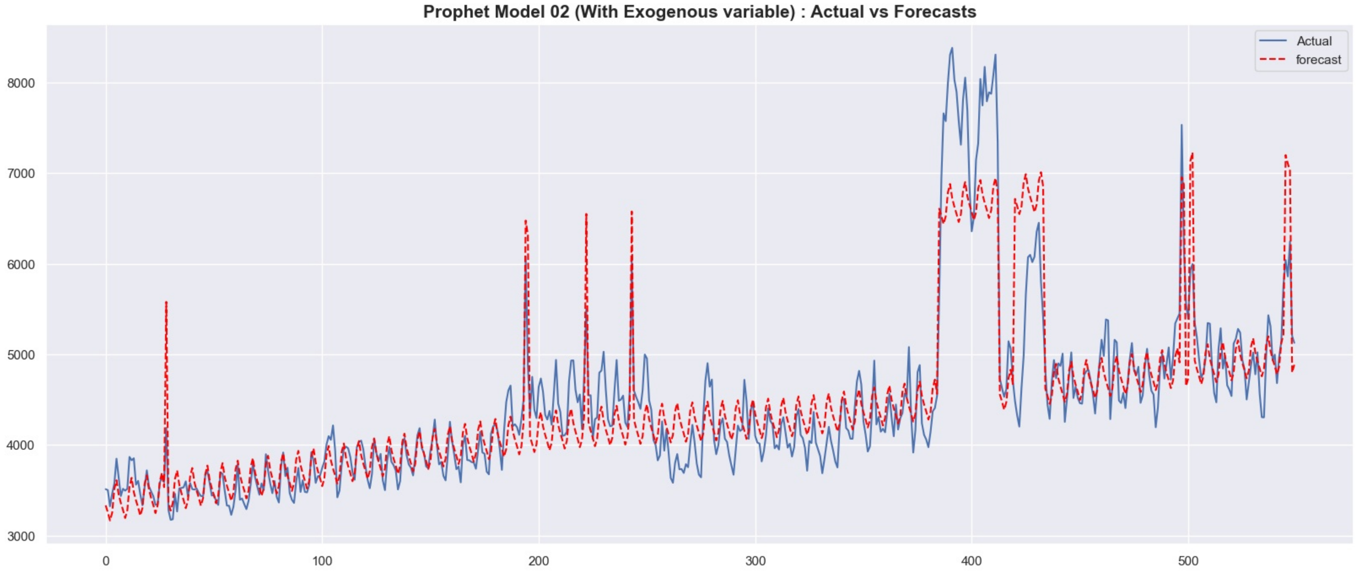
```
prophet2 = Prophet(weekly_seasonality=True)
prophet2.add_regressor('exog')
prophet2.fit(time_series[:-30])
#future2 = prophet2.make_future_dataframe(periods=30, freq= 'D')
forecast2 = prophet2.predict(time_series)
fig2 = prophet2.plot(forecast2)
```

```
16:41:32 - cmdstanpy - INFO - Chain [1] start processing
16:41:32 - cmdstanpy - INFO - Chain [1] done processing
```



In [90]:

```
actual = time_series['y'].values
forecast = forecast2['yhat'].values

plt.figure(figsize = (20,8))
plt.plot(actual, label = 'Actual')
plt.plot(forecast, label = 'forecast', color = 'red', linestyle='dashed')
plt.legend(loc="upper right")
plt.title(f'Prophet Model 02 (With Exogenous variable) : Actual vs Forecasts', fontsize = 15, fontweight = 'bol
plt.show()
```



In [91]:
```
errors = abs(actual - forecast)
mape = np.mean(errors/abs(actual))
mape
```

Out[91]:
```
0.059450593946555136
```

Inferences and Recommendations :

- inferences made from the data visualizations:
  - Total 7 languages found in data.
  - English has the highest number of pages.
  - 3 access types:
    - all-access 51.2295 %
    - mobile-web 24.7748 %
    - desktop 23.9958 %
  - 2 access origins:
    - agents 75.932526 %
    - spider 24.067474 %
  - English language has the highest pages.
  - Maximum ads should be run on English Page.

- What does the decomposition of series do?
  - 0The decomposition of a time series refers to the process of separating a time series into its components, such as trend, seasonality, and residuals.
  - These components are intended to represent different underlying patterns in the data. The idea behind decomposition is to break down a complex time series into simpler components that can be more easily understood and analyzed.
  - Trend component represents the underlying pattern in the data over time, reflecting long-term changes.
  - Seasonality component represents regular patterns that repeat over a fixed interval, such as daily, weekly, or yearly.
  - Residual component represents the remaining random fluctuations in the data after removing the trend and seasonality components.
  - Decomposition is often used in time series analysis to identify and isolate different patterns in the data and to forecast future values. It is also used to remove seasonality and trend components from the data before applying statistical or machine learning models to the residuals, as this can help to improve the performance of these models.

- What level of differencing gave you a stationary series?
  - Stationarity is an important property of a time series because many time series analysis techniques assume that the time series is stationary.

- A time series is stationary if its mean, variance, and autocorrelation structure are constant over time.
- Differencing is a common technique used to make a time series stationary.
- It involves subtracting the value of the time series at a previous time step from the current time step.
- This can help to remove trend and seasonality components from the data, making it more stationary.
- The order of differencing refers to the number of times the differencing operation is performed.
- in this case study, differencing once yield a stationary time series.

- Difference between arima, sarima & sarimax.

- ARIMA (AutoRegressive Integrated Moving Average) is a statistical model for time series data that accounts for both autoregression (the use of past values to predict future values) and moving average (the use of the residuals of past predictions to predict future values).
- It is a flexible method for modeling non-stationary time series data and can be used for both univariate and multivariate time series.
- ARIMA models are denoted by the notations ARIMA(p, d, q), where p is the order of the autoregression component, d is the order of differencing used to make the time series stationary, and q is the order of the moving average component.

- SARIMA (Seasonal AutoRegressive Integrated Moving Average) is a variation of ARIMA that accounts for both seasonality and non-stationarity in time series data.
- Seasonality refers to repeating patterns in the data over fixed time intervals, such as daily, weekly, or yearly. SARIMA models are denoted by the notations SARIMA(p, d, q)(P, D, Q, S), where p, d, and q are the same as in ARIMA models, P is the order of the seasonal autoregression component, D is the order of seasonal differencing, Q is the order of the seasonal moving average component, and S is the number of seasons in the data.

- SARIMAX (Seasonal AutoRegressive Integrated Moving Average with exogenous regressors) is an extension of SARIMA that allows for the inclusion of exogenous variables, or variables that are not part of the time series data, in the modeling process.
- SARIMAX models are useful when the time series data is influenced by other variables that are not part of the time series data, and can provide more accurate forecasts.
- SARIMAX models are denoted by the notations SARIMAX(p, d, q)(P, D, Q, S)x, where p, d, q, P, D, Q, and S are the same as in SARIMA models and x represents the number of exogenous variables included in the model.

- The equation for a SARIMA (Seasonal AutoRegressive Integrated Moving Average) model can be expressed as follows:

```
ARIMA(p, d, q)(P, D, Q, S):

   y(t) = c + φ1 * y(t-1) + φ2 * y(t-2) + ... + φp * y(t-p)
            + θ1 * e(t-1) + θ2 * e(t-2) + ... + θq * e(t-q)
            + δ * y(t-S) + Φ1 * y(t-S-1) + Φ2 * y(t-S-2) + ... + ΦP * y(t-S-P)
            + θ1 * e(t-S-1) + θ2 * e(t-S-2) + ... + θQ * e(t-S-Q) + e(t)

          where:

          y(t) is the value of the time series at time step t.
          c is a constant.
          φ1, φ2, ..., φp are the autoregression coefficients.
          θ1, θ2, ..., θq are the moving average coefficients.
          δ is a coefficient for the seasonal autoregression term.
          Φ1, Φ2, ..., ΦP are the seasonal autoregression coefficients.
          θ1, θ2, ..., θQ are the seasonal moving average coefficients.
          e(t), e(t-1), ..., e(t-q), e(t-S), e(t-S-1), ..., e(t-S-Q) are the residuals.
 - In a SARIMA model, the order of differencing (d) is used to make the time series stationary,
  the autoregression and moving average components (p and q) are used to model the
autocorrelation structure of the residuals,
  and the seasonal components (P, D, Q, and S) are used to model the seasonal patterns in the
data.
   The coefficients in the model are estimated using maximum likelihood estimation or other
optimization techniques,
    and the residuals are used to assess the goodness-of-fit of the model.
```

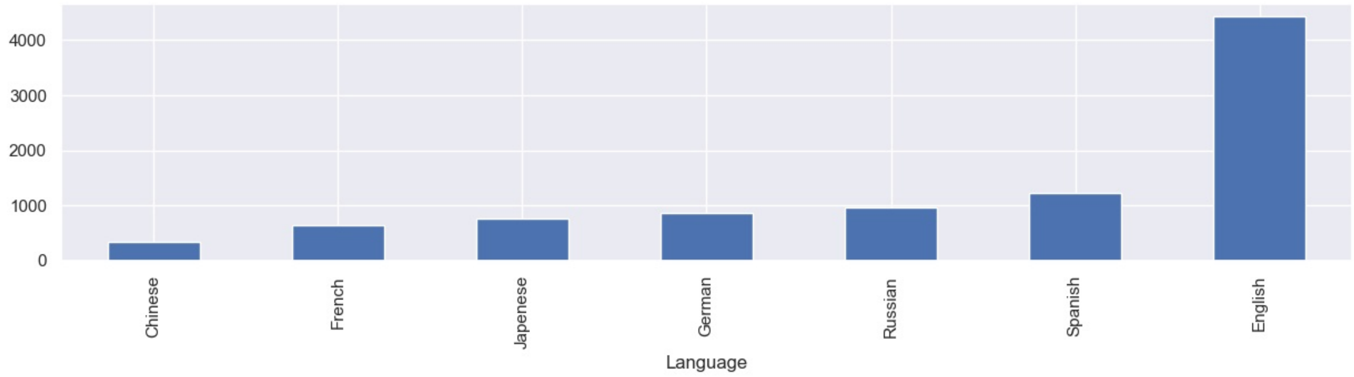
- Compare the number of views in different languages

In [92]:

```
aggregated_data.mean().sort_values().plot(kind = 'bar')
```

Out[92]:

```
<Axes: xlabel='Language'>
```



- What other methods other than grid search would be suitable to get the model for all languages?
  - When estimating the values of p, q, and d from the ACF and PACF plots of a time series, the following steps can be taken:
    - Determine if the time series is stationary by conducting an augmented Dickey-Fuller test.
    - If the time series is stationary, attempt to fit an ARMA model. If it is non-stationary, determine the value of d.
    - If stationarity is achieved, plot the autocorrelation and partial autocorrelation graphs of the data.
    - Plot the partial autocorrelation graph (PACF) to determine the value of p, as the cut-off point in the PACF is equal to p.
    - Plot the autocorrelation graph (ACF) to determine the value of q, as the cut-off point in the ACF is equal to q

```
In [ ]:
```