In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from sklearn.impute import SimpleImputer
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
from sklearn.metrics import mean_absolute_percentage_error as mape
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima.model import ARIMA
```

In [2]:
```
df = pd.read_csv('train_1.csv')
```

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

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

NAN Values

In [4]:
```
df.isna().sum()
```

Out[4]:

	Page
2015-07-01	20740
2015-07-02	20816
2015-07-03	20544
2015-07-04	20654
2015-07-05	20659
2015-07-06	20483
2015-07-07	20664
2015-07-08	20294
2015-07-09	20244
2015-07-10	20342
2015-07-11	20525
2015-07-12	20485
2015-07-13	20399
2015-07-14	20140
2015-07-15	20106
2015-07-16	19987
2015-07-17	20048
2015-07-18	20295
2015-07-19	20142
2015-07-20	19979
2015-07-21	19688
2015-07-22	19573
2015-07-23	19581
2015-07-24	19593
2015-07-25	19589
2015-07-26	19865
2015-07-27	19759
2015-07-28	19553
2015-07-29	19347
2015-07-30	19592
2015-07-31	19708
2015-08-01	19640
2015-08-02	19844
2015-08-03	19842
2015-08-04	19595
2015-08-05	19607
2015-08-06	19424
2015-08-07	19528
2015-08-08	19525
2015-08-09	19320
2015-08-10	19370
2015-08-11	19385
2015-08-12	19507
2015-08-13	19317
2015-08-14	19272
2015-08-15	19320
2015-08-16	19213
2015-08-17	18954
2015-08-18	19104
2015-08-19	18954
2015-08-20	18923
2015-08-21	19012
2015-08-22	18973
2015-08-23	19039
2015-08-24	18781
2015-08-25	18704
2015-08-26	18526
2015-08-27	18870
2015-08-28	18691
2015-08-29	18926
2015-08-30	18993
2015-08-31	18915
2015-09-01	18793

2015-09-02	18716
2015-09-03	18404
2015-09-04	18412
2015-09-05	18464
2015-09-06	18543
2015-09-07	18296
2015-09-08	18263
2015-09-09	18448
2015-09-10	18497
2015-09-11	18329
2015-09-12	18279
2015-09-13	18235
2015-09-14	18407
2015-09-15	18313
2015-09-16	18177
2015-09-17	18097
2015-09-18	18207
2015-09-19	18244
2015-09-20	18345
2015-09-21	18231
2015-09-22	18345
2015-09-23	18155
2015-09-24	18067
2015-09-25	18015
2015-09-26	18102
2015-09-27	18153
2015-09-28	17812
2015-09-29	17897
2015-09-30	17807
2015-10-01	17759
2015-10-02	17695
2015-10-03	17835
2015-10-04	17714
2015-10-05	17594
2015-10-06	17528
2015-10-07	17598
2015-10-08	17725
2015-10-09	17771
2015-10-10	17746
2015-10-11	17844
2015-10-12	17419
2015-10-13	17425
2015-10-14	17244
2015-10-15	17146
2015-10-16	17356
2015-10-17	17275
2015-10-18	17337
2015-10-19	17411
2015-10-20	17206
2015-10-21	17150
2015-10-22	16965
2015-10-23	17189
2015-10-24	17513
2015-10-25	17238
2015-10-26	17216
2015-10-27	17149
2015-10-28	17196
2015-10-29	16743
2015-10-30	16810
2015-10-31	17079
2015-11-01	17094
2015-11-02	17078
2015-11-03	15734
2015-11-04	15714

2015-11-05	15791
2015-11-06	15841
2015-11-07	15930
2015-11-08	15825
2015-11-09	15769
2015-11-10	15699
2015-11-11	15842
2015-11-12	15761
2015-11-13	15895
2015-11-14	15911
2015-11-15	15852
2015-11-16	15835
2015-11-17	15974
2015-11-18	15734
2015-11-19	15419
2015-11-20	15872
2015-11-21	15888
2015-11-22	15743
2015-11-23	15620
2015-11-24	15662
2015-11-25	15793
2015-11-26	15599
2015-11-27	15735
2015-11-28	15847
2015-11-29	15778
2015-11-30	15644
2015-12-01	15449
2015-12-02	15324
2015-12-03	15184
2015-12-04	15354
2015-12-05	15480
2015-12-06	15335
2015-12-07	15338
2015-12-08	15060
2015-12-09	15245
2015-12-10	15187
2015-12-11	15188
2015-12-12	15128
2015-12-13	14884
2015-12-14	14970
2015-12-15	14900
2015-12-16	14846
2015-12-17	14814
2015-12-18	14883
2015-12-19	14966
2015-12-20	14498
2015-12-21	14695
2015-12-22	14710
2015-12-23	14647
2015-12-24	14642
2015-12-25	14667
2015-12-26	14518
2015-12-27	14521
2015-12-28	14247
2015-12-29	14253
2015-12-30	14286
2015-12-31	14303
2016-01-01	14415
2016-01-02	14150
2016-01-03	14313
2016-01-04	14138
2016-01-05	14010
2016-01-06	13950
2016-01-07	13890

```
2016-01-08    13879
2016-01-09    13787
2016-01-10    13861
2016-01-11    13712
2016-01-12    13425
2016-01-13    13390
2016-01-14    13396
2016-01-15    13514
2016-01-16    13625
2016-01-17    13667
2016-01-18    13448
2016-01-19    13260
2016-01-20    13355
2016-01-21    13176
2016-01-22    13248
2016-01-23    13222
2016-01-24    13016
2016-01-25    12932
2016-01-26    13019
2016-01-27    12897
2016-01-28    12750
2016-01-29    12652
2016-01-30    12652
2016-01-31    12550
2016-02-01    12700
2016-02-02    12744
2016-02-03    12709
2016-02-04    12673
2016-02-05    12696
2016-02-06    12501
2016-02-07    12595
2016-02-08    12525
2016-02-09    12636
2016-02-10    12380
2016-02-11    12057
2016-02-12    12261
2016-02-13    12375
2016-02-14    12417
2016-02-15    12382
2016-02-16    12253
2016-02-17    12140
2016-02-18    12196
2016-02-19    12196
2016-02-20    12137
2016-02-21    12148
2016-02-22    11998
2016-02-23    11868
2016-02-24    12116
2016-02-25    11935
2016-02-26    11707
2016-02-27    11740
2016-02-28    11917
2016-02-29    11923
2016-03-01    11730
2016-03-02    11724
2016-03-03    11755
2016-03-04    11431
2016-03-05    11629
2016-03-06    11665
2016-03-07    11485
2016-03-08    11616
2016-03-09    11661
2016-03-10    11738
2016-03-11    11850
```

2016-03-12	11725
2016-03-13	11582
2016-03-14	11527
2016-03-15	11329
2016-03-16	11440
2016-03-17	10926
2016-03-18	11312
2016-03-19	11123
2016-03-20	11229
2016-03-21	11123
2016-03-22	11116
2016-03-23	10939
2016-03-24	10872
2016-03-25	11006
2016-03-26	10927
2016-03-27	10853
2016-03-28	10774
2016-03-29	10765
2016-03-30	10528
2016-03-31	10593
2016-04-01	10385
2016-04-02	10489
2016-04-03	10744
2016-04-04	10279
2016-04-05	10024
2016-04-06	10053
2016-04-07	10103
2016-04-08	10007
2016-04-09	10175
2016-04-10	10152
2016-04-11	10036
2016-04-12	10059
2016-04-13	9950
2016-04-14	9727
2016-04-15	9916
2016-04-16	10130
2016-04-17	9998
2016-04-18	9561
2016-04-19	9961
2016-04-20	9718
2016-04-21	9614
2016-04-22	9560
2016-04-23	9880
2016-04-24	9730
2016-04-25	9555
2016-04-26	9679
2016-04-27	9371
2016-04-28	9743
2016-04-29	9457
2016-04-30	9629
2016-05-01	9705
2016-05-02	9560
2016-05-03	9265
2016-05-04	9259
2016-05-05	9499
2016-05-06	9295
2016-05-07	9204
2016-05-08	9207
2016-05-09	9325
2016-05-10	9073
2016-05-11	9181
2016-05-12	9183
2016-05-13	8996
2016-05-14	9320

2016-05-15	9222
2016-05-16	9096
2016-05-17	9247
2016-05-18	8995
2016-05-19	8971
2016-05-20	9320
2016-05-21	9216
2016-05-22	9012
2016-05-23	8845
2016-05-24	8936
2016-05-25	8974
2016-05-26	8909
2016-05-27	9055
2016-05-28	8787
2016-05-29	9001
2016-05-30	8622
2016-05-31	8431
2016-06-01	8351
2016-06-02	8324
2016-06-03	8488
2016-06-04	8226
2016-06-05	8433
2016-06-06	8670
2016-06-07	8477
2016-06-08	8484
2016-06-09	8390
2016-06-10	8380
2016-06-11	7981
2016-06-12	8433
2016-06-13	8083
2016-06-14	8045
2016-06-15	8071
2016-06-16	8152
2016-06-17	8224
2016-06-18	7971
2016-06-19	8302
2016-06-20	8074
2016-06-21	7930
2016-06-22	7966
2016-06-23	7728
2016-06-24	7882
2016-06-25	7677
2016-06-26	7992
2016-06-27	7774
2016-06-28	7896
2016-06-29	7738
2016-06-30	7892
2016-07-01	7612
2016-07-02	7696
2016-07-03	8107
2016-07-04	7855
2016-07-05	7800
2016-07-06	7446
2016-07-07	7649
2016-07-08	7946
2016-07-09	7508
2016-07-10	7836
2016-07-11	7569
2016-07-12	7210
2016-07-13	7441
2016-07-14	7193
2016-07-15	7684
2016-07-16	7313
2016-07-17	7819

2016-07-18	7523
2016-07-19	7415
2016-07-20	7310
2016-07-21	7384
2016-07-22	7153
2016-07-23	6985
2016-07-24	7443
2016-07-25	7159
2016-07-26	7087
2016-07-27	7116
2016-07-28	6818
2016-07-29	6873
2016-07-30	6826
2016-07-31	7154
2016-08-01	7023
2016-08-02	6964
2016-08-03	6793
2016-08-04	6917
2016-08-05	6799
2016-08-06	6677
2016-08-07	7169
2016-08-08	6879
2016-08-09	6833
2016-08-10	6813
2016-08-11	6712
2016-08-12	6735
2016-08-13	6366
2016-08-14	6722
2016-08-15	6655
2016-08-16	6693
2016-08-17	6599
2016-08-18	6601
2016-08-19	6413
2016-08-20	6060
2016-08-21	6275
2016-08-22	6135
2016-08-23	6010
2016-08-24	6137
2016-08-25	5978
2016-08-26	6077
2016-08-27	5988
2016-08-28	6310
2016-08-29	6022
2016-08-30	6076
2016-08-31	6091
2016-09-01	5842
2016-09-02	6126
2016-09-03	5673
2016-09-04	5991
2016-09-05	5999
2016-09-06	5638
2016-09-07	5529
2016-09-08	5606
2016-09-09	5882
2016-09-10	5637
2016-09-11	5577
2016-09-12	5205
2016-09-13	5233
2016-09-14	5466
2016-09-15	5243
2016-09-16	5334
2016-09-17	5279
2016-09-18	5613
2016-09-19	5560

2016-09-20	5455
2016-09-21	5600
2016-09-22	5426
2016-09-23	5457
2016-09-24	5138
2016-09-25	5627
2016-09-26	5355
2016-09-27	5518
2016-09-28	5556
2016-09-29	5464
2016-09-30	5481
2016-10-01	4989
2016-10-02	5497
2016-10-03	5434
2016-10-04	5423
2016-10-05	5245
2016-10-06	5480
2016-10-07	5489
2016-10-08	4932
2016-10-09	5355
2016-10-10	5226
2016-10-11	5018
2016-10-12	5069
2016-10-13	4941
2016-10-14	4952
2016-10-15	4720
2016-10-16	4928
2016-10-17	4762
2016-10-18	4858
2016-10-19	4803
2016-10-20	4980
2016-10-21	4881
2016-10-22	4355
2016-10-23	4882
2016-10-24	5143
2016-10-25	4864
2016-10-26	4964
2016-10-27	4829
2016-10-28	4739
2016-10-29	4424
2016-10-30	4768
2016-10-31	4807
2016-11-01	4739
2016-11-02	4471
2016-11-03	4539
2016-11-04	4665
2016-11-05	4405
2016-11-06	4626
2016-11-07	4386
2016-11-08	4529
2016-11-09	4866
2016-11-10	4709
2016-11-11	4632
2016-11-12	4234
2016-11-13	4661
2016-11-14	4422
2016-11-15	4639
2016-11-16	4498
2016-11-17	4357
2016-11-18	4380
2016-11-19	4169
2016-11-20	4502
2016-11-21	4401
2016-11-22	4153

```
2016-11-23     4374
2016-11-24     4295
2016-11-25     4372
2016-11-26     3779
2016-11-27     4416
2016-11-28     3979
2016-11-29     4071
2016-11-30     4229
2016-12-01     4060
2016-12-02     4243
2016-12-03     3675
2016-12-04     4290
2016-12-05     4231
2016-12-06     4103
2016-12-07     4130
2016-12-08     3962
2016-12-09     4179
2016-12-10     3625
2016-12-11     3581
2016-12-12     3538
2016-12-13     3802
2016-12-14     4108
2016-12-15     4078
2016-12-16     3566
2016-12-17     3559
2016-12-18     3666
2016-12-19     3652
2016-12-20     3268
2016-12-21     3236
2016-12-22     3853
2016-12-23     3584
2016-12-24     3189
2016-12-25     3744
2016-12-26     3918
2016-12-27     3701
2016-12-28     3822
2016-12-29     3826
2016-12-30     3635
2016-12-31     3465
dtype: int64
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145063 entries, 0 to 145062
Columns: 551 entries, Page to 2016-12-31
dtypes: float64(550), object(1)
memory usage: 609.8+ MB
```

In [6]:
```
exog = pd.read_csv('Exog_Campaign_eng.csv')
```

In [7]:
```
exog.shape
```

Out[7]:
```
(550, 1)
```

In [8]:
```
df.head(10)
```

Out[8]:

	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	20 07
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	1
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	4
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	
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	1
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
5	5566_zh.wikipedia.org_all-access_spider	12.0	7.0	4.0	5.0	20.0	8.0	5.0	17.0	24.0	7.0	1
6	91Days_zh.wikipedia.org_all-access_spider	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
7	A'N'D_zh.wikipedia.org_all-access_spider	118.0	26.0	30.0	24.0	29.0	127.0	53.0	37.0	20.0	32.0	1
8	AKB48_zh.wikipedia.org_all-access_spider	5.0	23.0	14.0	12.0	9.0	9.0	35.0	15.0	14.0	22.0	
9	ASCII_zh.wikipedia.org_all-access_spider	6.0	3.0	5.0	12.0	6.0	5.0	4.0	13.0	9.0	15.0	1

In [9]:
```
# The page name contains data in this format:

# SPECIFIC NAME _ LANGUAGE.wikipedia.org _ ACCESS TYPE _ ACCESS ORIGIN
```

In [10]:
```
import re
# lang_list.extend([string for string in val if re.search(pattern, string)])
def get_language_list(values):
    ans = []
    pattern = r'_(\w{2,3})\.wikipedia\.org'
    for val in values:
        match = re.search(pattern, val)
        if match:
            language_code = match.group(1)
            ans.append(language_code)  # Output: zh
        else:
            ans.append("NA")
    return ans
```

In [11]:
```
df['language_code'] = get_language_list(df['Page'])
```

In [12]:
```
df['Page'] = df['language_code']
```

In [13]:
```
df.rename({'Page':'Language'},axis=1,inplace=True)
```

In [14]:
```
df.drop('language_code',axis=1,inplace=True)
```

In [15]:
```
Exo_lang = df['Language']
```

In [16]:
```
df = df.loc[:,'2015-07-01':'2016-12-31']
```

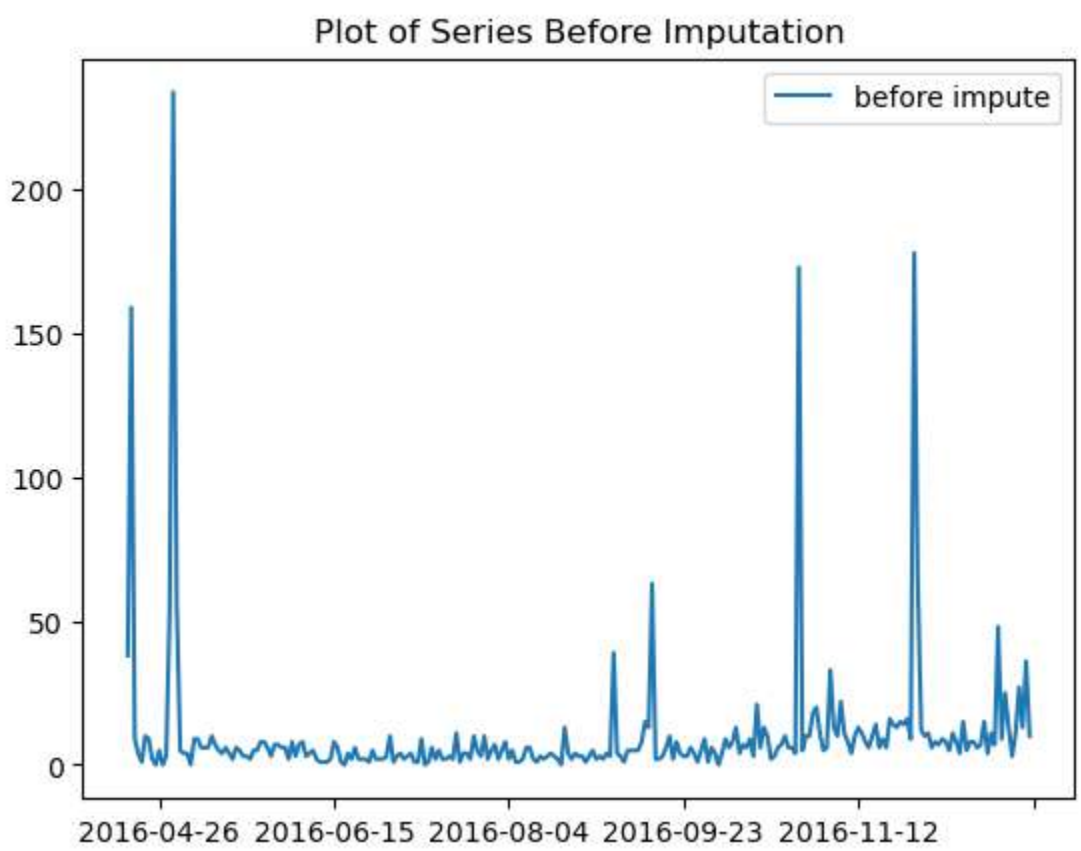
Data Processing

In [17]:
```
(df.tail(20))
```

Out[17]:

	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	2015-07-14	2015-07-15
145043	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145044	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145045	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145046	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145047	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145048	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145049	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145050	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145051	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145052	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145053	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145054	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145055	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145056	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145057	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145058	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145059	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145060	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145061	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
145062	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [18]:
```
plt.title('Plot of Series Before Imputation')
df.iloc[4].plot(label='before impute')
plt.legend()
plt.show()
```



In [19]:
```
np.all(df.loc[145062].isna())
```

Out[19]:
```
True
```

`Approach`

1. Check Whether all Values are NAN.

2. If all values are NAN the Flag those row as 'Y' else 'N'

In [20]:
```
def flag_all_nan_rows(df):
    ans = []
    for i in range(df.shape[0]):
        if(np.all(df.loc[i].isna())==True):
            ans.append('Y')
        else:
            ans.append('N')
    return ans
df['Nan Flag'] = flag_all_nan_rows(df)
```

Filtering all rows which is not having NAN Values.

In [21]:
```
df['Language'] = Exo_lang
```

In [22]:
```
df = df[df['Nan Flag']=='N'].reset_index()
```

In [23]:
```
exolang = df['Language']
```

In [24]:
```
df.drop(['Language','Nan Flag','index'],axis=1,inplace=True)
```

New Shape of Data

In [25]:
```
df.shape
```

Out[25]:
```
(144411, 550)
```

`Approach`

Filling all row NAN values with the median values because each row is independent of each other

In [26]:
```
def fill_na_imputation(df):
    for i in range(df.shape[0]):
        df.loc[i].fillna(df.loc[i].median(),inplace=True)
    return df
```

In [27]:
```
df = fill_na_imputation(df)
```

NAN Values are fixed now

In [28]:
```
df.isna().sum()
```

Out[28]:

2015-07-01	0
2015-07-02	0
2015-07-03	0
2015-07-04	0
2015-07-05	0
2015-07-06	0
2015-07-07	0
2015-07-08	0
2015-07-09	0
2015-07-10	0
2015-07-11	0
2015-07-12	0
2015-07-13	0
2015-07-14	0
2015-07-15	0
2015-07-16	0
2015-07-17	0
2015-07-18	0
2015-07-19	0
2015-07-20	0
2015-07-21	0
2015-07-22	0
2015-07-23	0
2015-07-24	0
2015-07-25	0
2015-07-26	0
2015-07-27	0
2015-07-28	0
2015-07-29	0
2015-07-30	0
2015-07-31	0
2015-08-01	0
2015-08-02	0
2015-08-03	0
2015-08-04	0
2015-08-05	0
2015-08-06	0
2015-08-07	0
2015-08-08	0
2015-08-09	0
2015-08-10	0
2015-08-11	0
2015-08-12	0
2015-08-13	0
2015-08-14	0
2015-08-15	0
2015-08-16	0
2015-08-17	0
2015-08-18	0
2015-08-19	0
2015-08-20	0
2015-08-21	0
2015-08-22	0
2015-08-23	0
2015-08-24	0
2015-08-25	0
2015-08-26	0
2015-08-27	0
2015-08-28	0
2015-08-29	0
2015-08-30	0
2015-08-31	0
2015-09-01	0
2015-09-02	0

```
2015-09-03    0
2015-09-04    0
2015-09-05    0
2015-09-06    0
2015-09-07    0
2015-09-08    0
2015-09-09    0
2015-09-10    0
2015-09-11    0
2015-09-12    0
2015-09-13    0
2015-09-14    0
2015-09-15    0
2015-09-16    0
2015-09-17    0
2015-09-18    0
2015-09-19    0
2015-09-20    0
2015-09-21    0
2015-09-22    0
2015-09-23    0
2015-09-24    0
2015-09-25    0
2015-09-26    0
2015-09-27    0
2015-09-28    0
2015-09-29    0
2015-09-30    0
2015-10-01    0
2015-10-02    0
2015-10-03    0
2015-10-04    0
2015-10-05    0
2015-10-06    0
2015-10-07    0
2015-10-08    0
2015-10-09    0
2015-10-10    0
2015-10-11    0
2015-10-12    0
2015-10-13    0
2015-10-14    0
2015-10-15    0
2015-10-16    0
2015-10-17    0
2015-10-18    0
2015-10-19    0
2015-10-20    0
2015-10-21    0
2015-10-22    0
2015-10-23    0
2015-10-24    0
2015-10-25    0
2015-10-26    0
2015-10-27    0
2015-10-28    0
2015-10-29    0
2015-10-30    0
2015-10-31    0
2015-11-01    0
2015-11-02    0
2015-11-03    0
2015-11-04    0
2015-11-05    0
```

2015-11-06	0
2015-11-07	0
2015-11-08	0
2015-11-09	0
2015-11-10	0
2015-11-11	0
2015-11-12	0
2015-11-13	0
2015-11-14	0
2015-11-15	0
2015-11-16	0
2015-11-17	0
2015-11-18	0
2015-11-19	0
2015-11-20	0
2015-11-21	0
2015-11-22	0
2015-11-23	0
2015-11-24	0
2015-11-25	0
2015-11-26	0
2015-11-27	0
2015-11-28	0
2015-11-29	0
2015-11-30	0
2015-12-01	0
2015-12-02	0
2015-12-03	0
2015-12-04	0
2015-12-05	0
2015-12-06	0
2015-12-07	0
2015-12-08	0
2015-12-09	0
2015-12-10	0
2015-12-11	0
2015-12-12	0
2015-12-13	0
2015-12-14	0
2015-12-15	0
2015-12-16	0
2015-12-17	0
2015-12-18	0
2015-12-19	0
2015-12-20	0
2015-12-21	0
2015-12-22	0
2015-12-23	0
2015-12-24	0
2015-12-25	0
2015-12-26	0
2015-12-27	0
2015-12-28	0
2015-12-29	0
2015-12-30	0
2015-12-31	0
2016-01-01	0
2016-01-02	0
2016-01-03	0
2016-01-04	0
2016-01-05	0
2016-01-06	0
2016-01-07	0
2016-01-08	0

2016-01-09	0
2016-01-10	0
2016-01-11	0
2016-01-12	0
2016-01-13	0
2016-01-14	0
2016-01-15	0
2016-01-16	0
2016-01-17	0
2016-01-18	0
2016-01-19	0
2016-01-20	0
2016-01-21	0
2016-01-22	0
2016-01-23	0
2016-01-24	0
2016-01-25	0
2016-01-26	0
2016-01-27	0
2016-01-28	0
2016-01-29	0
2016-01-30	0
2016-01-31	0
2016-02-01	0
2016-02-02	0
2016-02-03	0
2016-02-04	0
2016-02-05	0
2016-02-06	0
2016-02-07	0
2016-02-08	0
2016-02-09	0
2016-02-10	0
2016-02-11	0
2016-02-12	0
2016-02-13	0
2016-02-14	0
2016-02-15	0
2016-02-16	0
2016-02-17	0
2016-02-18	0
2016-02-19	0
2016-02-20	0
2016-02-21	0
2016-02-22	0
2016-02-23	0
2016-02-24	0
2016-02-25	0
2016-02-26	0
2016-02-27	0
2016-02-28	0
2016-02-29	0
2016-03-01	0
2016-03-02	0
2016-03-03	0
2016-03-04	0
2016-03-05	0
2016-03-06	0
2016-03-07	0
2016-03-08	0
2016-03-09	0
2016-03-10	0
2016-03-11	0
2016-03-12	0

2016-03-13	0
2016-03-14	0
2016-03-15	0
2016-03-16	0
2016-03-17	0
2016-03-18	0
2016-03-19	0
2016-03-20	0
2016-03-21	0
2016-03-22	0
2016-03-23	0
2016-03-24	0
2016-03-25	0
2016-03-26	0
2016-03-27	0
2016-03-28	0
2016-03-29	0
2016-03-30	0
2016-03-31	0
2016-04-01	0
2016-04-02	0
2016-04-03	0
2016-04-04	0
2016-04-05	0
2016-04-06	0
2016-04-07	0
2016-04-08	0
2016-04-09	0
2016-04-10	0
2016-04-11	0
2016-04-12	0
2016-04-13	0
2016-04-14	0
2016-04-15	0
2016-04-16	0
2016-04-17	0
2016-04-18	0
2016-04-19	0
2016-04-20	0
2016-04-21	0
2016-04-22	0
2016-04-23	0
2016-04-24	0
2016-04-25	0
2016-04-26	0
2016-04-27	0
2016-04-28	0
2016-04-29	0
2016-04-30	0
2016-05-01	0
2016-05-02	0
2016-05-03	0
2016-05-04	0
2016-05-05	0
2016-05-06	0
2016-05-07	0
2016-05-08	0
2016-05-09	0
2016-05-10	0
2016-05-11	0
2016-05-12	0
2016-05-13	0
2016-05-14	0
2016-05-15	0

2016-05-16	0
2016-05-17	0
2016-05-18	0
2016-05-19	0
2016-05-20	0
2016-05-21	0
2016-05-22	0
2016-05-23	0
2016-05-24	0
2016-05-25	0
2016-05-26	0
2016-05-27	0
2016-05-28	0
2016-05-29	0
2016-05-30	0
2016-05-31	0
2016-06-01	0
2016-06-02	0
2016-06-03	0
2016-06-04	0
2016-06-05	0
2016-06-06	0
2016-06-07	0
2016-06-08	0
2016-06-09	0
2016-06-10	0
2016-06-11	0
2016-06-12	0
2016-06-13	0
2016-06-14	0
2016-06-15	0
2016-06-16	0
2016-06-17	0
2016-06-18	0
2016-06-19	0
2016-06-20	0
2016-06-21	0
2016-06-22	0
2016-06-23	0
2016-06-24	0
2016-06-25	0
2016-06-26	0
2016-06-27	0
2016-06-28	0
2016-06-29	0
2016-06-30	0
2016-07-01	0
2016-07-02	0
2016-07-03	0
2016-07-04	0
2016-07-05	0
2016-07-06	0
2016-07-07	0
2016-07-08	0
2016-07-09	0
2016-07-10	0
2016-07-11	0
2016-07-12	0
2016-07-13	0
2016-07-14	0
2016-07-15	0
2016-07-16	0
2016-07-17	0
2016-07-18	0

2016-07-19	0
2016-07-20	0
2016-07-21	0
2016-07-22	0
2016-07-23	0
2016-07-24	0
2016-07-25	0
2016-07-26	0
2016-07-27	0
2016-07-28	0
2016-07-29	0
2016-07-30	0
2016-07-31	0
2016-08-01	0
2016-08-02	0
2016-08-03	0
2016-08-04	0
2016-08-05	0
2016-08-06	0
2016-08-07	0
2016-08-08	0
2016-08-09	0
2016-08-10	0
2016-08-11	0
2016-08-12	0
2016-08-13	0
2016-08-14	0
2016-08-15	0
2016-08-16	0
2016-08-17	0
2016-08-18	0
2016-08-19	0
2016-08-20	0
2016-08-21	0
2016-08-22	0
2016-08-23	0
2016-08-24	0
2016-08-25	0
2016-08-26	0
2016-08-27	0
2016-08-28	0
2016-08-29	0
2016-08-30	0
2016-08-31	0
2016-09-01	0
2016-09-02	0
2016-09-03	0
2016-09-04	0
2016-09-05	0
2016-09-06	0
2016-09-07	0
2016-09-08	0
2016-09-09	0
2016-09-10	0
2016-09-11	0
2016-09-12	0
2016-09-13	0
2016-09-14	0
2016-09-15	0
2016-09-16	0
2016-09-17	0
2016-09-18	0
2016-09-19	0
2016-09-20	0

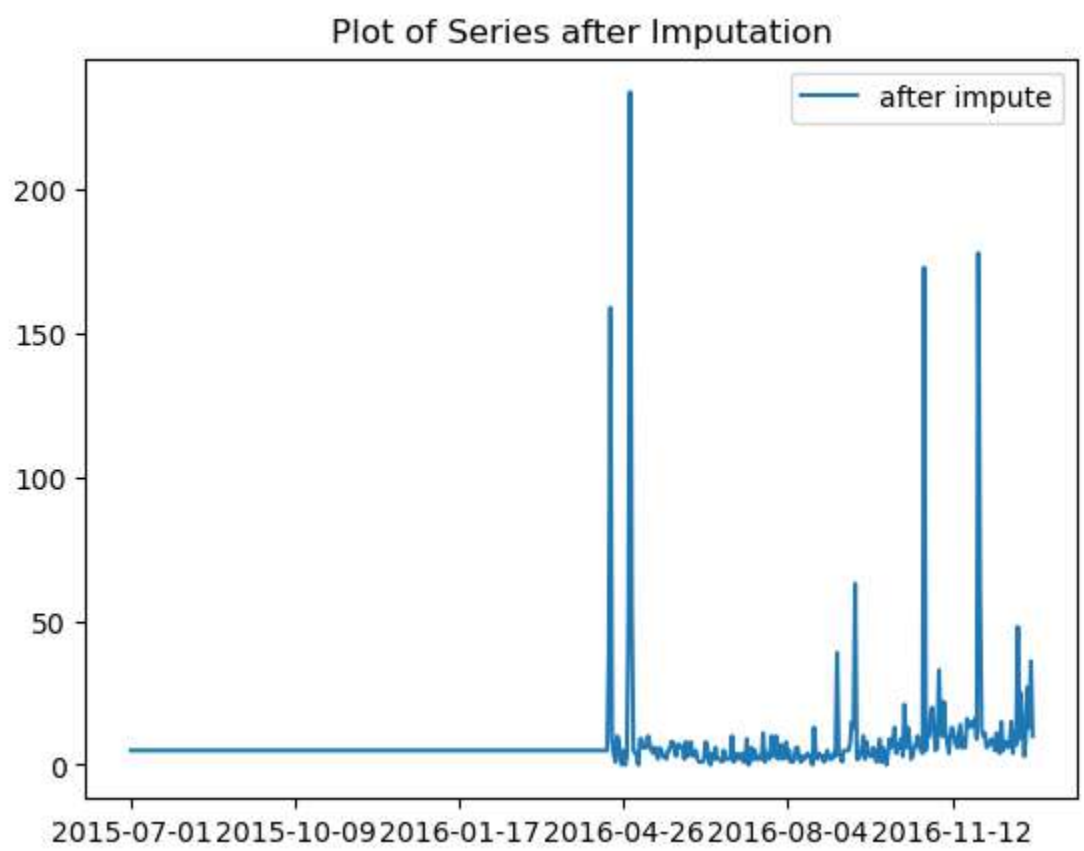
2016-09-21	0
2016-09-22	0
2016-09-23	0
2016-09-24	0
2016-09-25	0
2016-09-26	0
2016-09-27	0
2016-09-28	0
2016-09-29	0
2016-09-30	0
2016-10-01	0
2016-10-02	0
2016-10-03	0
2016-10-04	0
2016-10-05	0
2016-10-06	0
2016-10-07	0
2016-10-08	0
2016-10-09	0
2016-10-10	0
2016-10-11	0
2016-10-12	0
2016-10-13	0
2016-10-14	0
2016-10-15	0
2016-10-16	0
2016-10-17	0
2016-10-18	0
2016-10-19	0
2016-10-20	0
2016-10-21	0
2016-10-22	0
2016-10-23	0
2016-10-24	0
2016-10-25	0
2016-10-26	0
2016-10-27	0
2016-10-28	0
2016-10-29	0
2016-10-30	0
2016-10-31	0
2016-11-01	0
2016-11-02	0
2016-11-03	0
2016-11-04	0
2016-11-05	0
2016-11-06	0
2016-11-07	0
2016-11-08	0
2016-11-09	0
2016-11-10	0
2016-11-11	0
2016-11-12	0
2016-11-13	0
2016-11-14	0
2016-11-15	0
2016-11-16	0
2016-11-17	0
2016-11-18	0
2016-11-19	0
2016-11-20	0
2016-11-21	0
2016-11-22	0
2016-11-23	0

```
2016-11-24    0
2016-11-25    0
2016-11-26    0
2016-11-27    0
2016-11-28    0
2016-11-29    0
2016-11-30    0
2016-12-01    0
2016-12-02    0
2016-12-03    0
2016-12-04    0
2016-12-05    0
2016-12-06    0
2016-12-07    0
2016-12-08    0
2016-12-09    0
2016-12-10    0
2016-12-11    0
2016-12-12    0
2016-12-13    0
2016-12-14    0
2016-12-15    0
2016-12-16    0
2016-12-17    0
2016-12-18    0
2016-12-19    0
2016-12-20    0
2016-12-21    0
2016-12-22    0
2016-12-23    0
2016-12-24    0
2016-12-25    0
2016-12-26    0
2016-12-27    0
2016-12-28    0
2016-12-29    0
2016-12-30    0
2016-12-31    0
dtype: int64
```

Data Frame after imputation

In [29]:

```
plt.title('Plot of Series after Imputation')
df.iloc[4].plot(label= 'after impute')
plt.legend()
plt.show()
```



In [30]:
```
set(exolang)
```

Out[30]:
```
{'NA', '_de', 'de', 'en', 'es', 'fr', 'ja', 'ru', 'zh'}
```

EDA On the basis of Trend of language

In [31]:
```
df_language = pd.DataFrame(exolang.value_counts().reset_index())
```

In [32]:
```
df_language
```

Out[32]:

	index	Language
0	en	24010
1	ja	20340
2	de	18437
3	fr	17761
4	NA	17728
5	zh	17103
6	ru	14990
7	es	14041
8	_de	1

Full form of all language code

_de: German (This is typically used as a locale specifier, e.g., for formatting purposes.)

de: German

en: English

es: Spanish

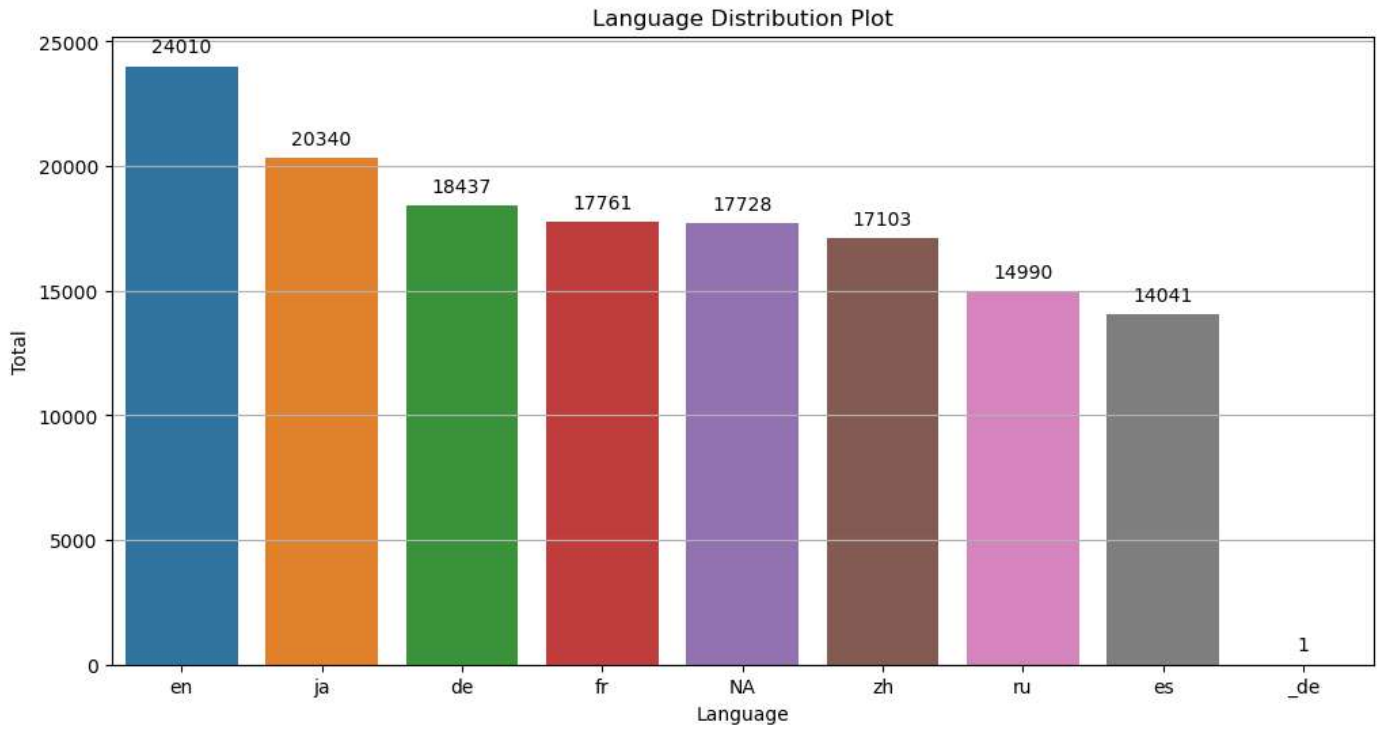
fr: French

ja: Japanese

ru: Russian

zh: Chinese

In [33]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Your dataframe
# df_language = ...
plt.figure(figsize=(12,6))
plt.grid()
plt.title('Language Distribution Plot')

# Create the bar plot
ax = sns.barplot(x=df_language['index'], y=df_language['Language'])

# Annotate each bar with its value
for p in ax.patches:
    ax.annotate(f"{p.get_height():.0f}",
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points')

plt.xlabel('Language')
plt.ylabel('Total')
plt.show()
```



Observation

1. Pages with english is more as compare to other.
2. Pages with Spanish is less.

In [34]:
```
df['language'] = exolang
```

Converting data to its General Form

In [35]:
```
df.head()
```

Out[35]:

	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	2015-07-14	2015-07-15	2015-07-16
0	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	15.0	8.0	16.0
1	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	38.0	20.0	62.0
2	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	6.0	8.0	6.0
3	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	145.0	14.0	17.0
4	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0

In [36]:
```
set(df['language'])
```

Out[36]:
```
{'NA', '_de', 'de', 'en', 'es', 'fr', 'ja', 'ru', 'zh'}
```

In [37]:
```
en_df = df.loc[(df['language']=='en')==True]
```

In [38]:
```
mean_val_df =  df.pivot_table(index='language', aggfunc='mean')
```

In [39]:
```
mean_val_df.head()
```

Out[39]:

language	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07
NA	124.284493	128.301077	123.393474	111.330127	119.047298	130.555252	135.840366	139.953
_de	11.000000	192.000000	192.000000	80.000000	72.000000	29.000000	44.000000	101.000
de	748.269078	738.514997	709.930981	653.911157	755.402831	828.627678	804.455823	815.614
en	3699.986672	3688.549667	3509.763578	3647.536381	3761.088692	4035.626885	3829.172886	3621.479
es	1124.820419	1076.750516	993.822199	934.806317	1013.400292	1149.757318	1122.161776	1090.163

In [40]:
```
df = mean_val_df.T
```

In [41]:
```
df.head()
```

Out[41]:

language	NA	_de	de	en	es	fr	ja	ru	
2015-07-01	124.284493	11.0	748.269078	3699.986672	1124.820419	518.275576	623.080703	691.852568	300.511
2015-07-02	128.301077	192.0	738.514997	3688.549667	1076.750516	521.232645	709.462561	702.874850	300.816
2015-07-03	123.393474	192.0	709.930981	3509.763578	993.822199	503.030516	644.695993	655.785690	298.917
2015-07-04	111.330127	80.0	653.911157	3647.536381	934.806317	534.642391	799.631563	620.469613	301.842
2015-07-05	119.047298	72.0	755.402831	3761.088692	1013.400292	525.623191	768.694764	656.572081	317.261

In [42]:
```
df.columns.name = None
```

In [43]:
```
df.reset_index(inplace=True)
```

In [44]:
```
#df.to_csv('AdEase_gen_data.csv',index=False)
```

In [45]:
```
df.rename(index={'index':'dates'},inplace=True)
```

In [46]:
```
#df.columns.name = 'dates'
```

In [47]:
```
df.set_index('index',inplace=True)
```

In [48]:
```
df.head()
```

Out[48]:

index	NA	_de	de	en	es	fr	ja	ru	zh
2015-07-01	124.284493	11.0	748.269078	3699.986672	1124.820419	518.275576	623.080703	691.852568	300.511928
2015-07-02	128.301077	192.0	738.514997	3688.549667	1076.750516	521.232645	709.462561	702.874850	300.816903
2015-07-03	123.393474	192.0	709.930981	3509.763578	993.822199	503.030516	644.695993	655.785690	298.917792
2015-07-04	111.330127	80.0	653.911157	3647.536381	934.806317	534.642391	799.631563	620.469613	301.842952
2015-07-05	119.047298	72.0	755.402831	3761.088692	1013.400292	525.623191	768.694764	656.572081	317.261095

In [49]:
```
#df.to_csv('AdEase_gen_data.csv')
```

In [50]:
```
df = pd.DataFrame(df)
```

In [51]:
```
df.head()
```

Out[51]:

index	NA	_de	de	en	es	fr	ja	ru	zh
2015-07-01	124.284493	11.0	748.269078	3699.986672	1124.820419	518.275576	623.080703	691.852568	300.511928
2015-07-02	128.301077	192.0	738.514997	3688.549667	1076.750516	521.232645	709.462561	702.874850	300.816903
2015-07-03	123.393474	192.0	709.930981	3509.763578	993.822199	503.030516	644.695993	655.785690	298.917792
2015-07-04	111.330127	80.0	653.911157	3647.536381	934.806317	534.642391	799.631563	620.469613	301.842952
2015-07-05	119.047298	72.0	755.402831	3761.088692	1013.400292	525.623191	768.694764	656.572081	317.261095

In [52]:
```
df.columns
```

Out[52]:
```
Index(['NA', '_de', 'de', 'en', 'es', 'fr', 'ja', 'ru', 'zh'], dtype='object')
```

Pageview Trend of Different Languages

In [53]:
```
list(df.columns)
```

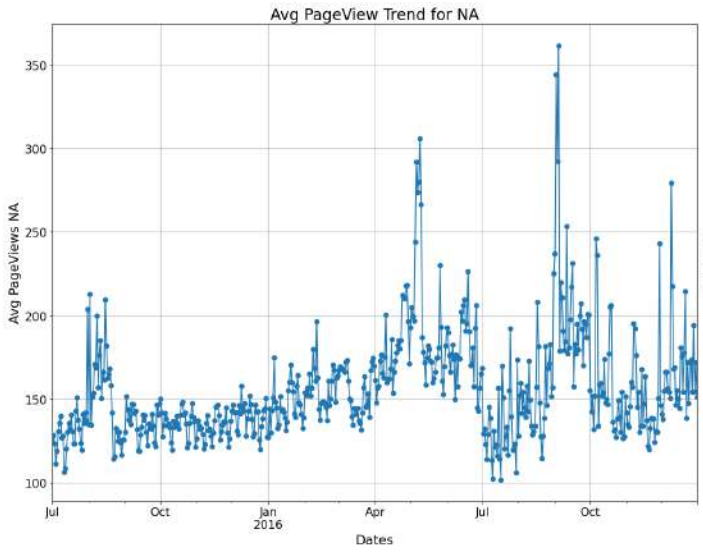
Out[53]:
```
['NA', '_de', 'de', 'en', 'es', 'fr', 'ja', 'ru', 'zh']
```

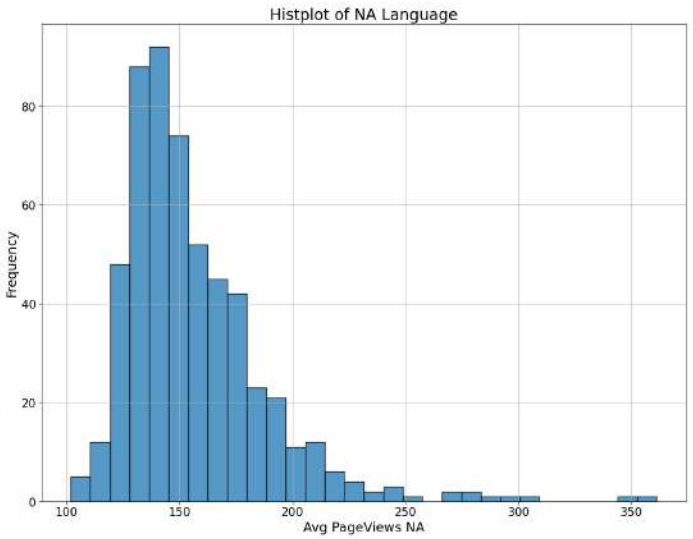
In [54]:
```
df.index = pd.to_datetime(df.index)
```

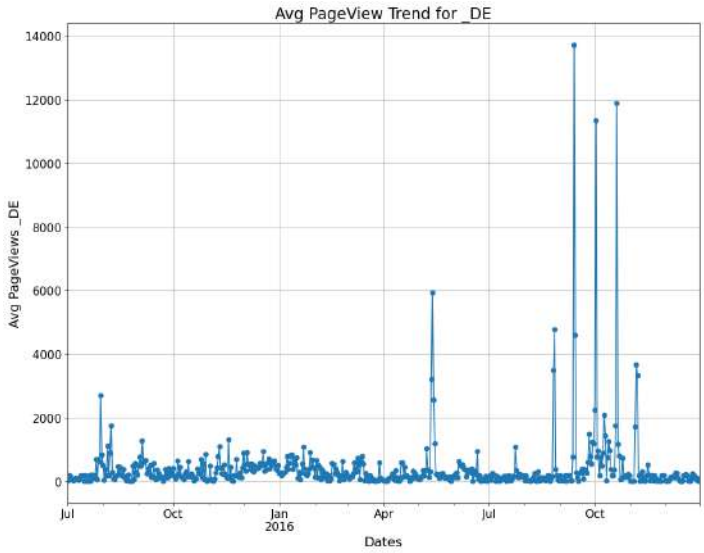
In [55]:
```
import matplotlib.pyplot as plt
import seaborn as sns

def plot_trend_lines(vals=list(df.columns)):
    """
    This function will plot both Trend Lines and Histograms for different languages.
    """
    for i in vals:
        # Create a new figure with 1 row and 2 columns
        fig, axes = plt.subplots(1, 2, figsize=(25, 10))

        # Plot the line plot on the first subplot
        df[i].plot(ax=axes[0], style='-o', label='original')
        axes[0].set_title(f'Avg PageView Trend for {i.upper()}', fontsize=20)
        axes[0].set_xlabel('Dates', fontsize=17)
        axes[0].set_ylabel(f'Avg PageViews {i.upper()}', fontsize=17)
        axes[0].tick_params(axis='both', labelsize=15)
        axes[0].grid(True)

        # Plot the histogram on the second subplot
        sns.histplot(df[i], ax=axes[1])
        axes[1].set_title(f'Histplot of {i.upper()} Language', fontsize=20)
        axes[1].set_xlabel(f'Avg PageViews {i.upper()}', fontsize=17)
        axes[1].set_ylabel('Frequency', fontsize=17)
        axes[1].tick_params(axis='both', labelsize=15)
        axes[1].grid(True)

        # Adjust layout for better spacing
        plt.tight_layout()

        # Show the plot
```

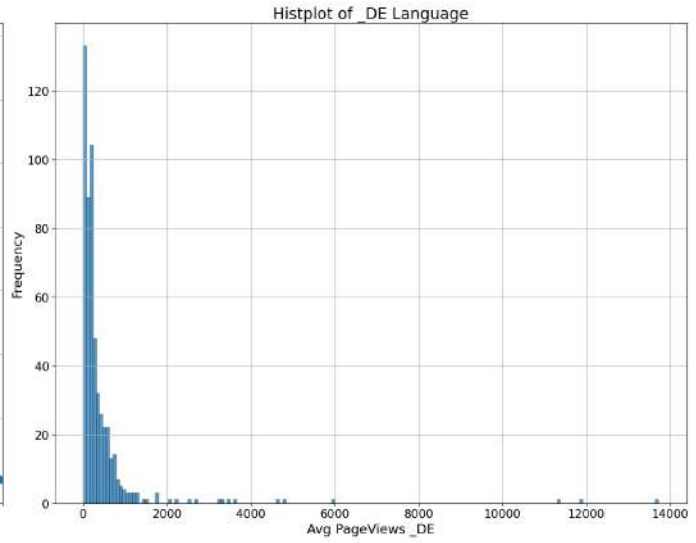
```
        plt.show()
        print('\n')
```

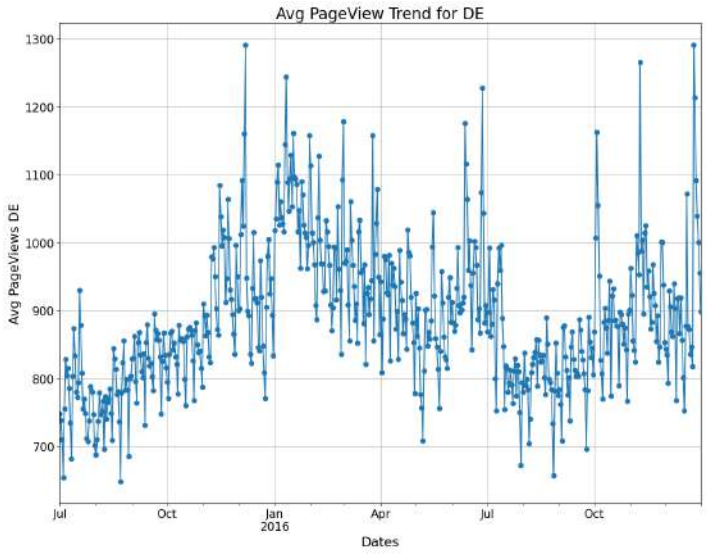
In [56]:
```
plot_trend_lines()
```

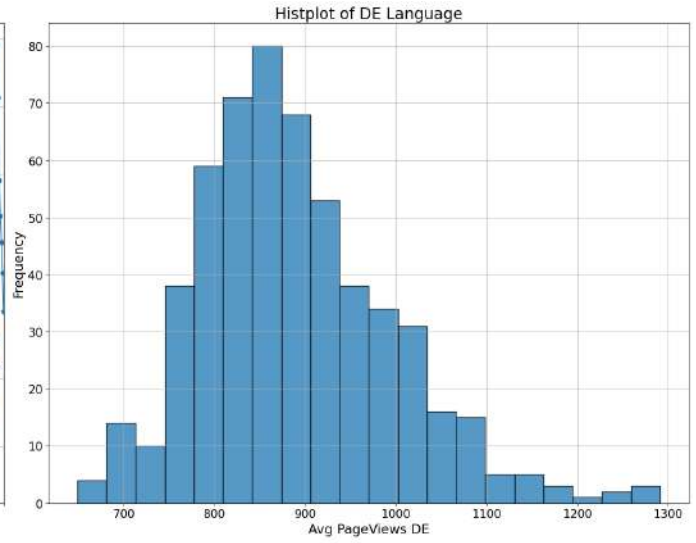


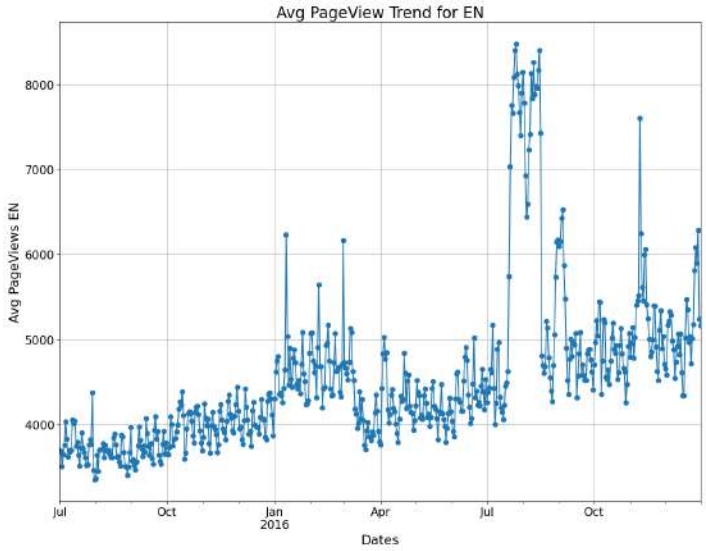


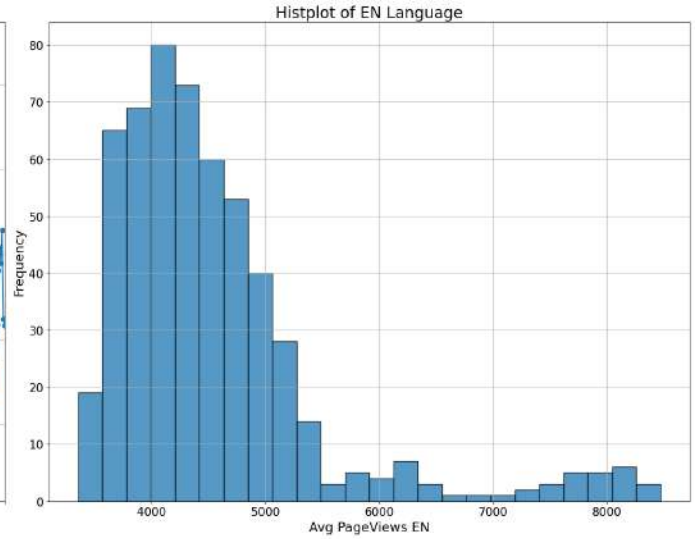


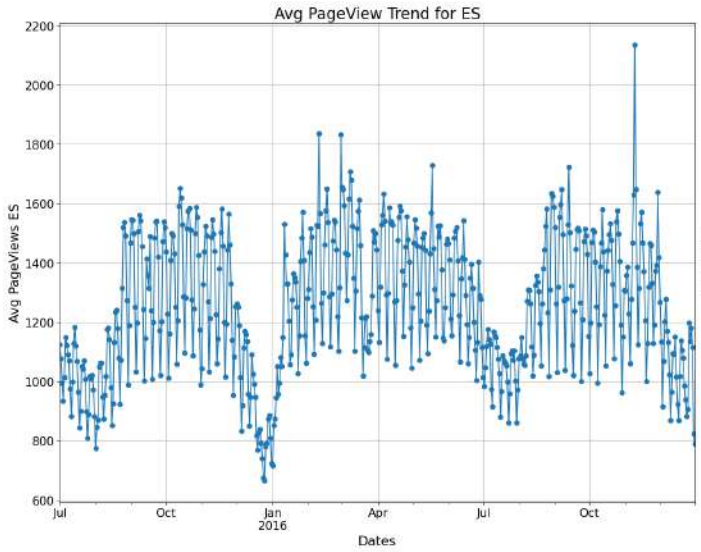


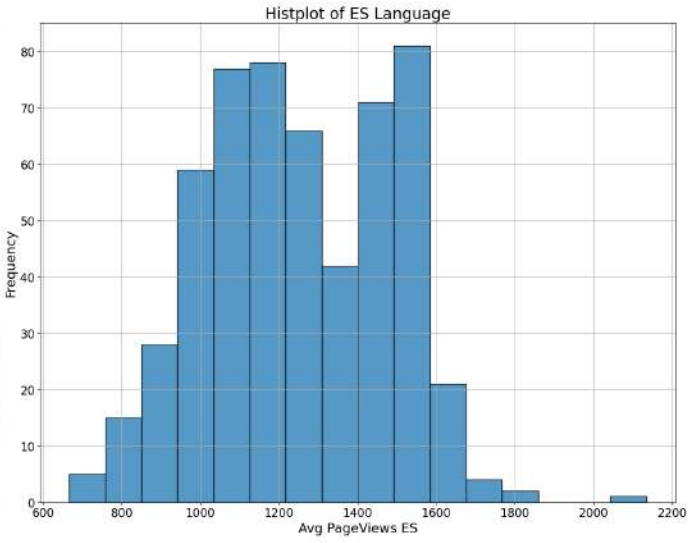


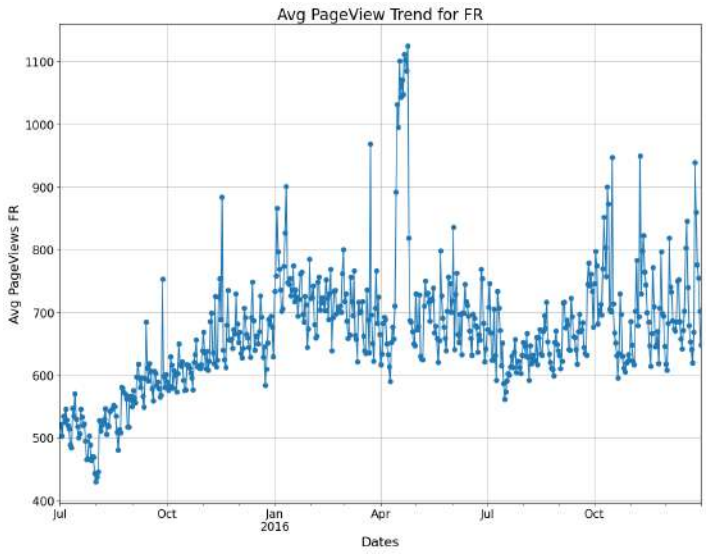


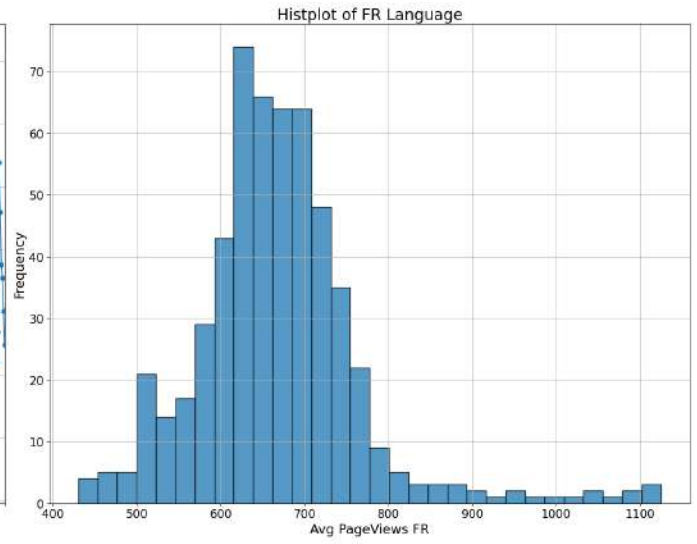


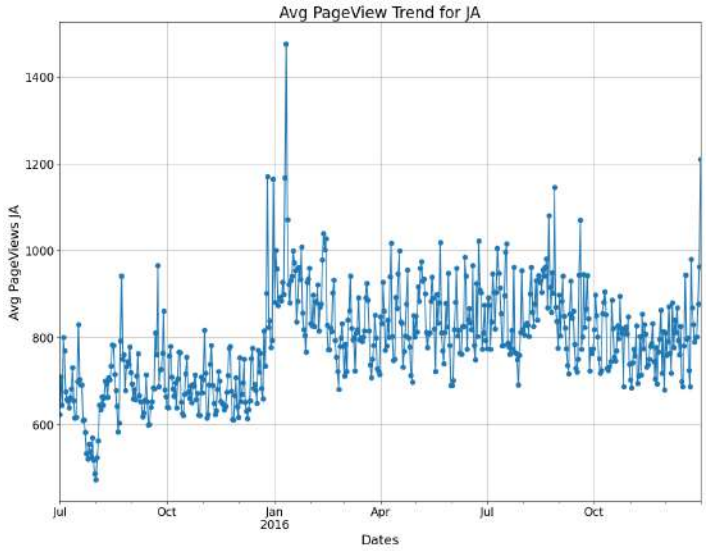


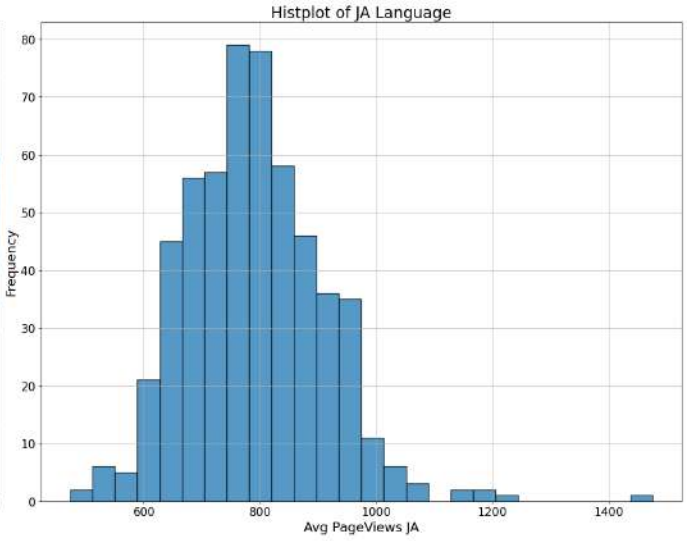


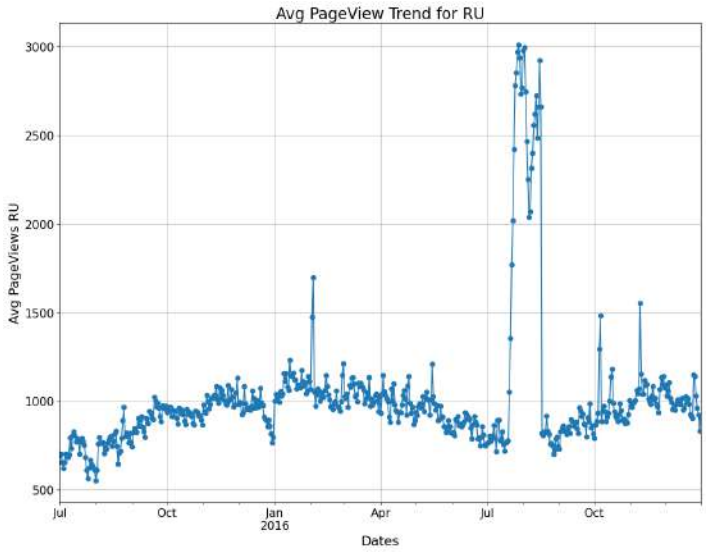


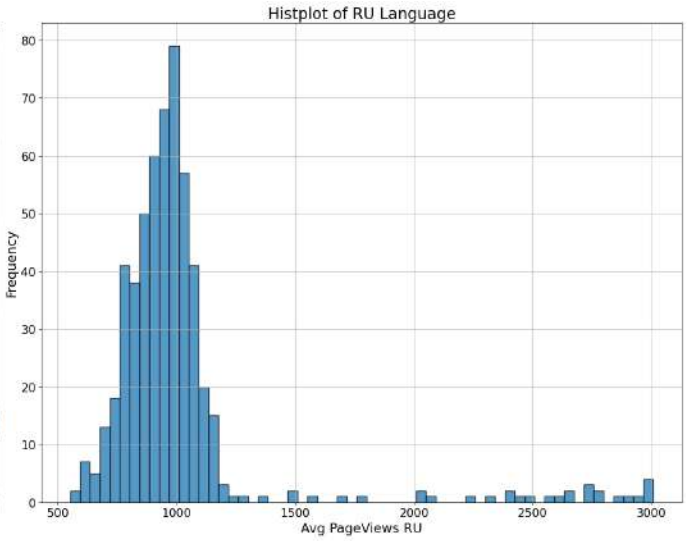


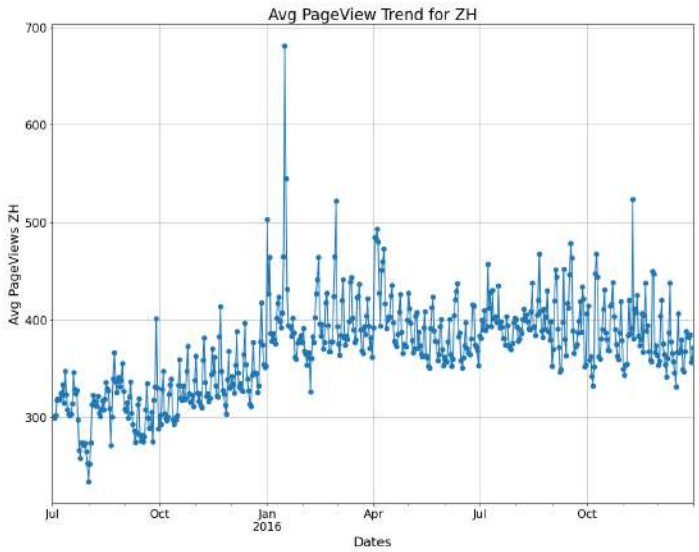


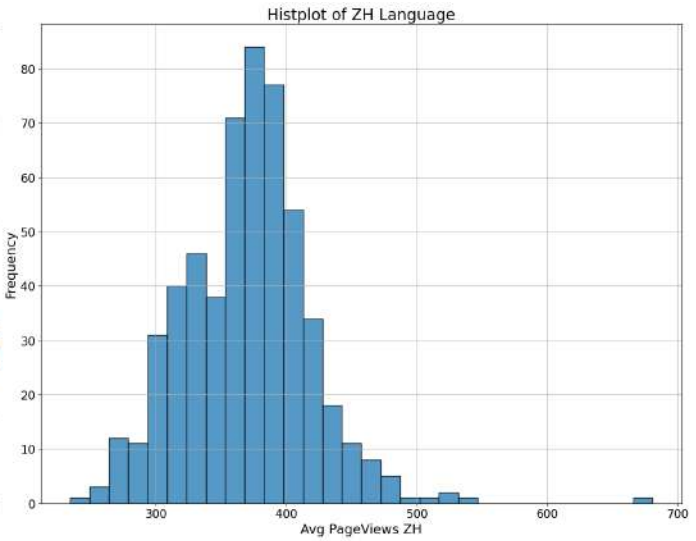










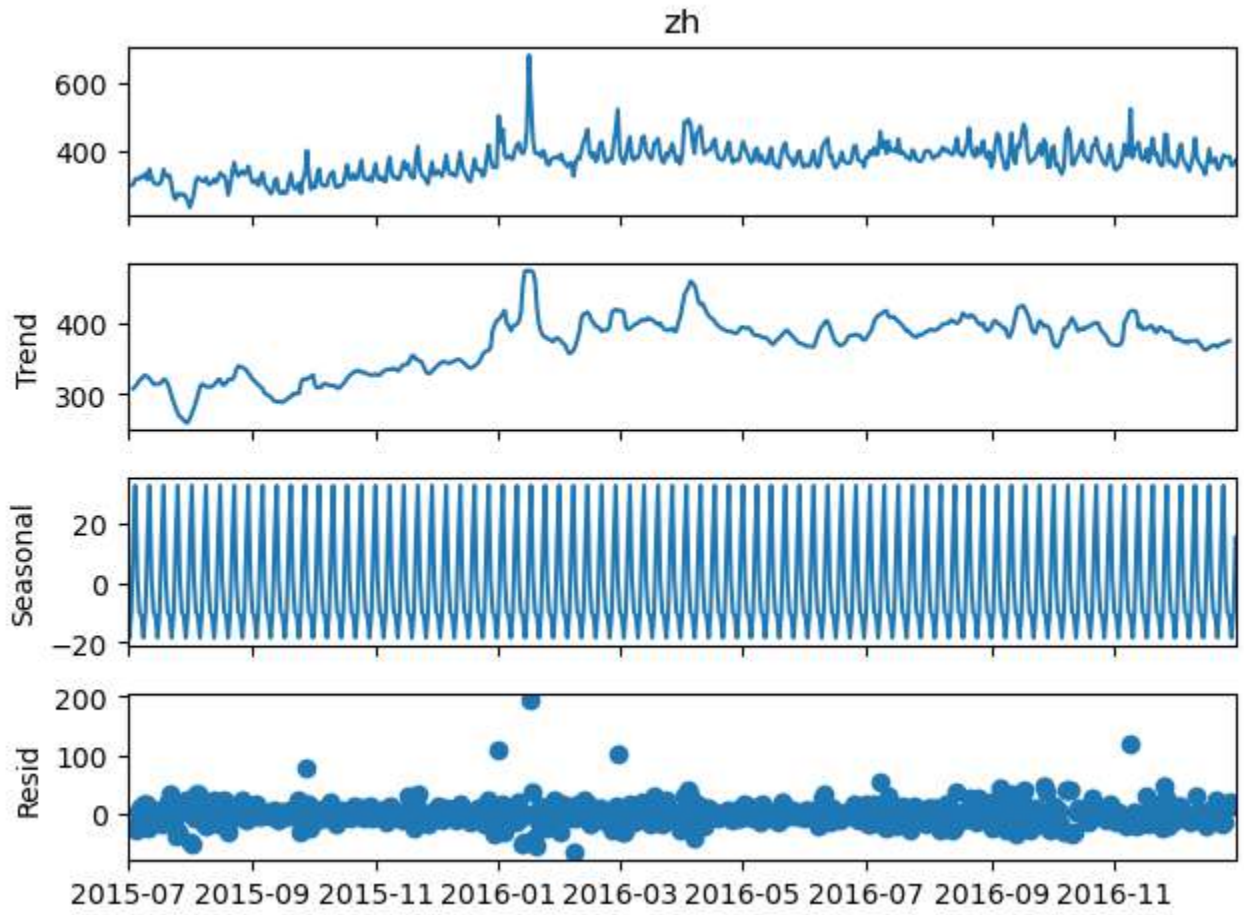


Trend and Seasonality Check for Each Trends of PageViews

`Trend and Seasonality Check for Chinese(ZH) Language Pageviews`

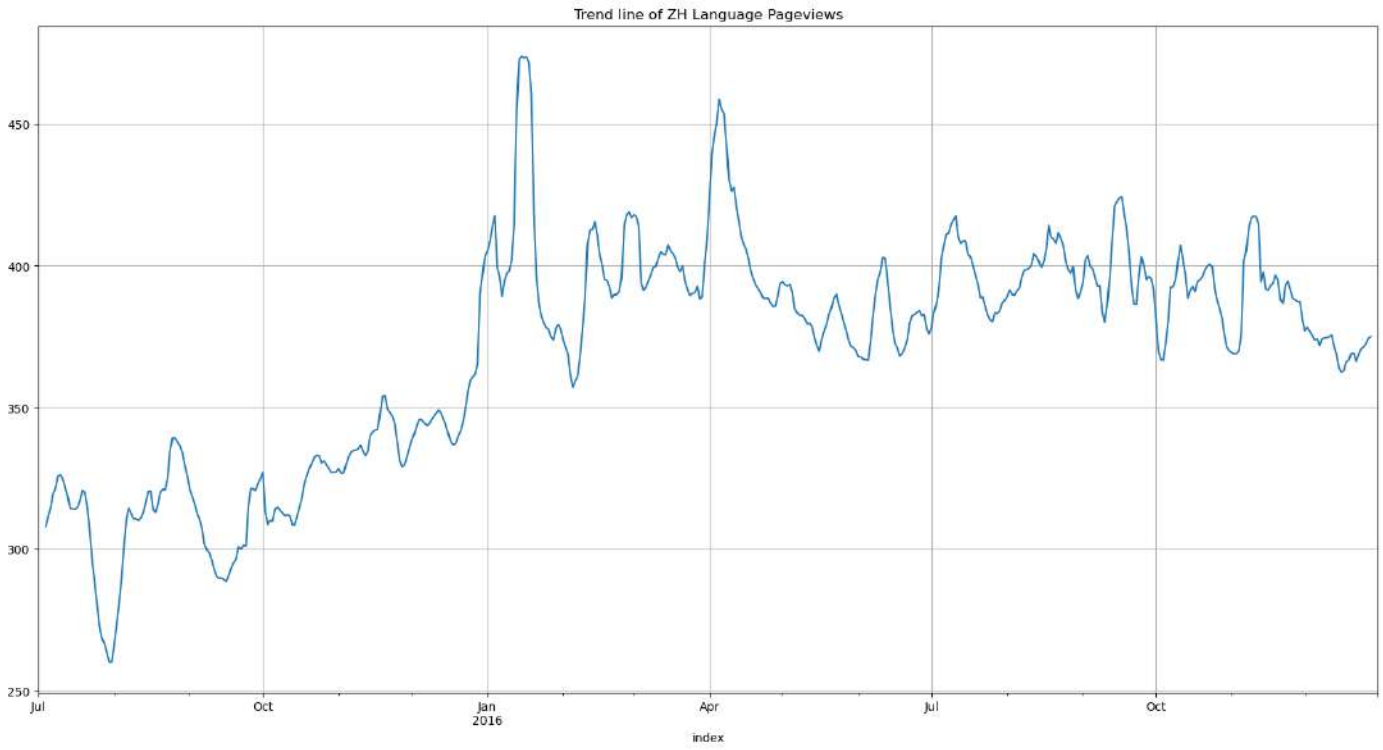
In [57]:
```
model_zh = sm.tsa.seasonal_decompose(df.zh)
```

In [58]:
```
model_zh.plot()
plt.show()
```

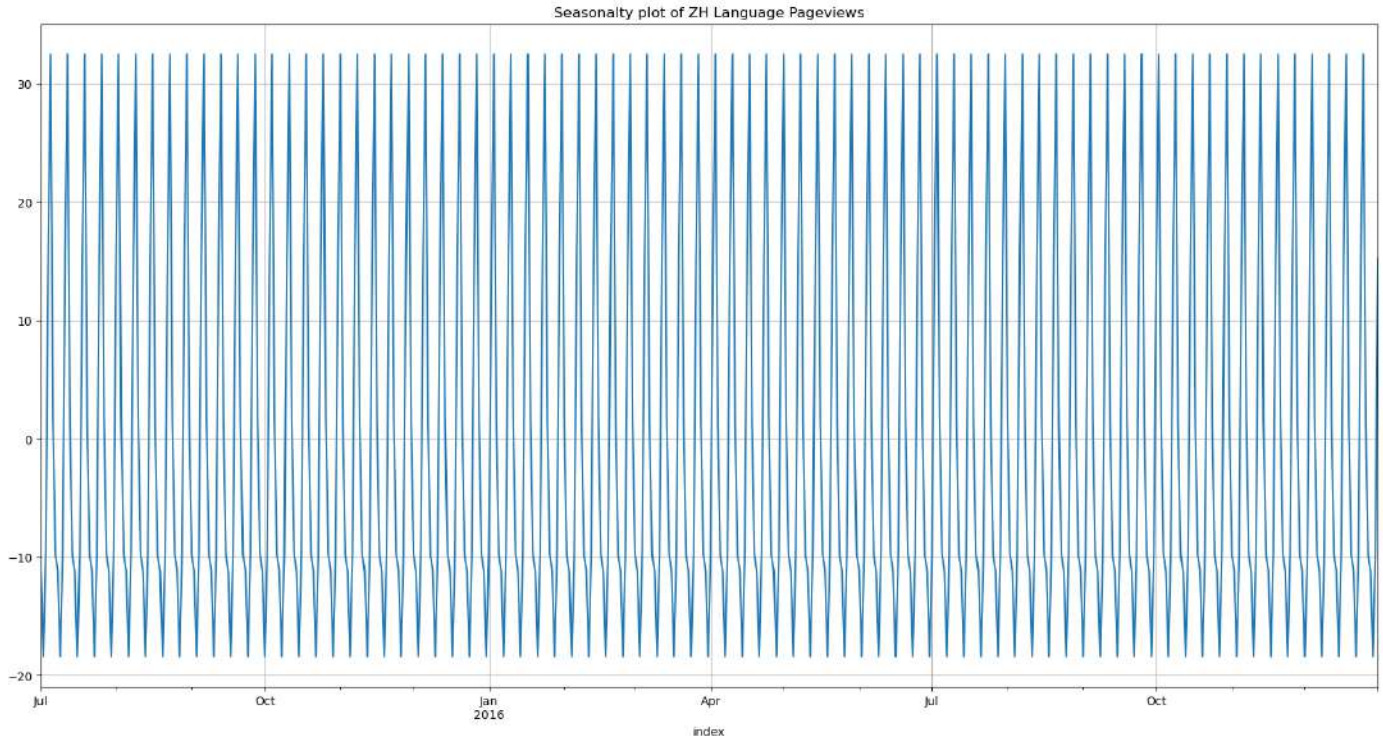


In [59]:
```
def get_insight_plot(vals,window,ln):
    """
    This Function will print Original plot as well as Rolling Window plot
    """
    plt.figure(figsize=(25,10))
    vals.plot(style='-o',label='original')
    vals.rolling(window=window,center=False).mean().plot(style='-o',label='rolling')
    vals.rolling(window=window,center=True).mean().plot(style='-o',label='rolling center
    plt.xlabel('Dates',fontsize=17)
    plt.ylabel(f'Original Vs Windowed plot for {ln}',fontsize=17)
    plt.legend()
    plt.xticks(fontsize=15)  # Adjust fontsize as needed
    plt.yticks(fontsize=15)
    plt.grid()
    plt.title(f'Avg PageView Trend for {ln}',fontsize=20)
    plt.show()
    print('\n')
```

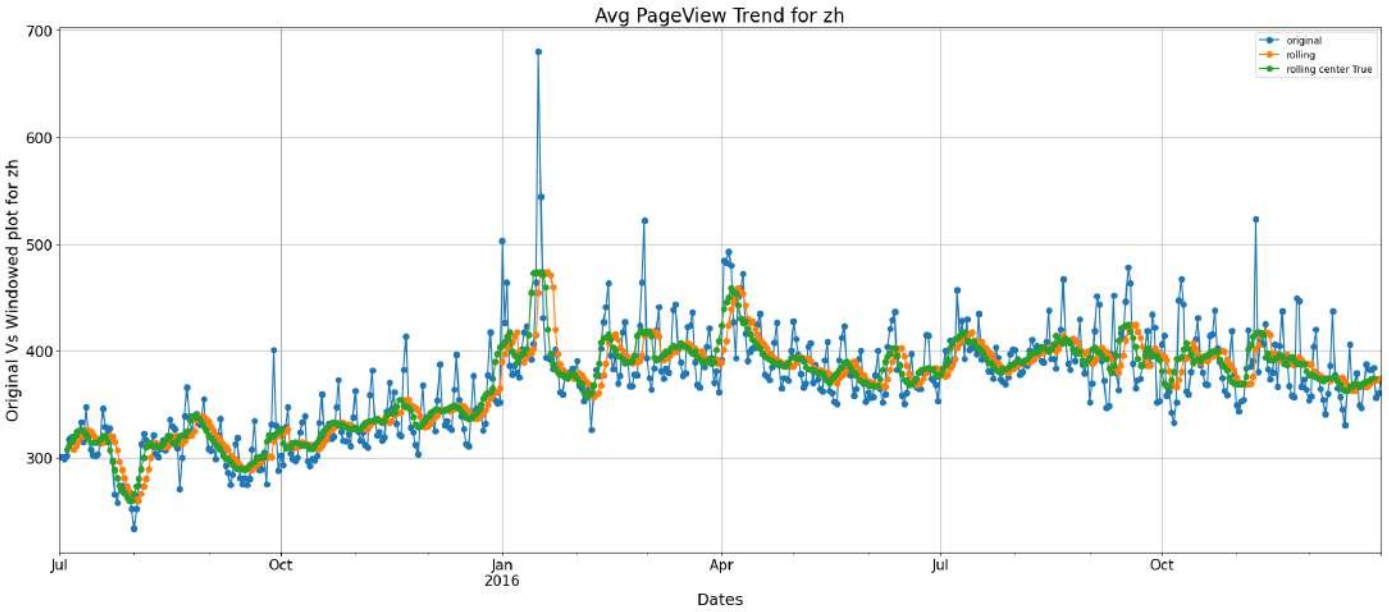
In [60]:
```
plt.title('Trend line of ZH Language Pageviews')
model_zh.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [61]:
```
plt.title('Seasonalty plot of ZH Language Pageviews')
model_zh.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [62]:
```
get_insight_plot(df['zh'],window=7,ln='zh')
```



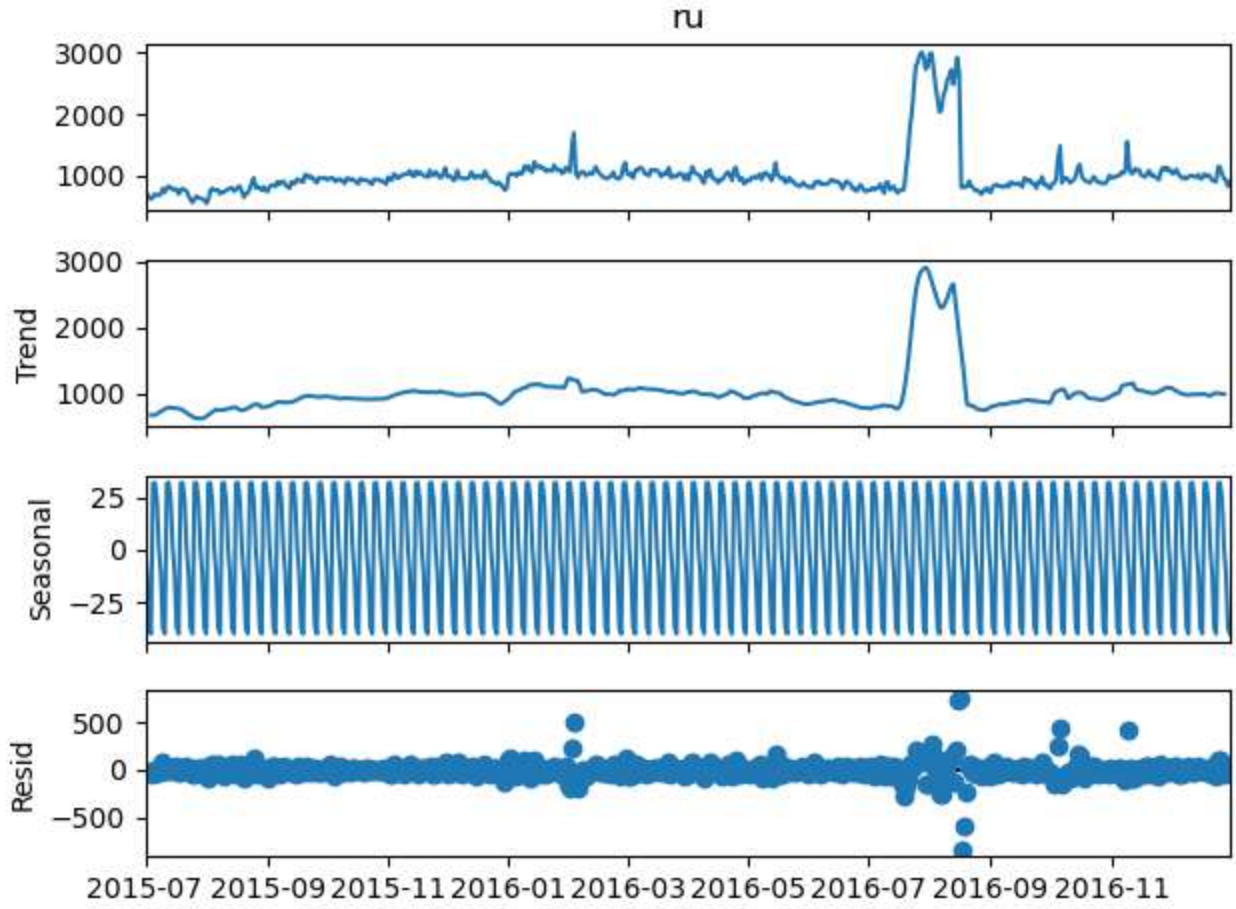
Observation

1. Trend is showing upwards direction flow this indicates that with moving forward dates the view may increase.
2. Seasonality shows that it is repeating in every `8` months for this language.

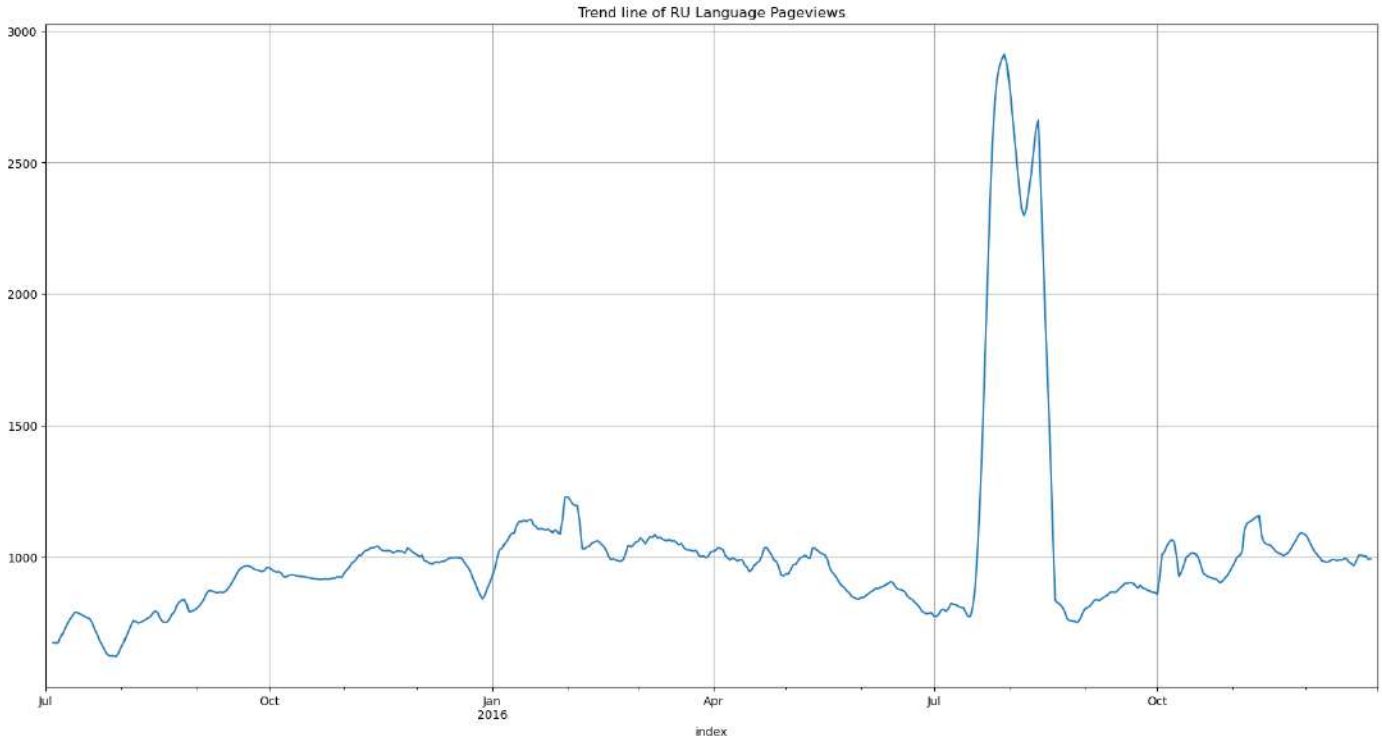
`Trend and Seasonality Check for Russian(RU) Language Pageviews`

In [63]:
```
model_ru = sm.tsa.seasonal_decompose(df.ru)
```

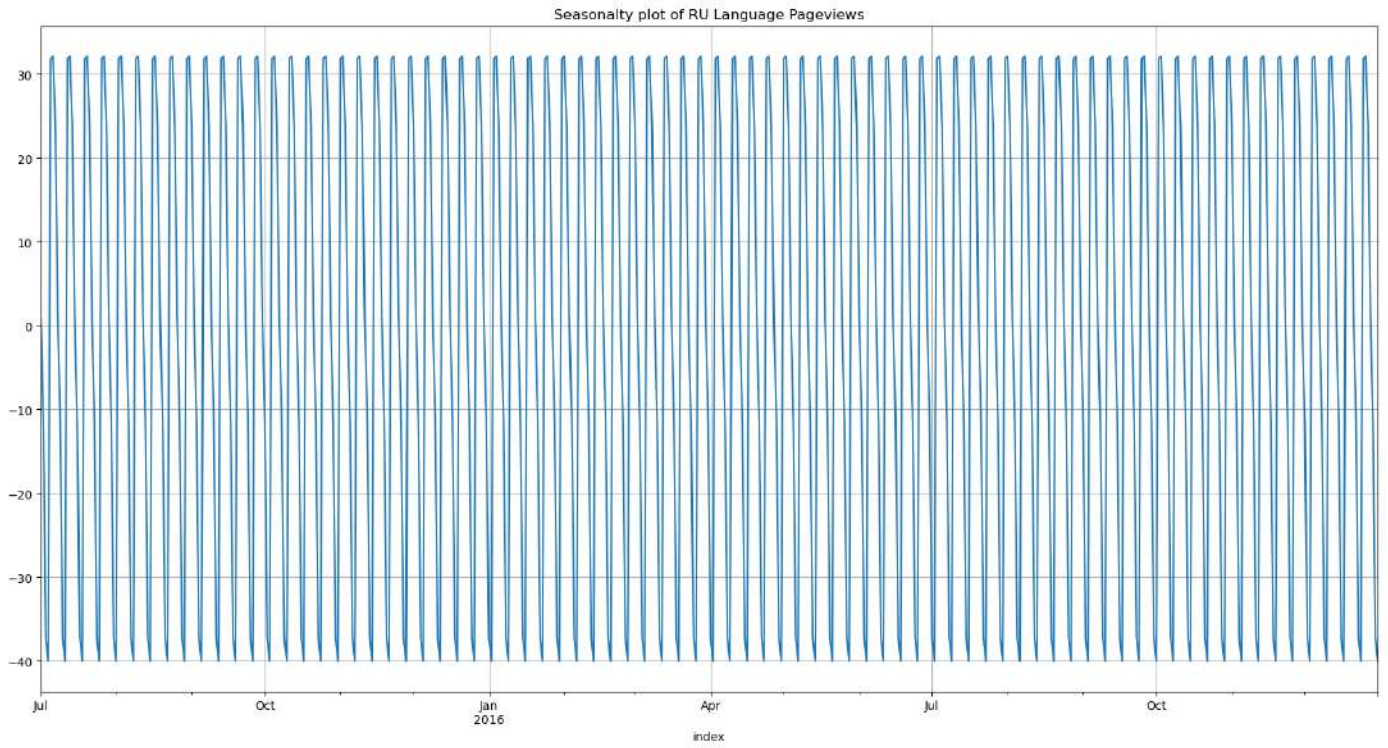
In [64]:
```
model_ru.plot()
plt.show()
```



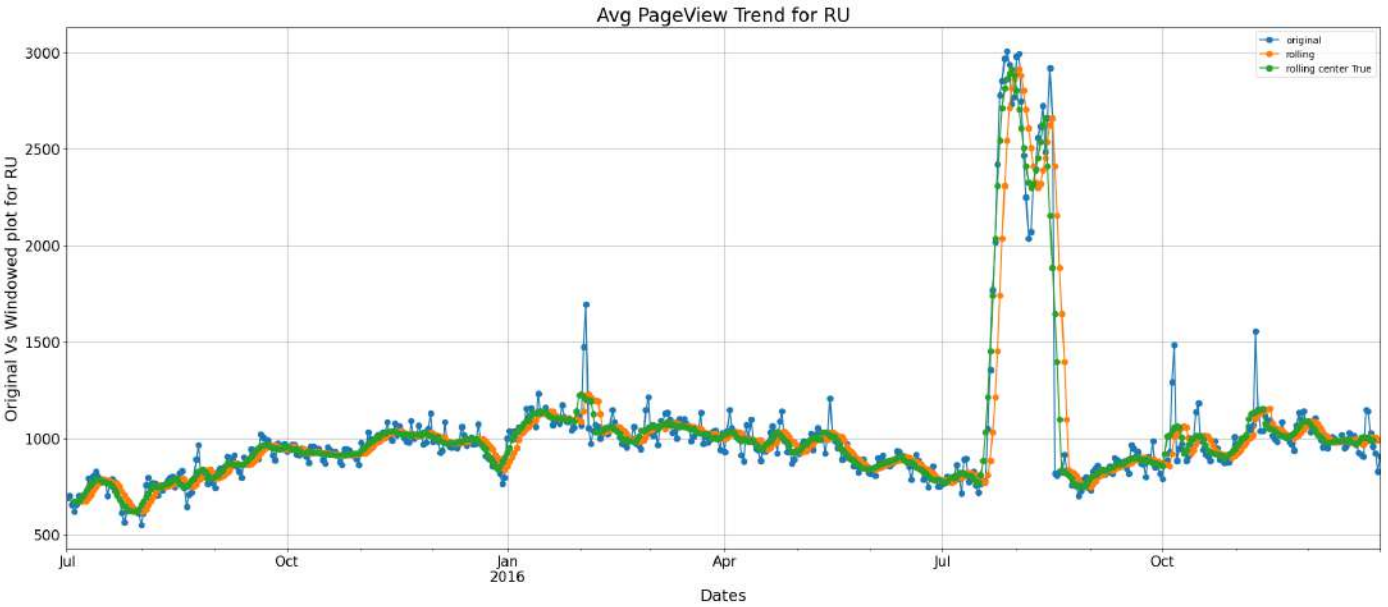
In [65]:
```
plt.title('Trend line of RU Language Pageviews')
model_ru.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [66]:
```
plt.title('Seasonalty plot of RU Language Pageviews')
model_ru.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [67]:
```
get_insight_plot(df['ru'],window=7,ln='RU')
```



Observation

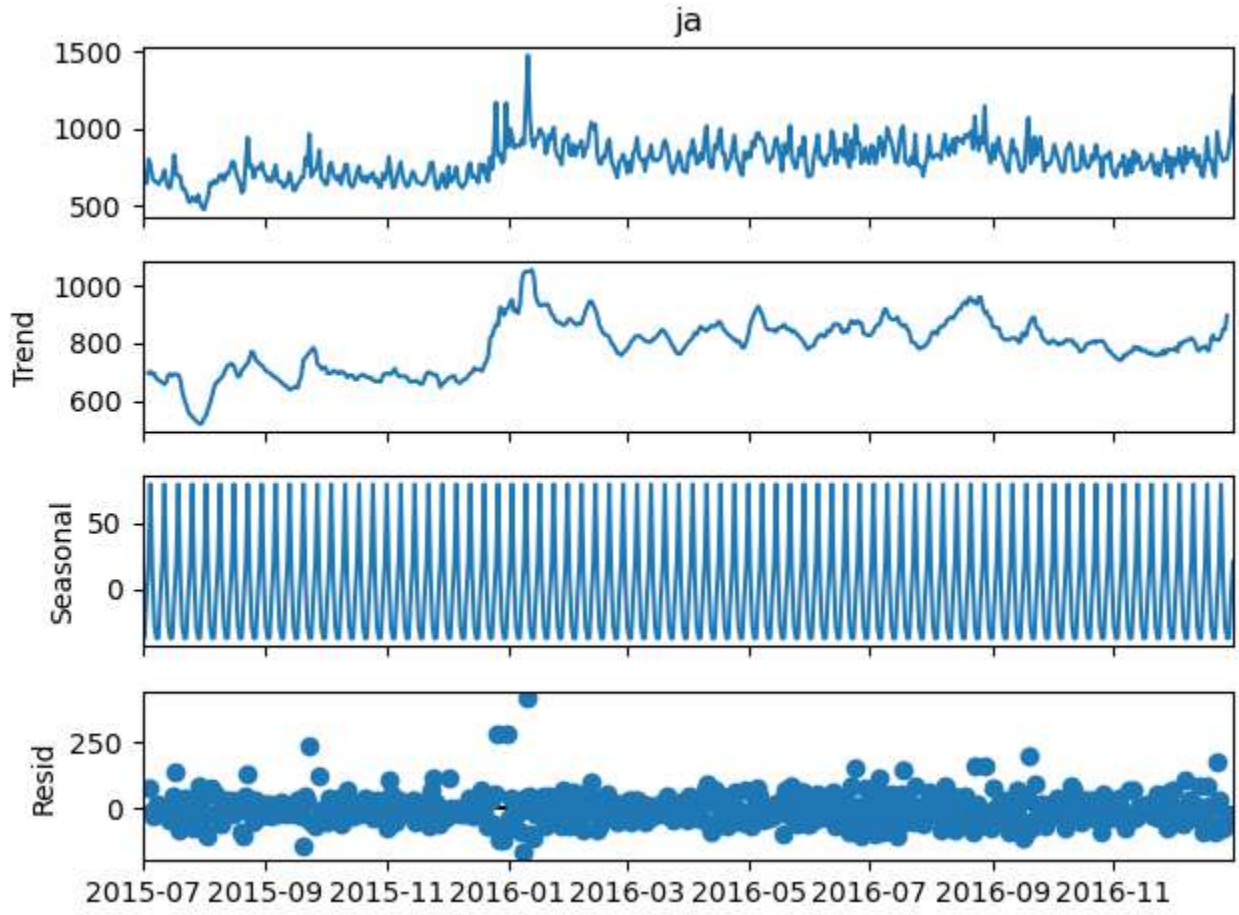
1. Trend is showing slightly downwards direction flow this indicates that with moving forward dates the pageviews may decrease.
2. Seasonality shows that it is repeating in every `8` months for this language.

`Trend and Seasonality Check for Japanese(JA) Language Pageviews`

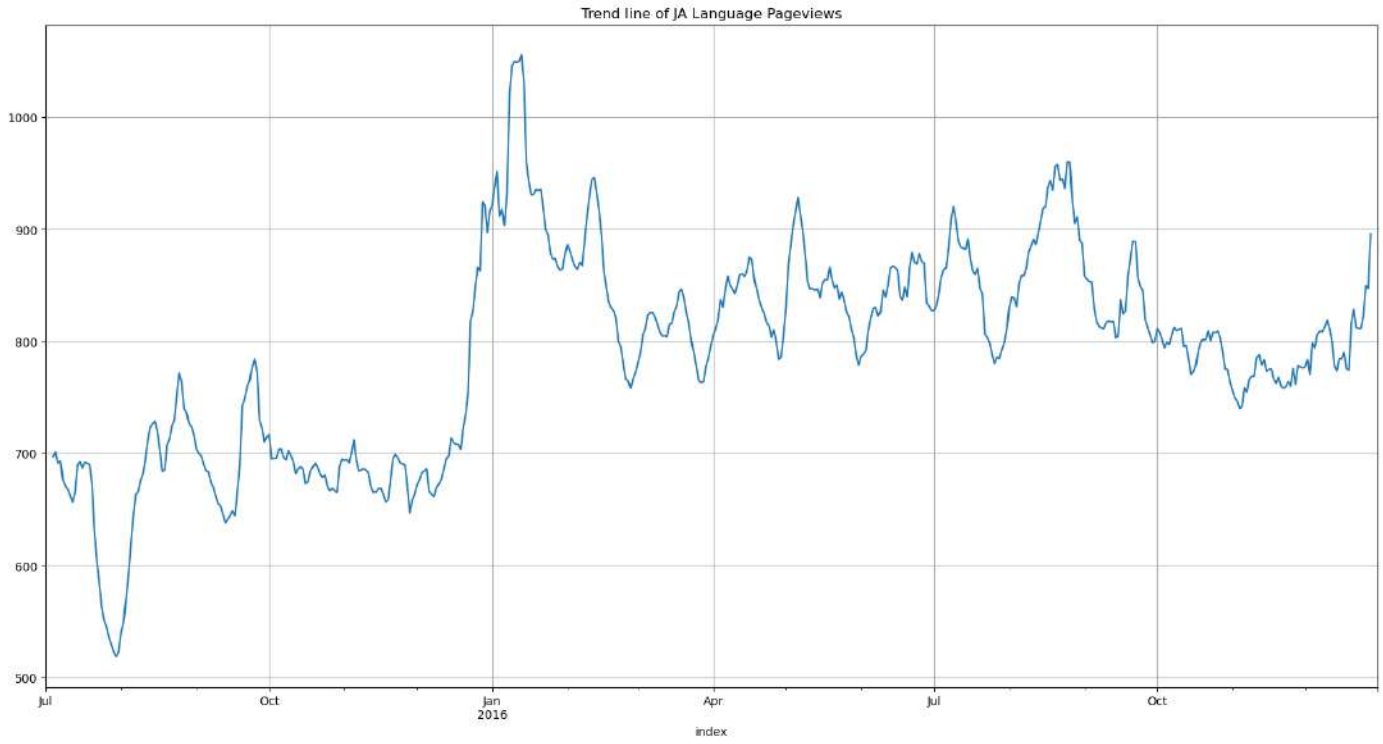
In [68]:
```
model_ja = sm.tsa.seasonal_decompose(df.ja)
```

In [69]:
```
model_ja.plot()
```

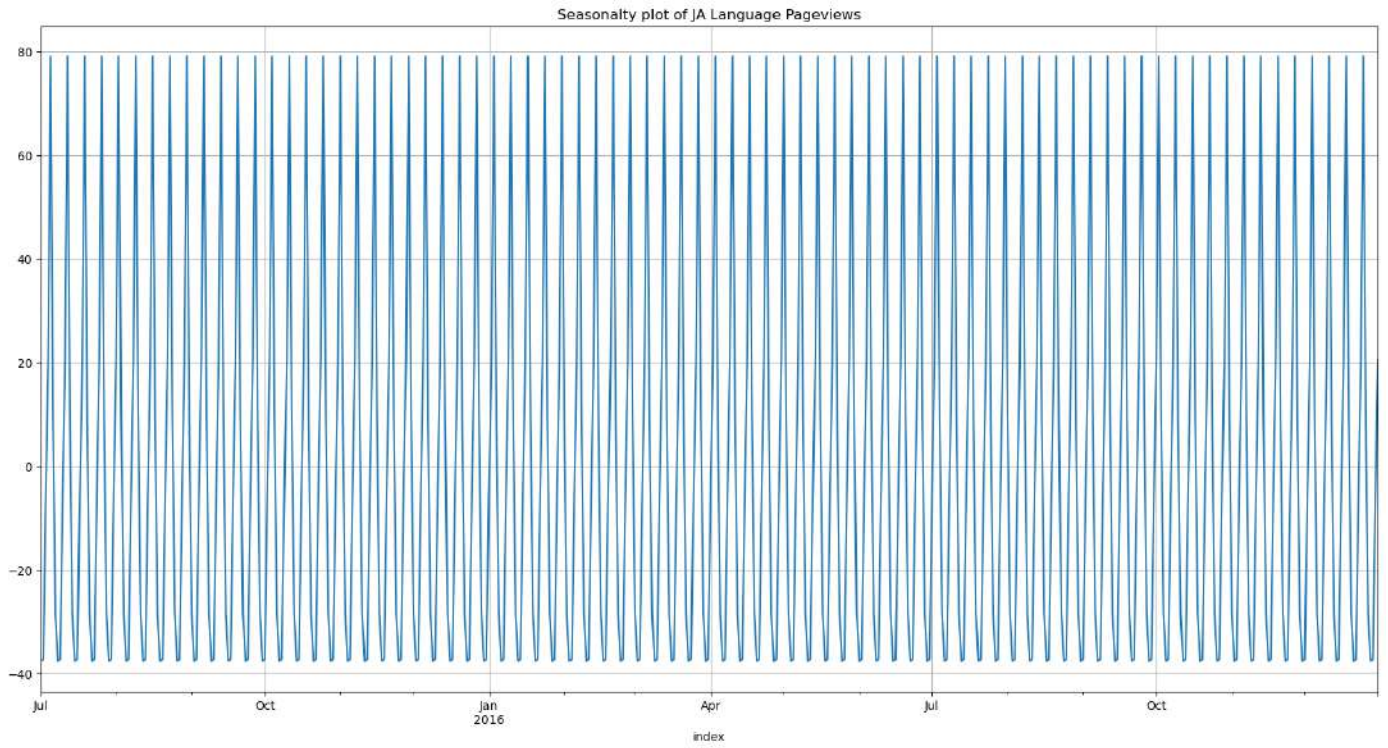
```
plt.show()
```



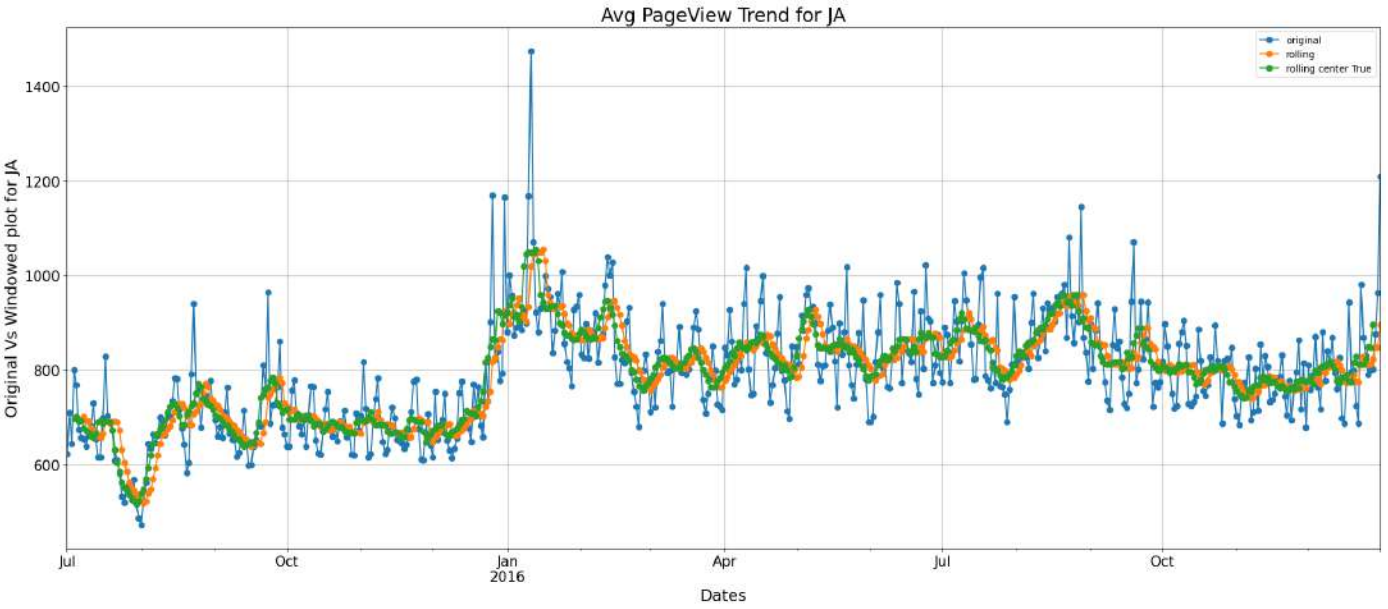
In [70]:
```
plt.title('Trend line of JA Language Pageviews')
model_ja.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [71]:
```
plt.title('Seasonalty plot of JA Language Pageviews')
model_ja.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [72]:
```
get_insight_plot(df['ja'],window=7,ln='JA')
```



Observation

1. Trend is showing upwards direction flow this indicates that with moving forward dates the pageviews will increase.

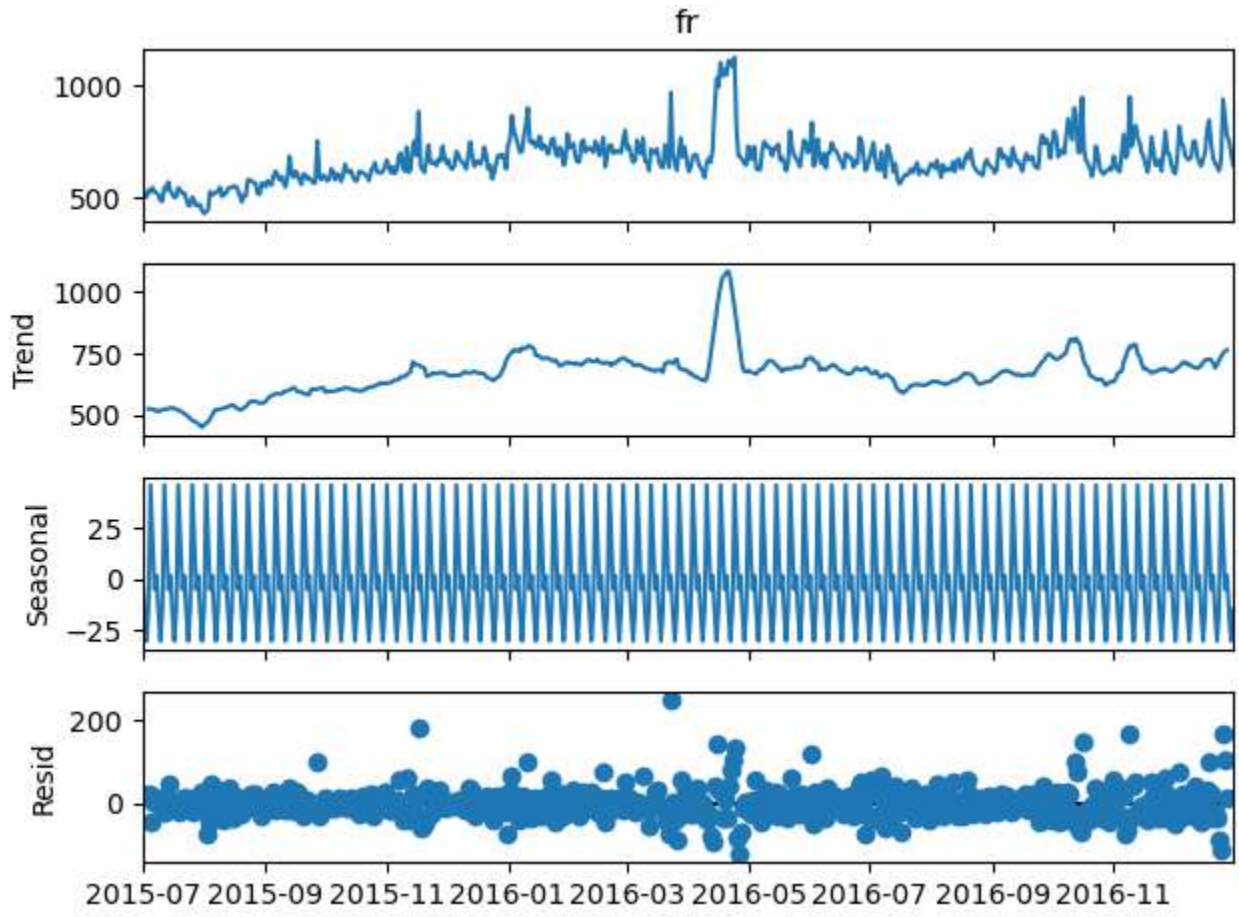
2. Seasonality shows that it is repeating in every `8` months for this language.

`Trend and Seasonality Check for French(FR) Language Pageviews`

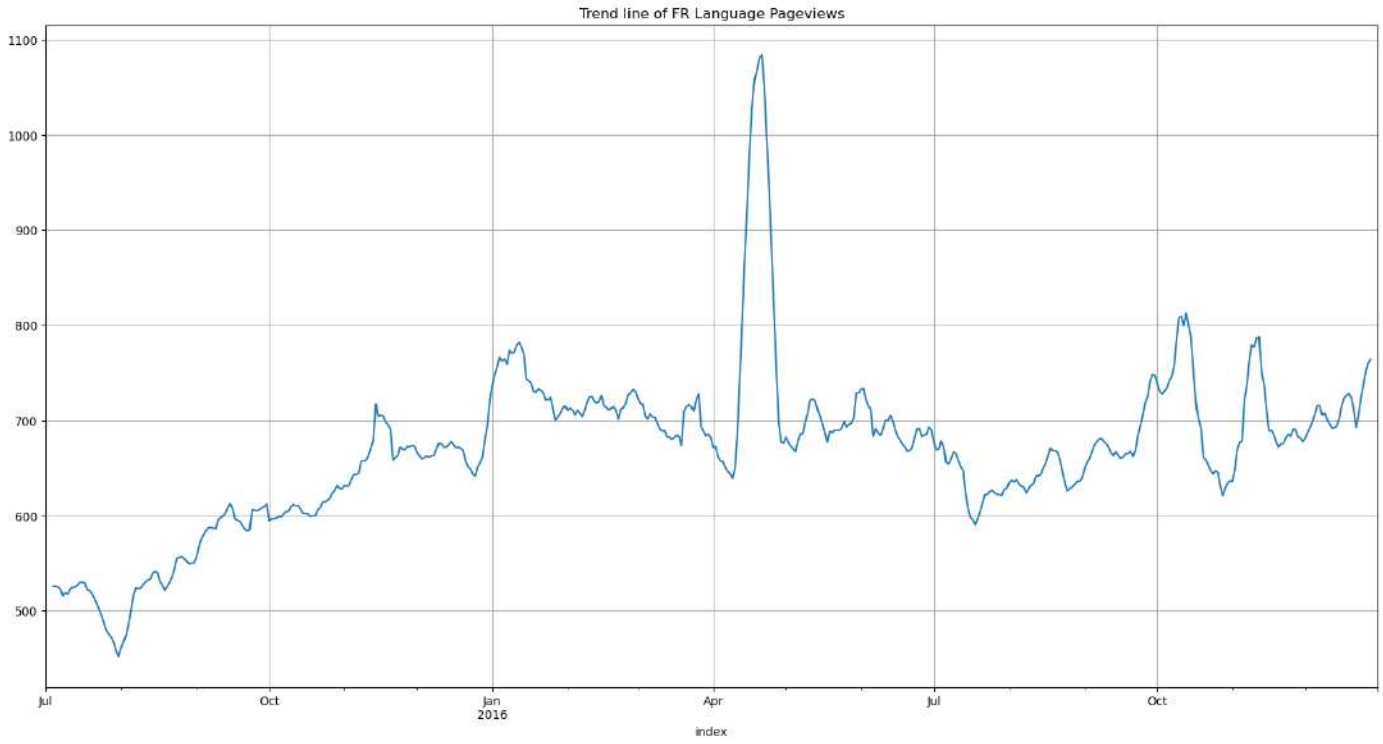
In [73]:
```
model_fr = sm.tsa.seasonal_decompose(df.fr)
```

In [74]:
```
model_fr.plot()
```

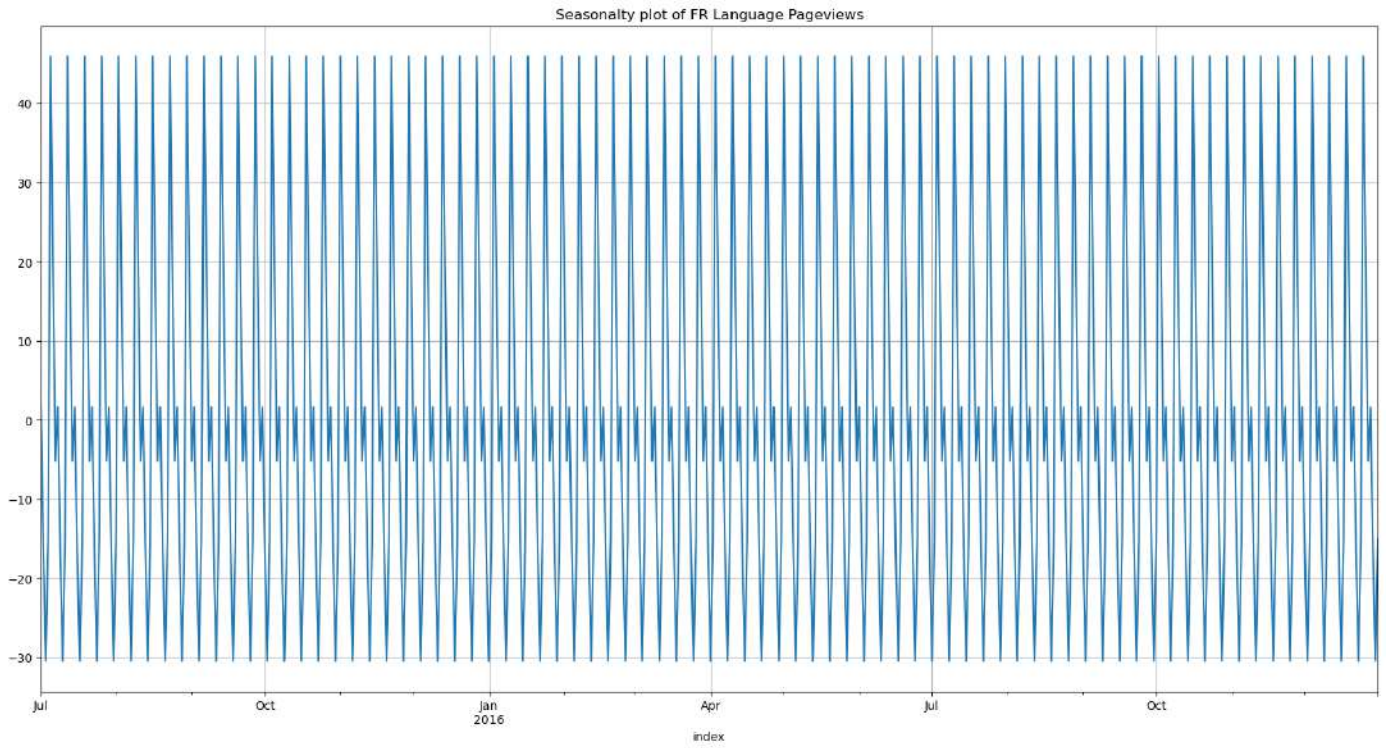
```
plt.show()
```



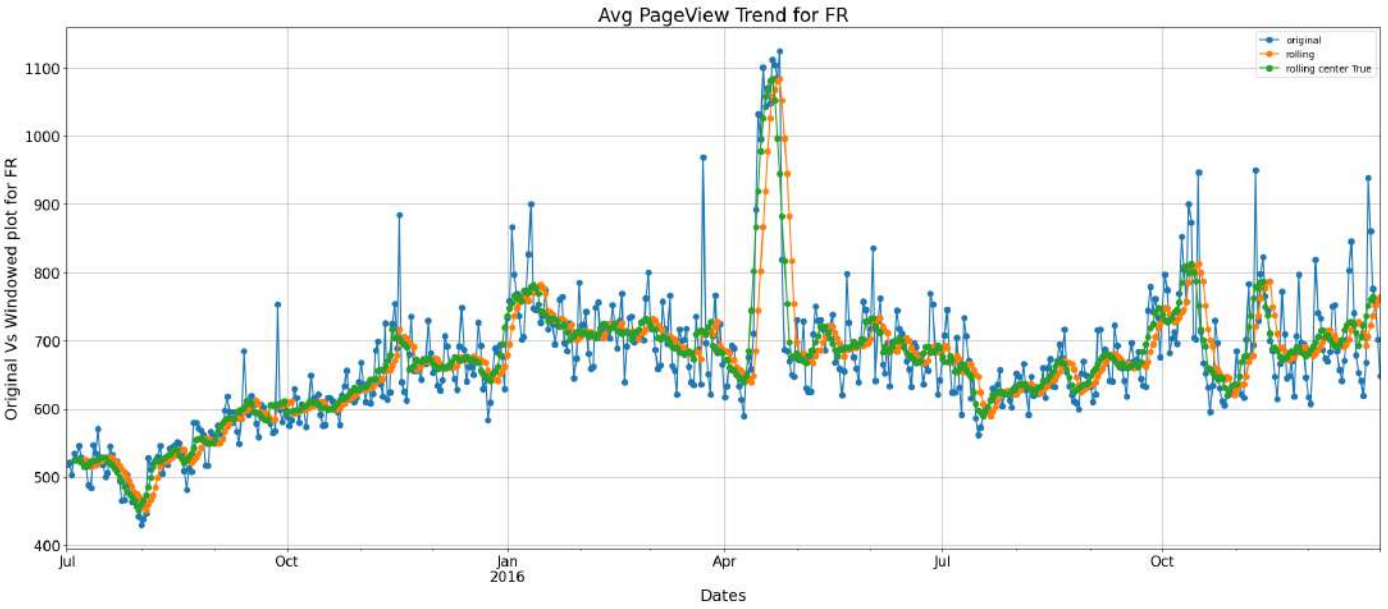
In [75]:
```
plt.title('Trend line of FR Language Pageviews')
model_fr.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [76]:
```
plt.title('Seasonalty plot of FR Language Pageviews')
model_fr.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [77]:
```
get_insight_plot(df['fr'],window=7,ln='FR')
```



Observation

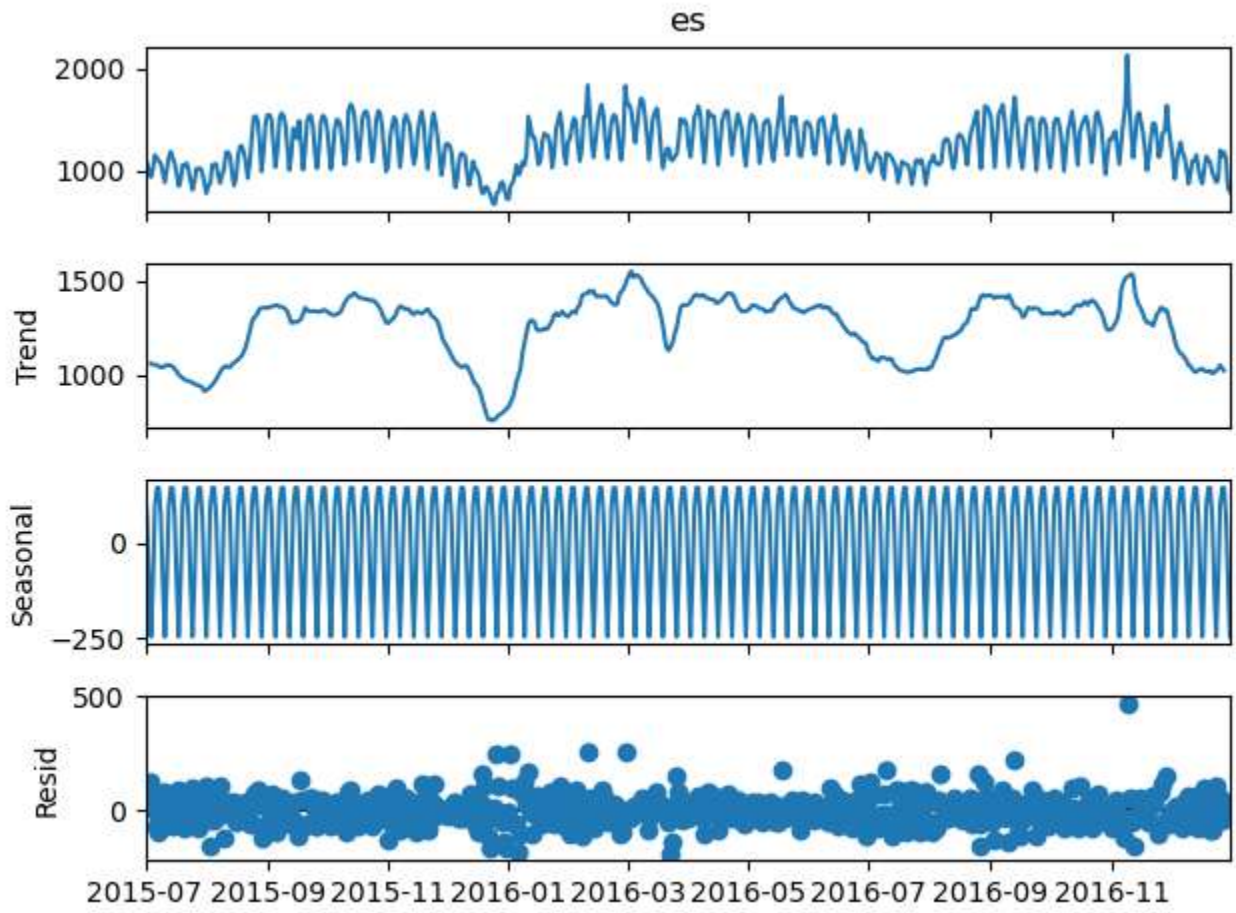
1. Trend is showing upward direction flow this indicates that with moving forward dates the pageviews may increase.
2. Seasonality shows that it is repeating in every `8` months for this language.

`Trend and Seasonality Check for Spanish(ES) Language Pageviews`

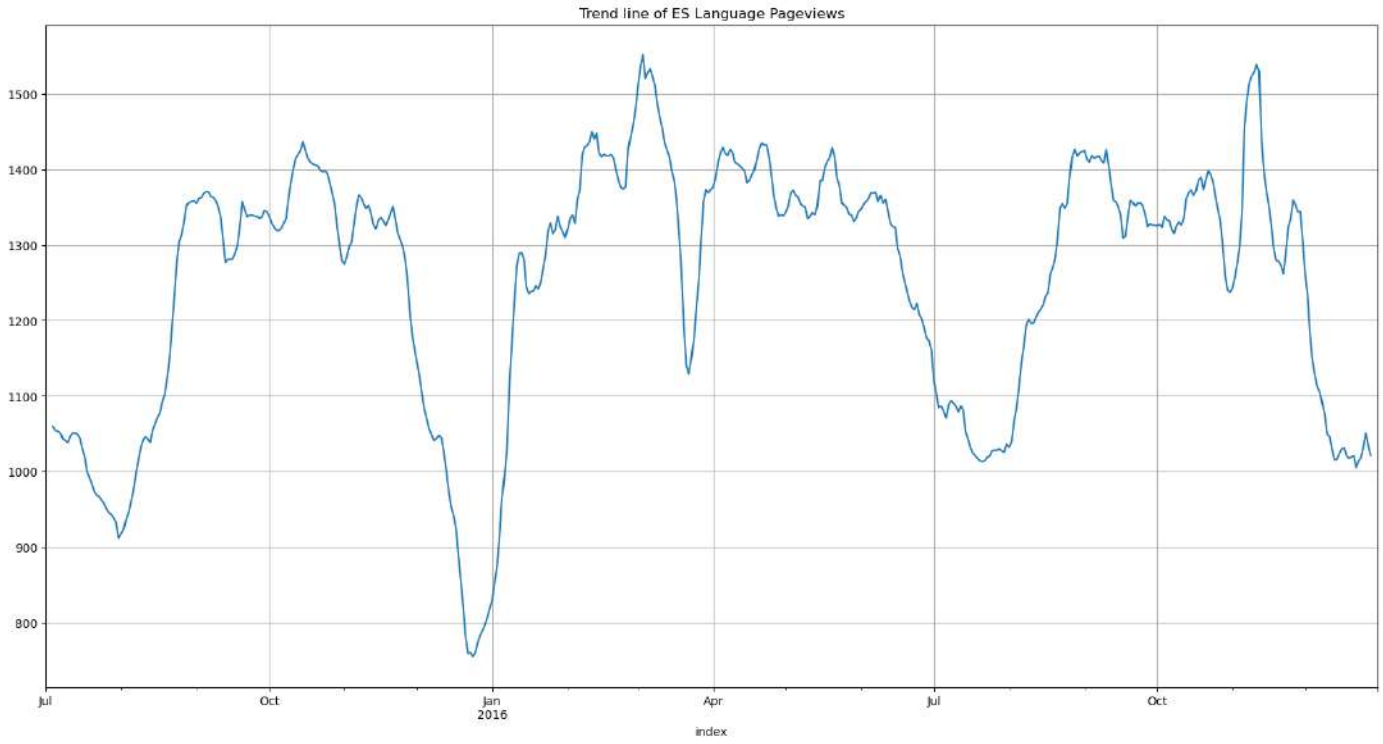
In [78]:
```
model_es = sm.tsa.seasonal_decompose(df.es)
```

In [79]:
```
model_es.plot()
```

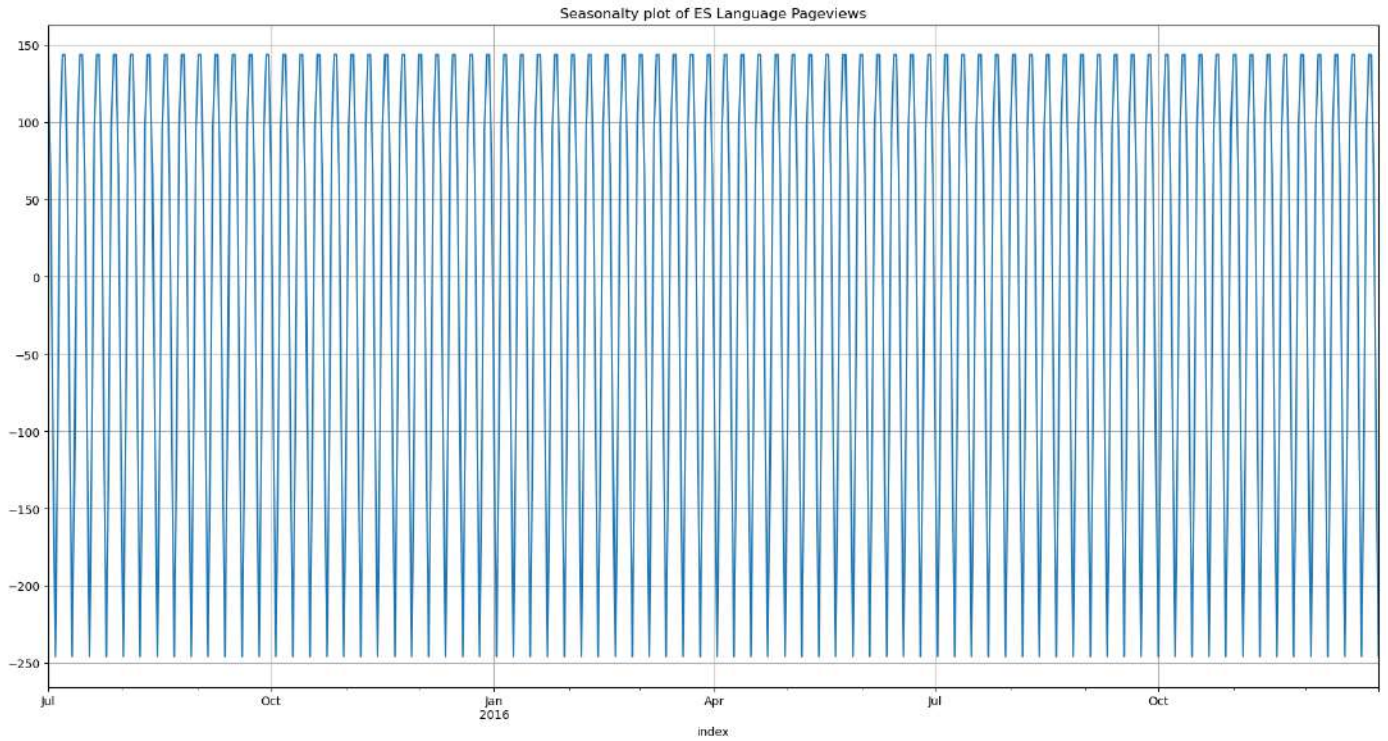
```
plt.show()
```



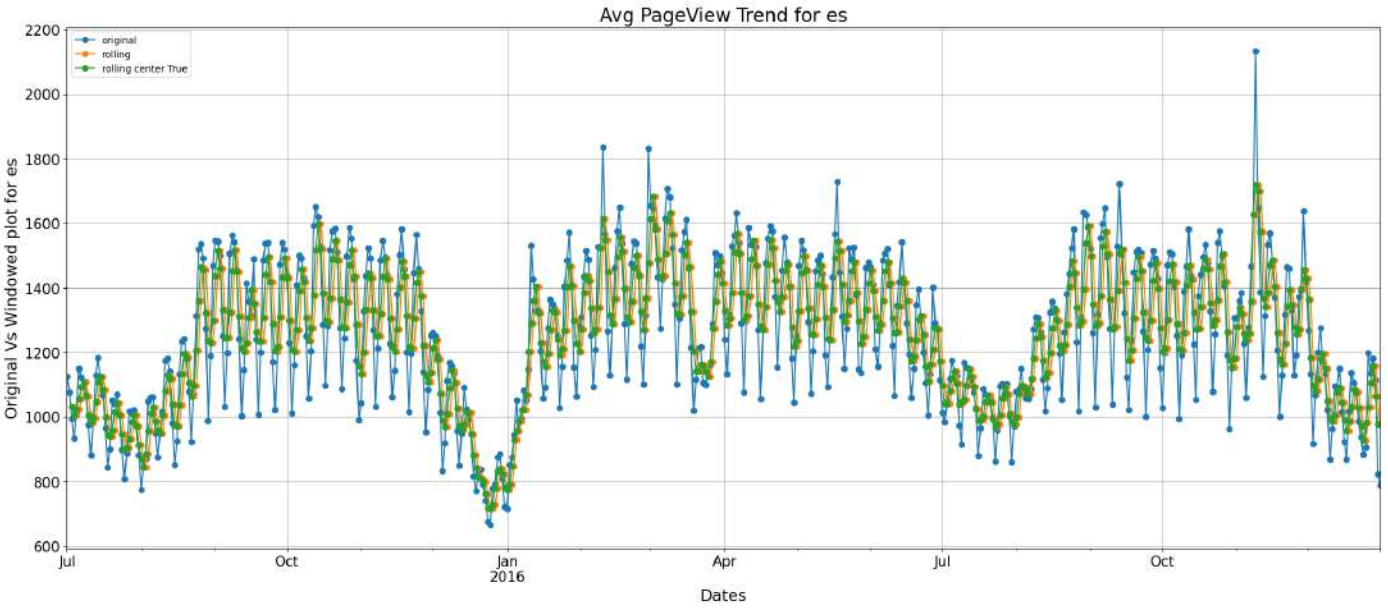
In [80]:
```
plt.title('Trend line of ES Language Pageviews')
model_es.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [81]:
```
plt.title('Seasonalty plot of ES Language Pageviews')
model_es.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [82]:
```
get_insight_plot(df['es'],window=4,ln='es')
```



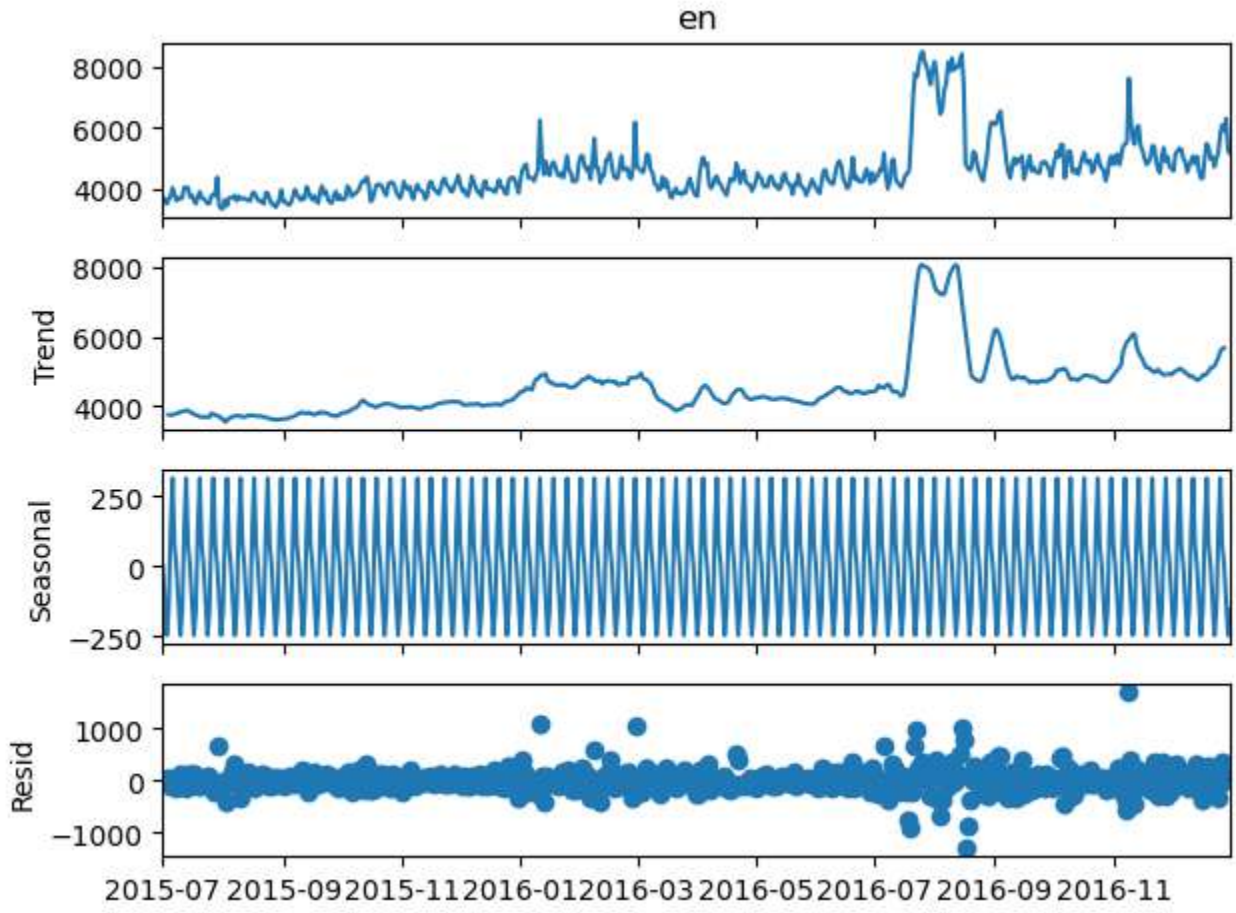
Observation

1. Trend is showing downward direction flow this indicates that with moving forward dates the pageviews may decrease.
2. Seasonality shows that it is repeating in every `7` months for this language.

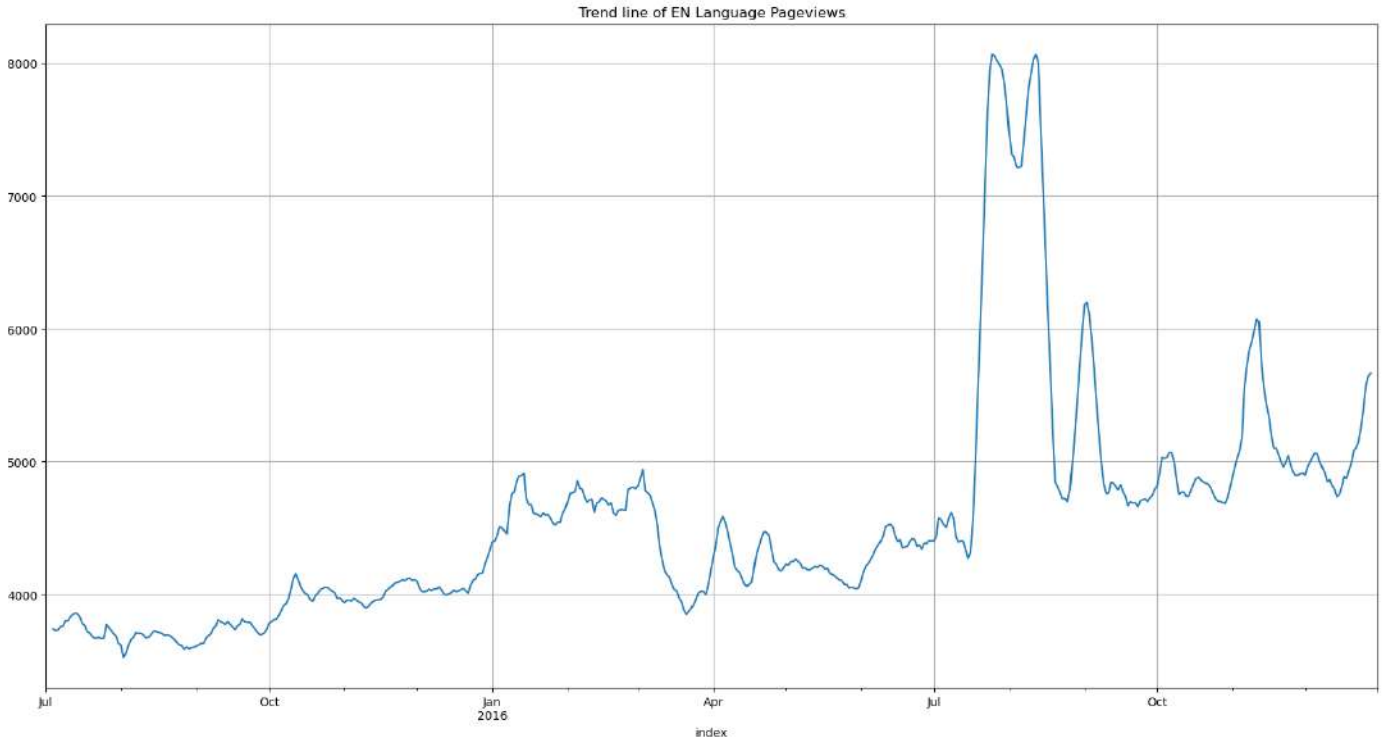
`Trend and Seasonality Check for English(EN) Language Pageviews`

In [83]:
```
model_en = sm.tsa.seasonal_decompose(df.en)
```

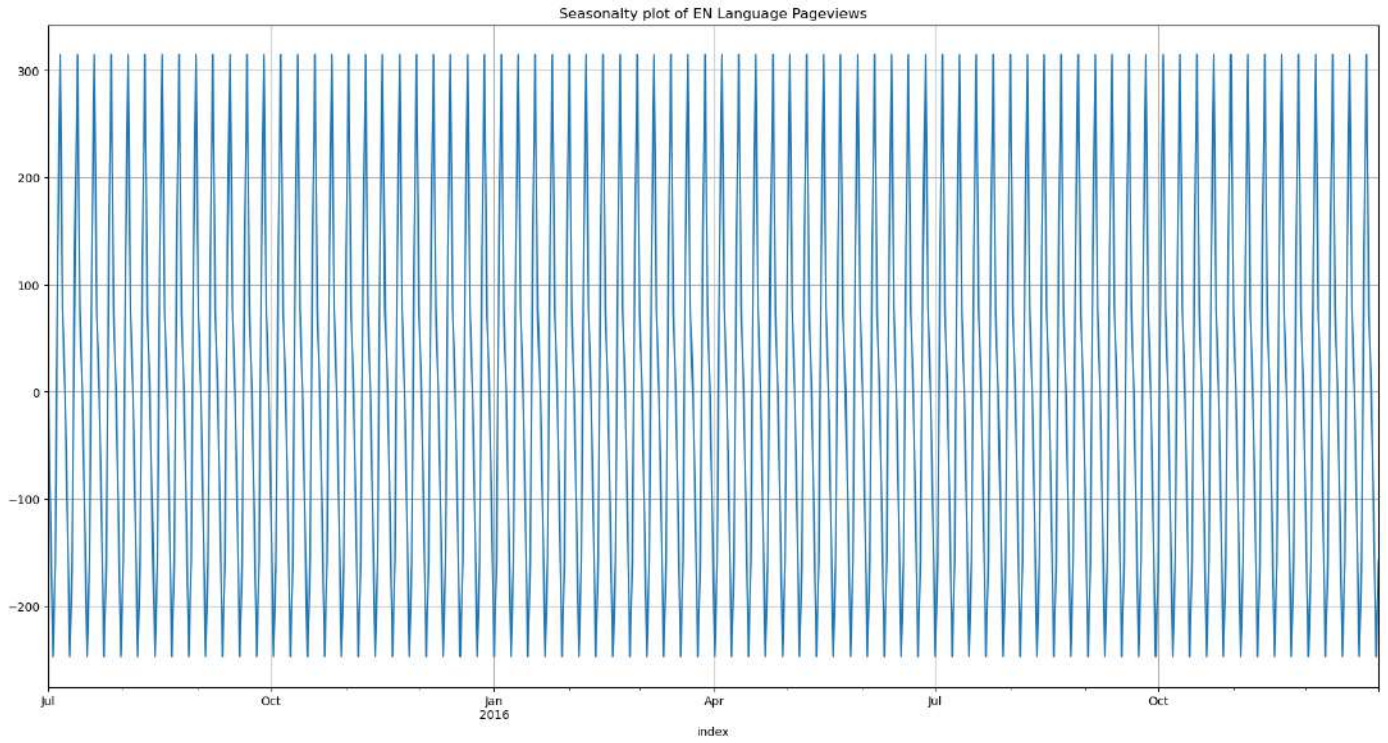
In [84]:
```
model_en.plot()
```



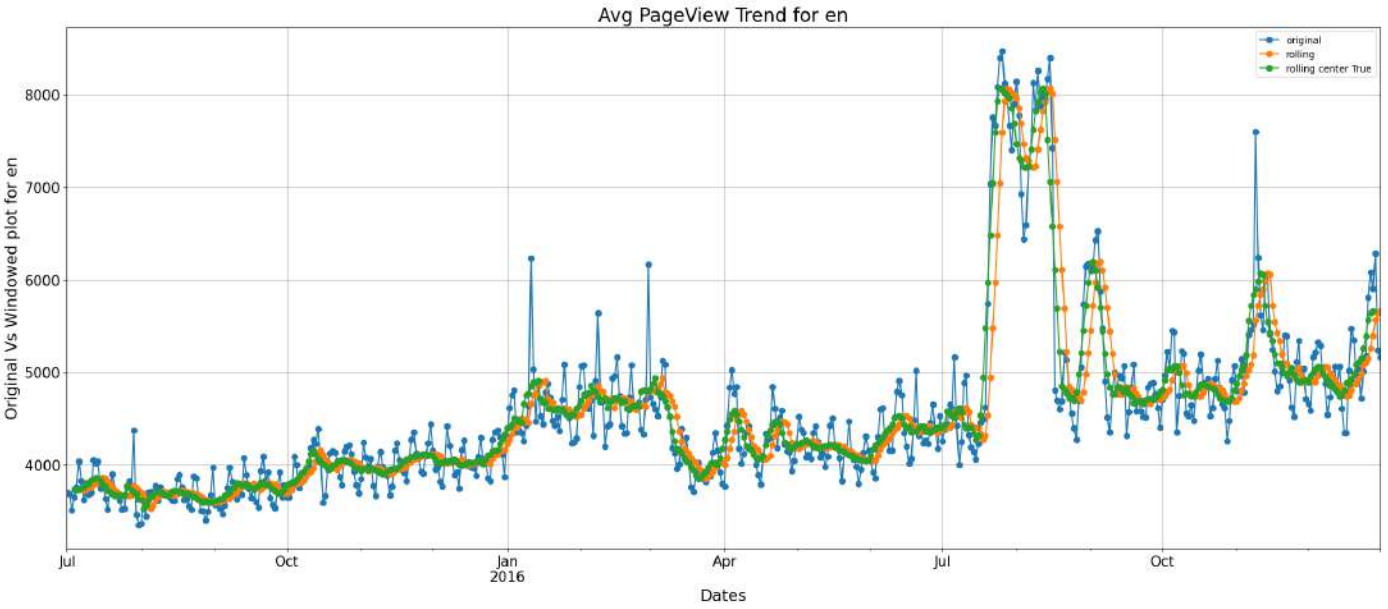
In [85]:
```
plt.title('Trend line of EN Language Pageviews')
model_en.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [86]:
```
plt.title('Seasonalty plot of EN Language Pageviews')
model_en.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [87]:
```
get_insight_plot(df['en'],window=7,ln='en')
```



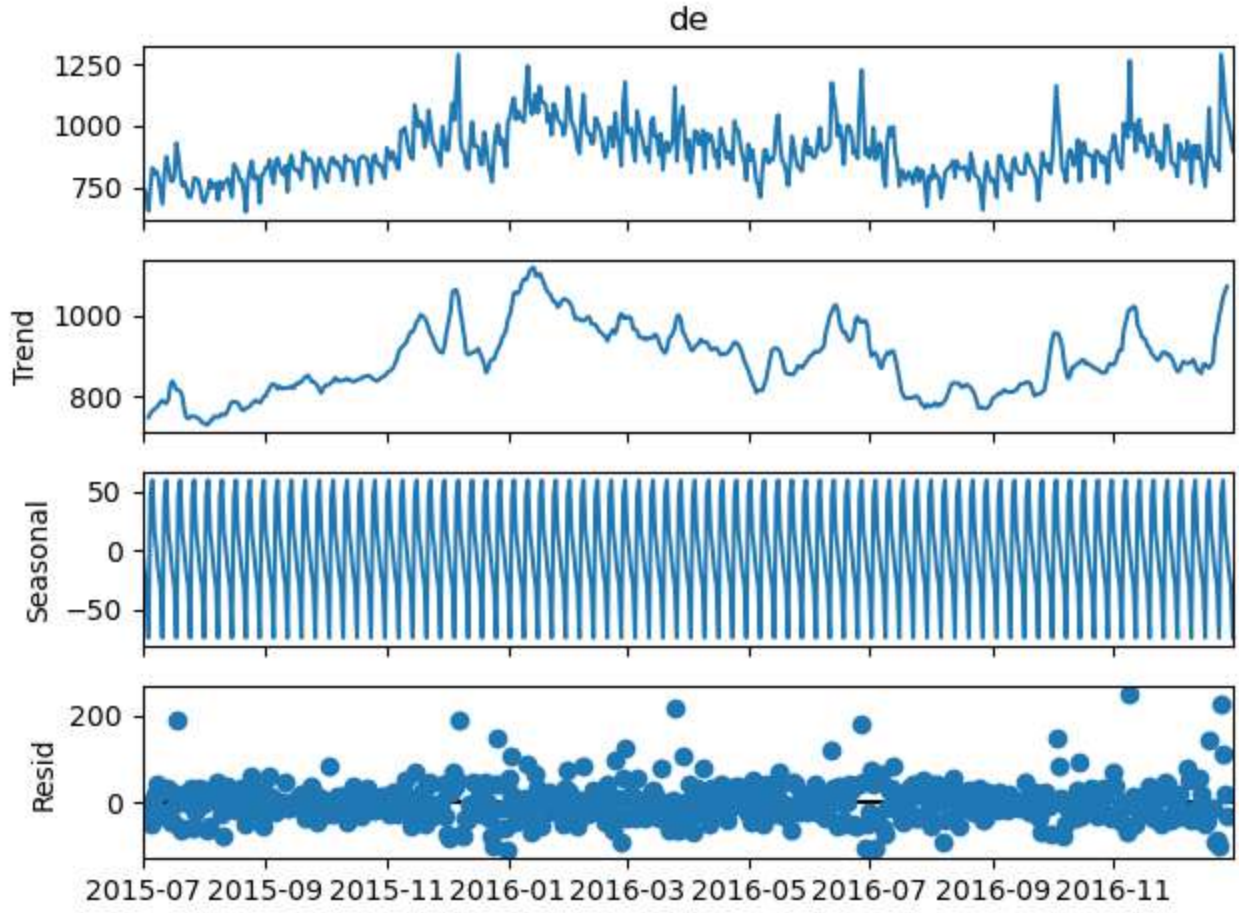
Observation

1. Trend is showing upward direction flow this indicates that with moving forward dates the pageviews may increase.
2. Seasonality shows that it is repeating in every `8` months for this language.

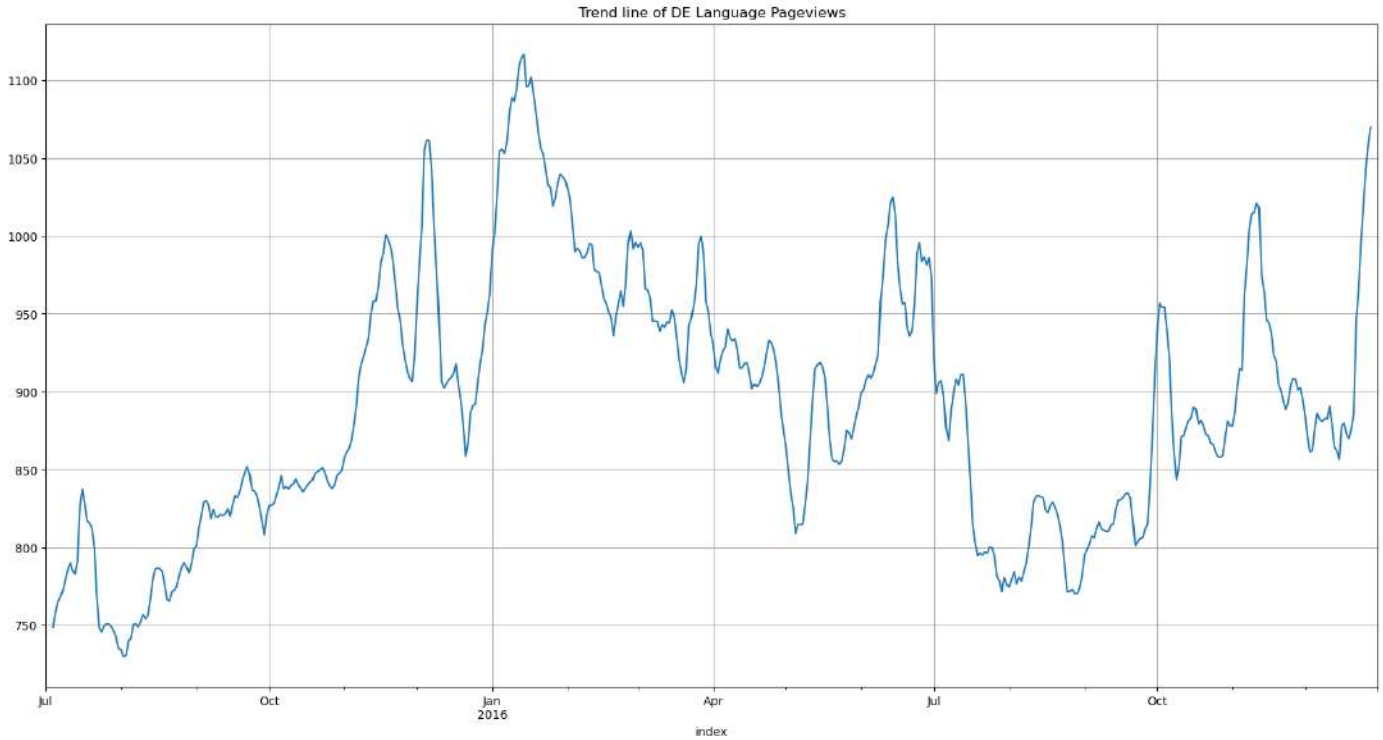
`Trend and Seasonality Check for German(DE) Language Pageviews`

In [88]:
```
model_de = sm.tsa.seasonal_decompose(df.de)
```

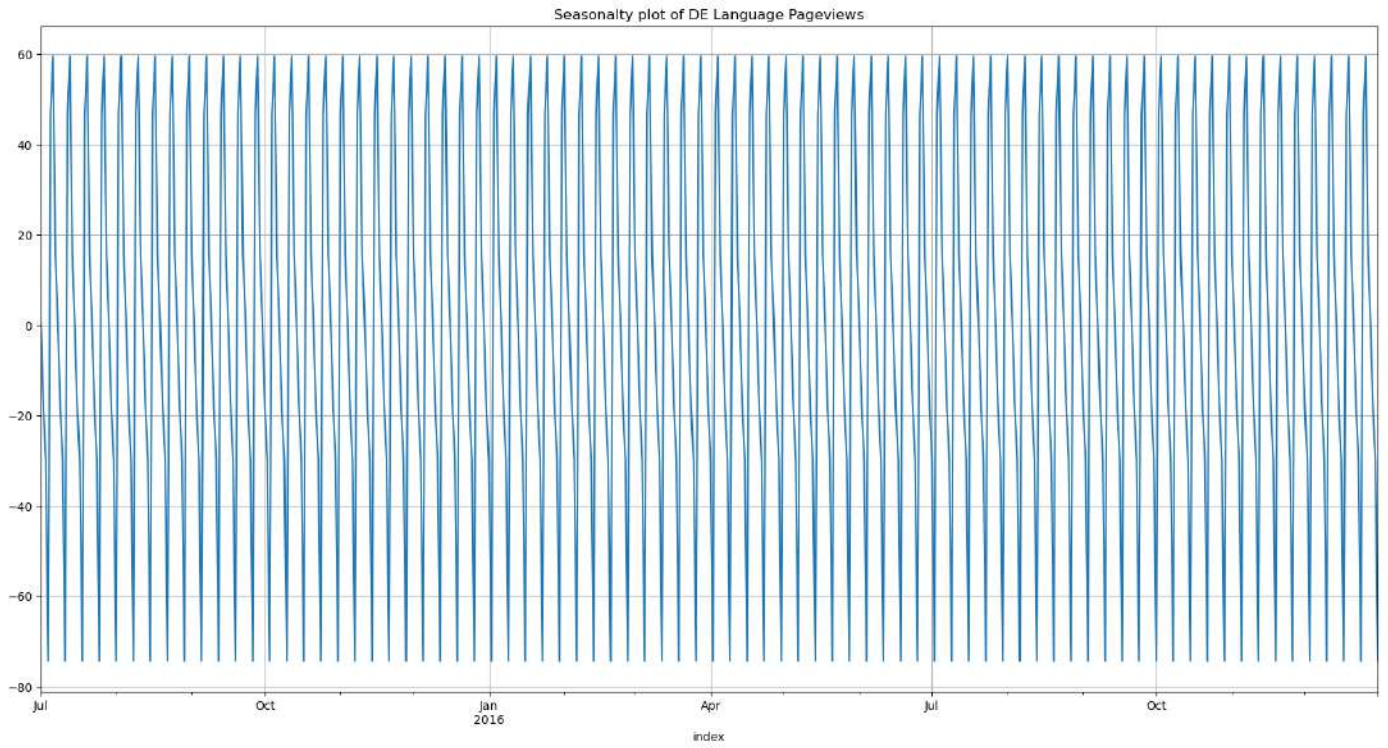
In [89]:
```
model_de.plot()
```



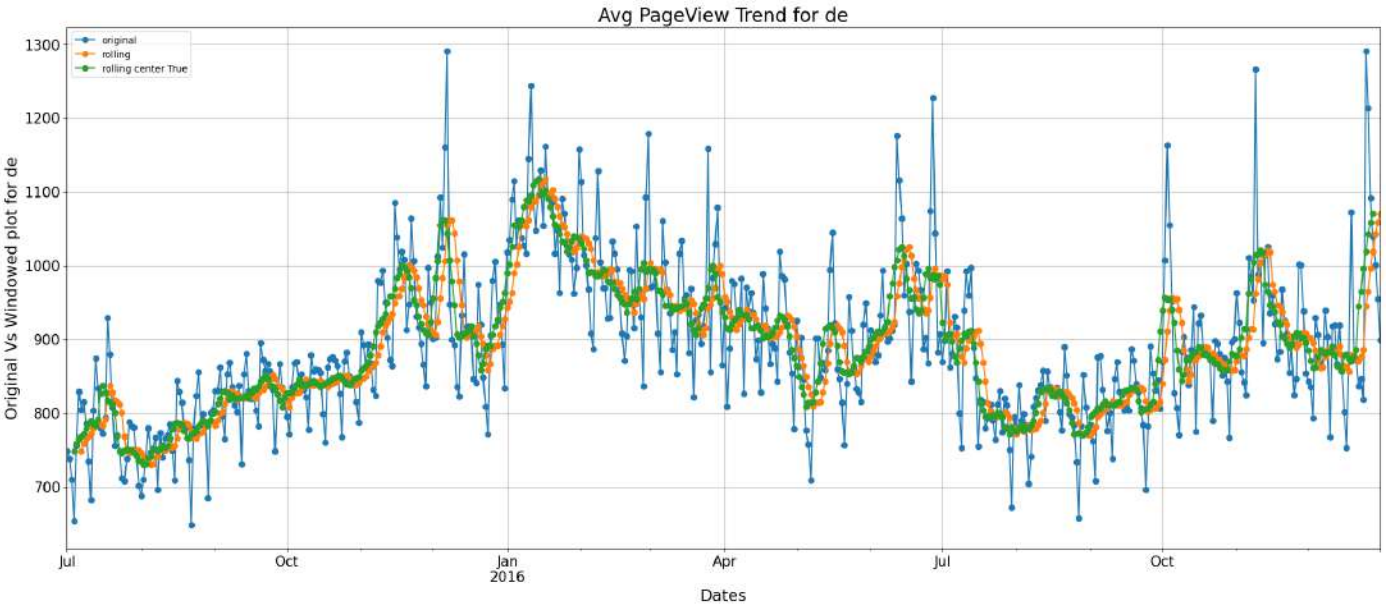
In [90]:
```
plt.title('Trend line of DE Language Pageviews')
model_de.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [91]:
```
plt.title('Seasonalty plot of DE Language Pageviews')
model_de.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [92]:
```
get_insight_plot(df['de'],window=7,ln='de')
```



Observation

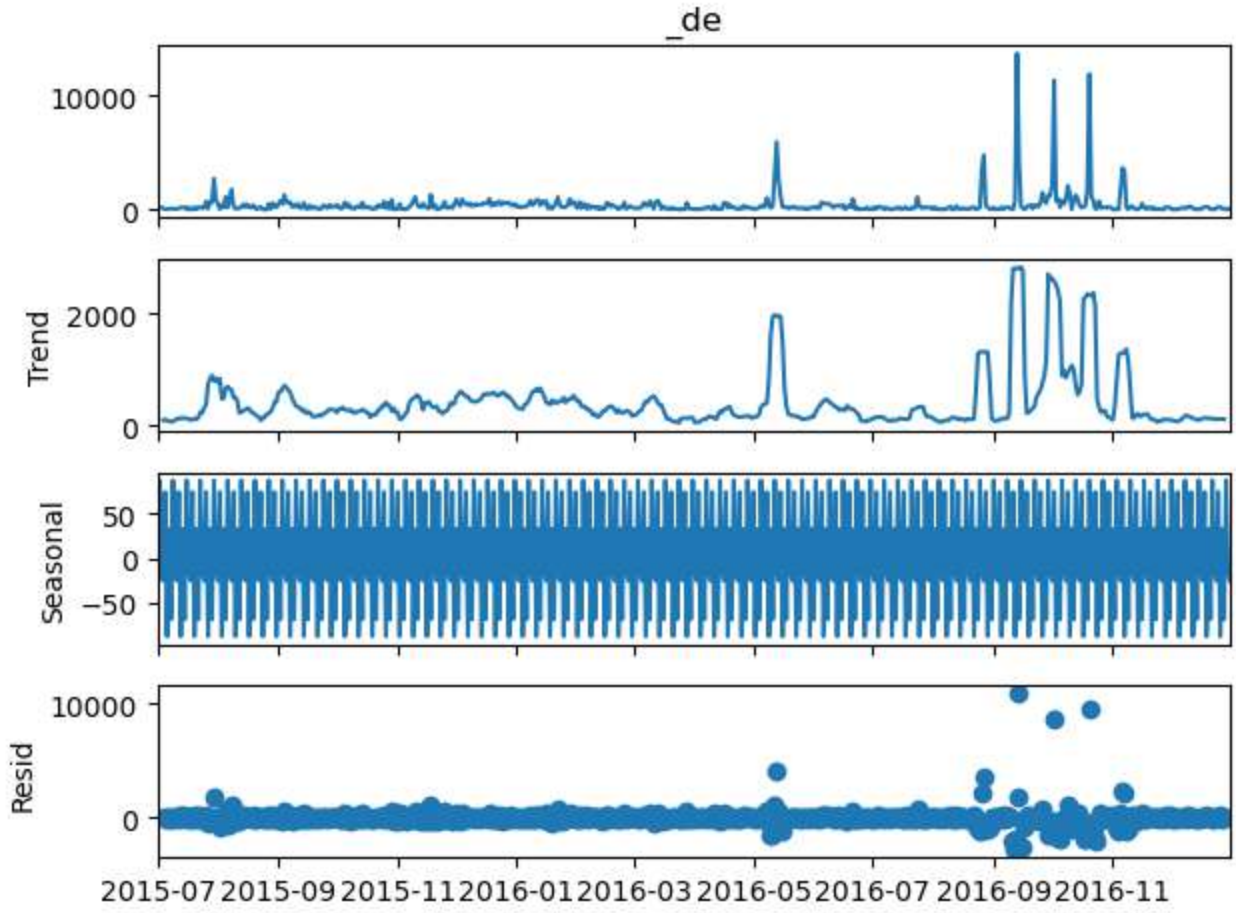
1. Trend is showing upward direction flow this indicates that with moving forward dates the pageviews may increase.
2. Seasonality shows that it is repeating in every `8` months for this language.

`Trend and Seasonality Check for German(_DE) Language Pageviews`

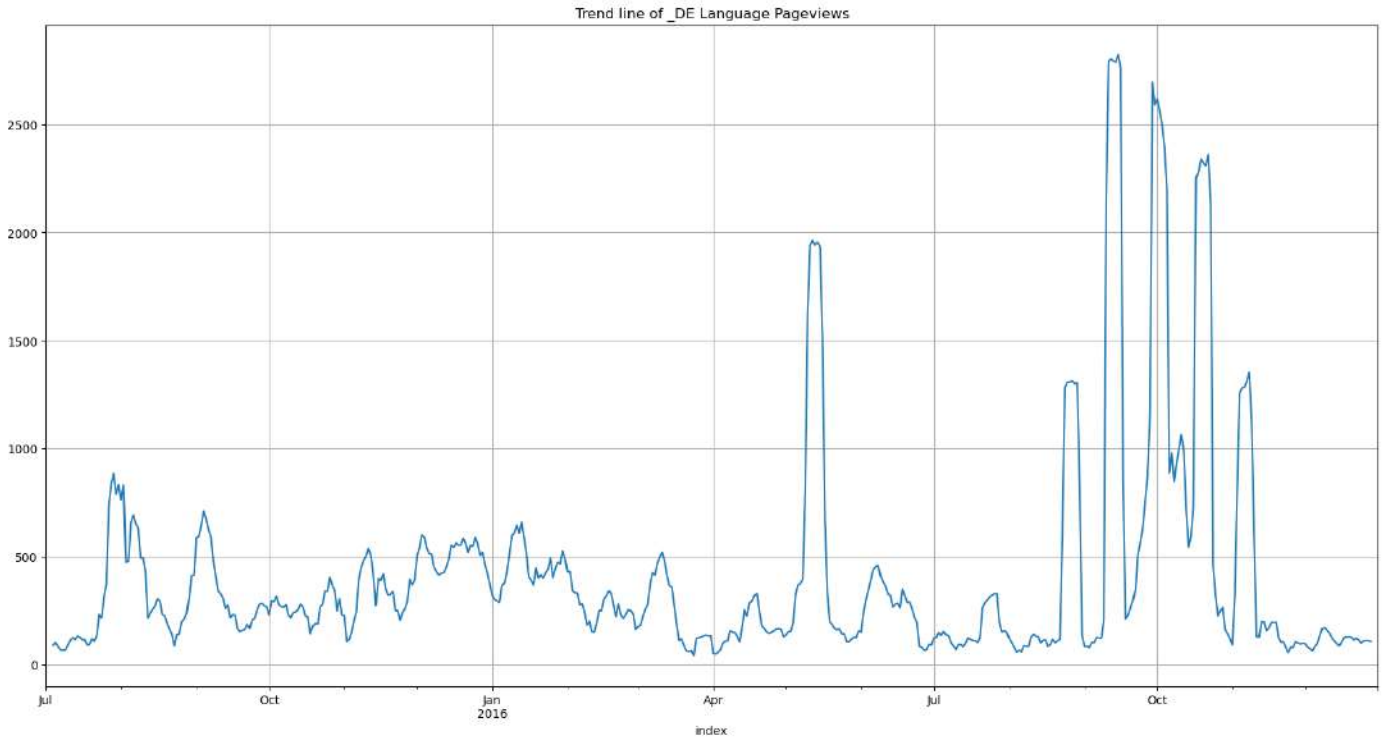
In [93]:
```
model__de = sm.tsa.seasonal_decompose(df._de)
```

In [94]:
```
model__de.plot()
```

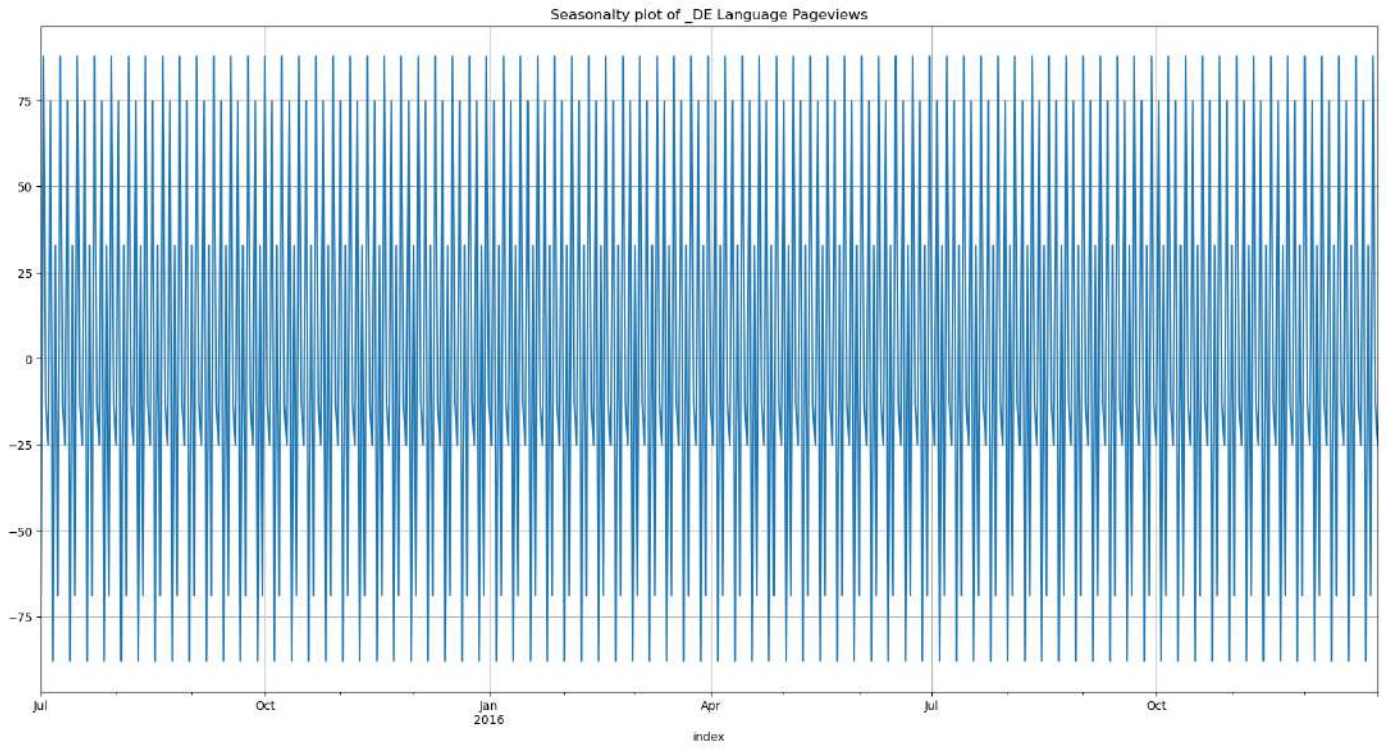
```
plt.show()
```



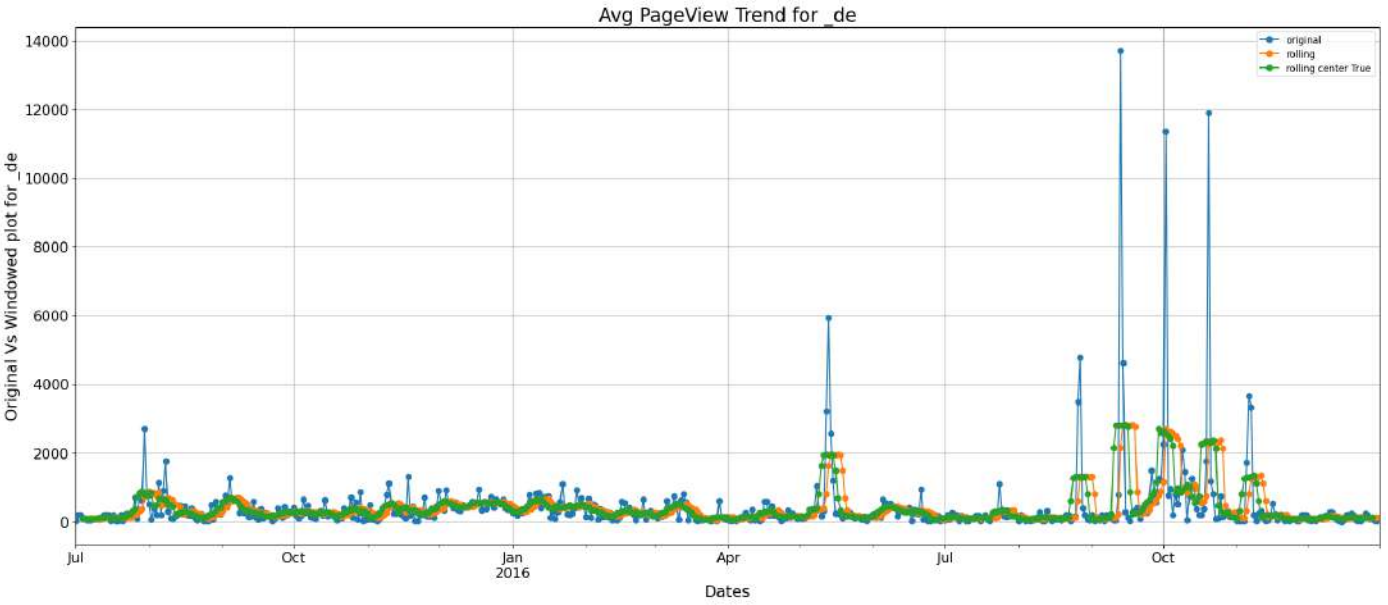
In [95]:
```
plt.title('Trend line of _DE Language Pageviews')
model__de.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



In [96]:
```
plt.title('Seasonalty plot of _DE Language Pageviews')
model__de.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [97]:
```
get_insight_plot(df['_de'],window=7,ln='_de')
```



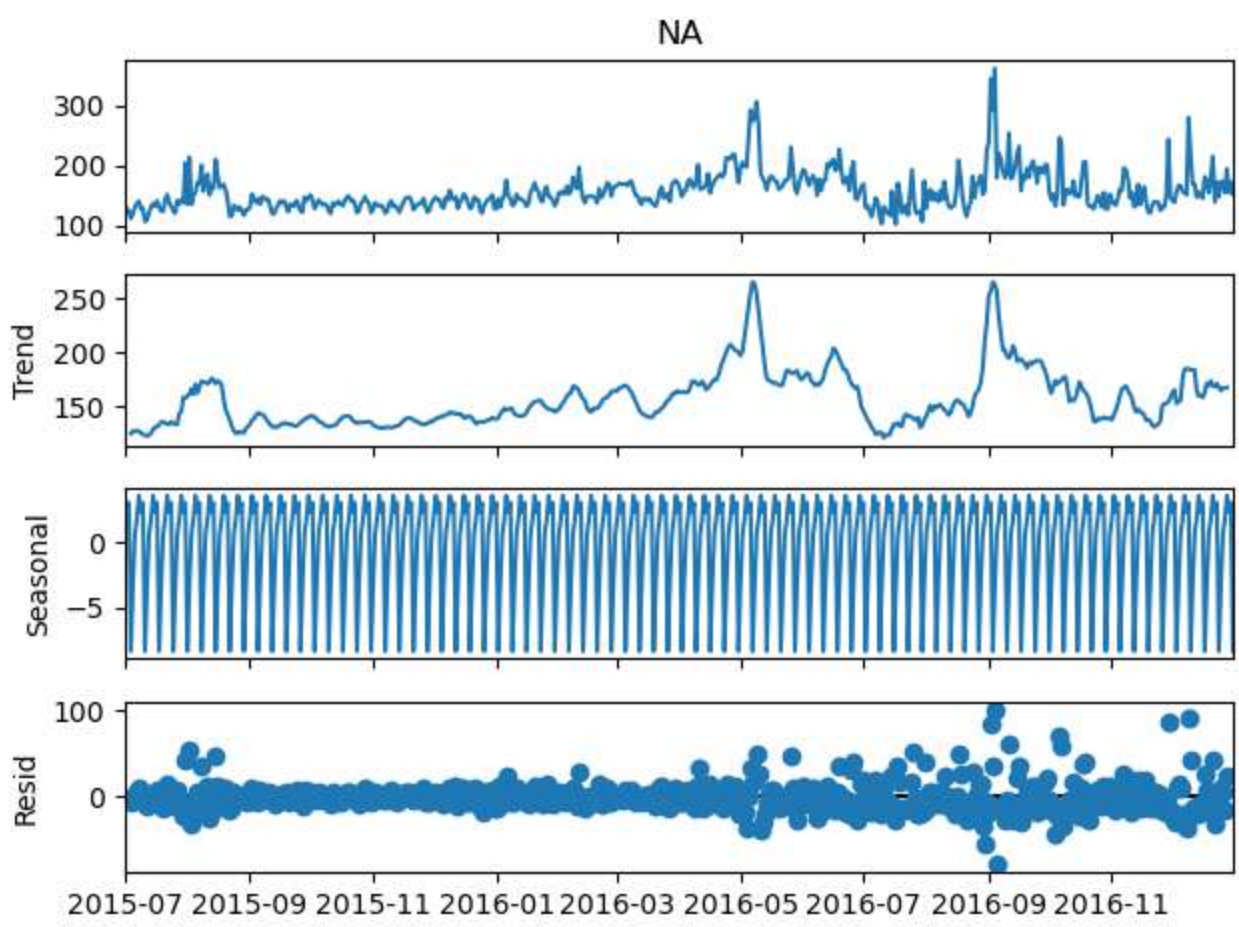
Observation

1. Trend is showing stationary flow this indicates that with moving forward dates the pageviews may be stationary in nature.

2. Seasonality shows that it is repeating in every `8` months for this language.

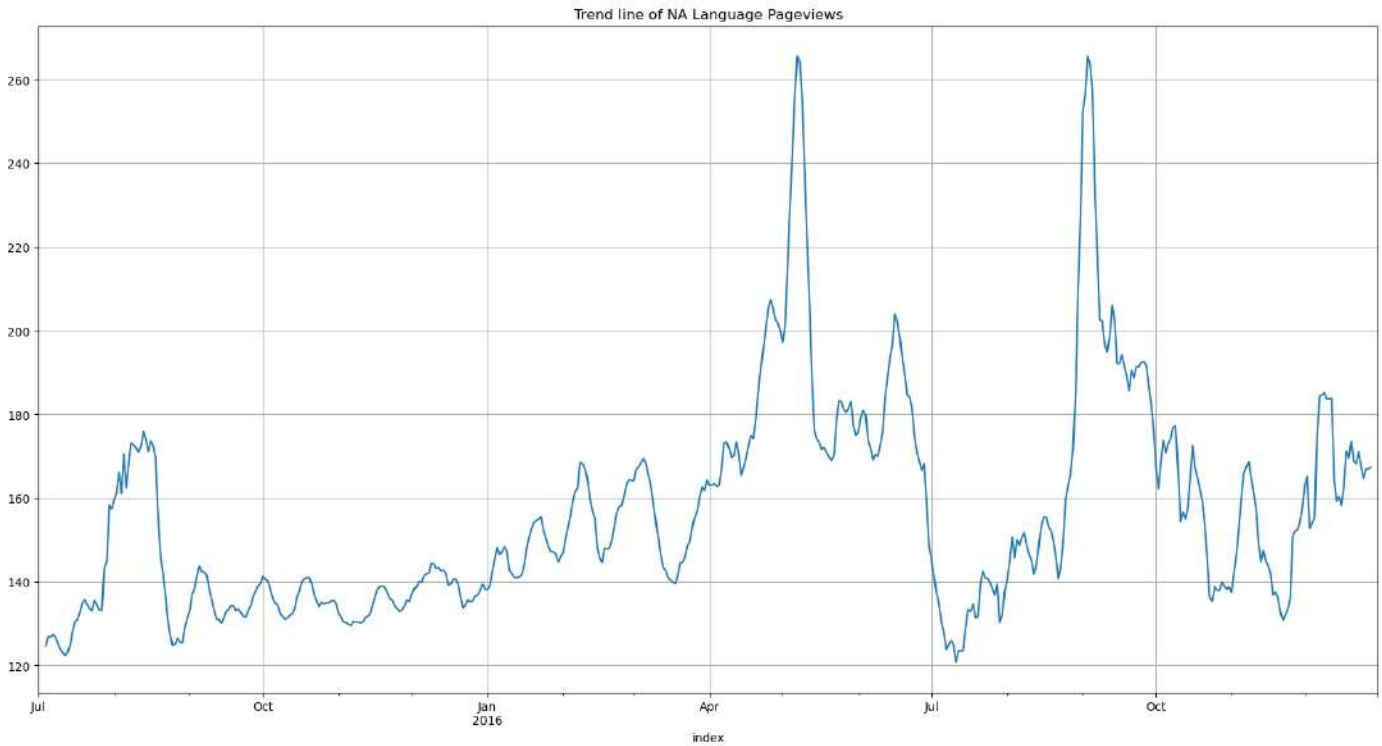
In [98]:
```
model_na = sm.tsa.seasonal_decompose(df.NA)
```

In [99]:
```
model_na.plot()
plt.show()
```



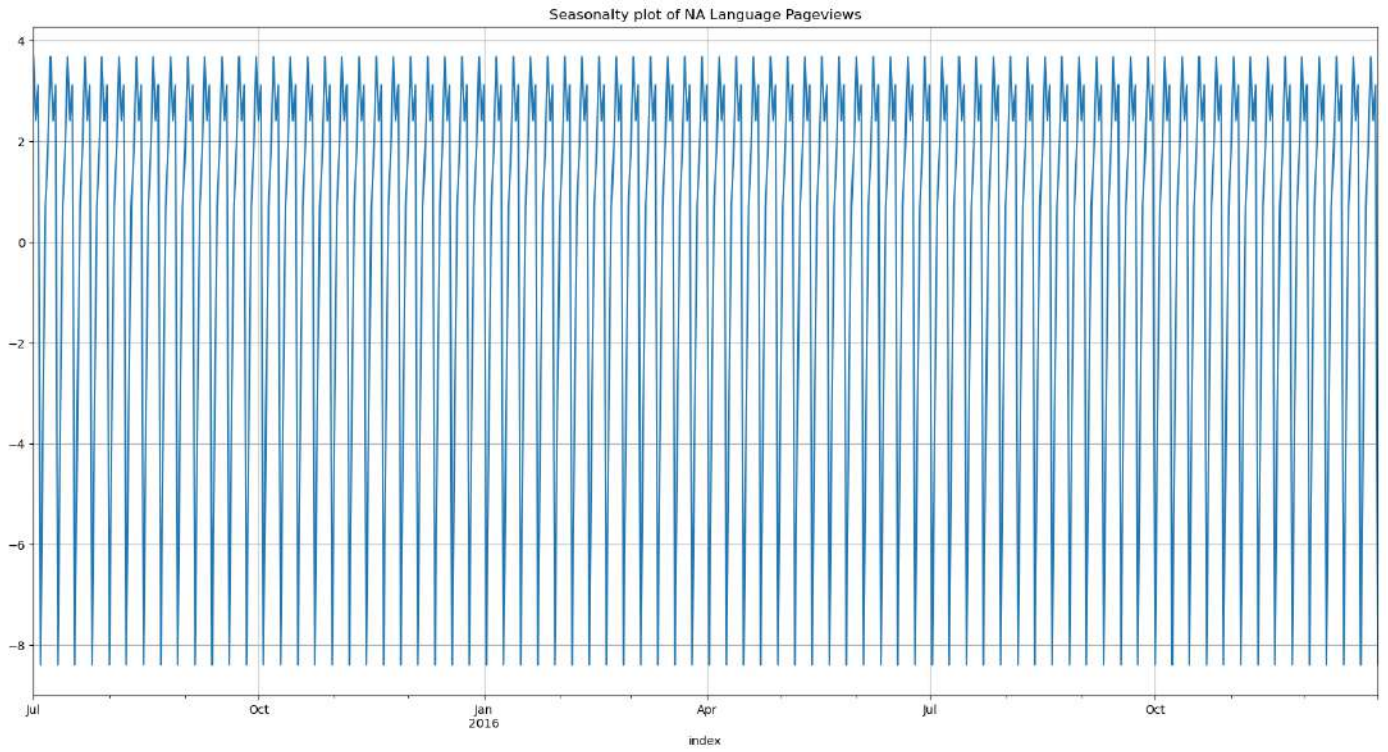
In [100…

```
plt.title('Trend line of NA Language Pageviews')
model_na.trend.plot(grid=True,figsize=(20,10))
plt.show()
```



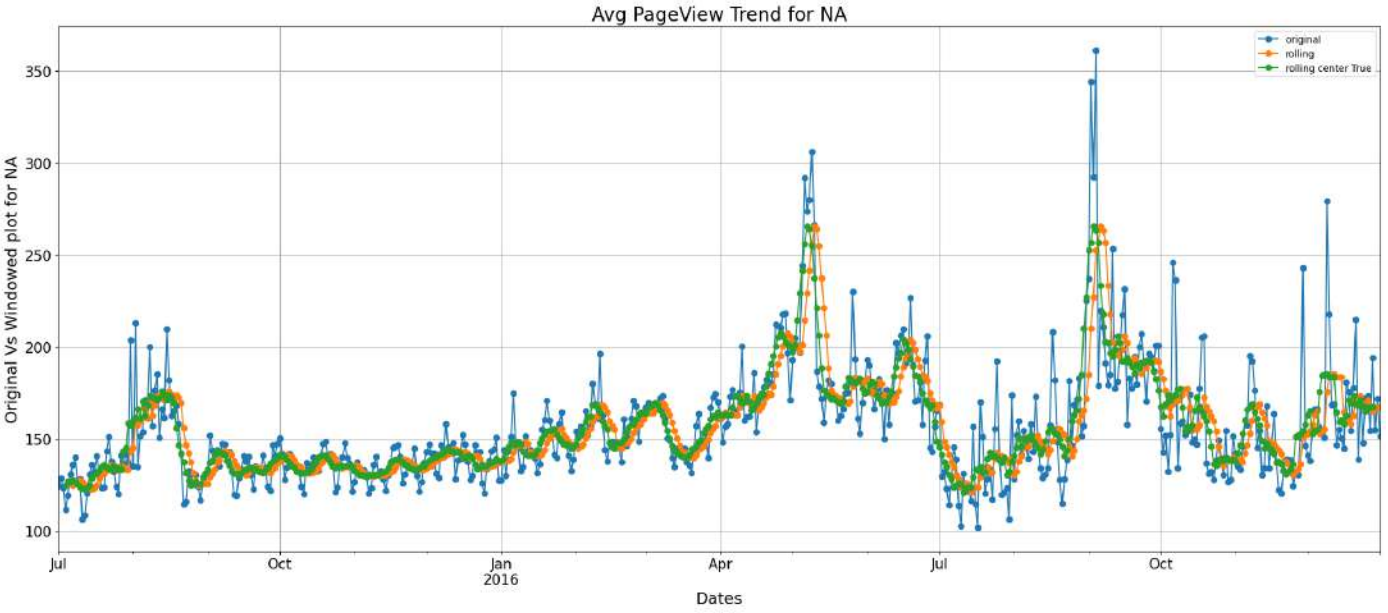
In [101…

```
plt.title('Seasonalty plot of NA Language Pageviews')
model_na.seasonal.plot(grid=True,figsize=(20,10))
plt.show()
```



In [102…

```
get_insight_plot(df['NA'],window=7,ln='NA')
```



Observation

1. Trend is showing upward flow this indicates that with moving forward dates the pageviews may increase.

2. Seasonality shows that it is repeating in every `8` months for this language.

Overall Observation from Above EDA Analysis

1. Whole Dataset has seasonality of 7 months.

2. Due to presence of outliers we can see spikes in trend, which we can fix either imputation or by clipping.

3. Following Language have shown below trend type

   ZH - > High Trend

   RU -> Lower Trend

   JA -> High Trend

   FR -> High Trend

   ES -> Lower Trend

   EN -> High Trend

   DE -> High Trend

   _DE -> Lower Trend

   NA -> High Trend

Stationarity Check For Data

In [103…

```
# H0 : TS is non stationary
# H1 : TS is stationary

def adf_test(data,sig_val=0.05):
    p_value = sm.tsa.stattools.adfuller(data)[1]
    #print(f'p-value : {p_value}')
    if(p_value <= sig_val):
        return('TS is stationary',p_value)
    else:
        return('TS is non stationary',p_value)
```

In [104…

```
zh_res,_ = adf_test(df['zh'])
print(f'Stationary Status for ZH Language: {zh_res}')

ru_res,_ = adf_test(df['ru'])
print(f'Stationary Status for RU Language: {ru_res}')

ja_res,_ = adf_test(df['ja'])
print(f'Stationary Status for JA Language: {ja_res}')

fr_res,_ = adf_test(df['fr'])
print(f'Stationary Status for FR Language: {fr_res}')

es_res,_ = adf_test(df['es'])
print(f'Stationary Status for ES Language: {es_res}')

en_res,_ = adf_test(df['en'])
print(f'Stationary Status for EN Language: {en_res}')

de_res,_ = adf_test(df['de'])
print(f'Stationary Status for DE Language: {de_res}')

adf_test(df['_de'])
```

```
print(f'Stationary Status for _DE Language: {_de_res}')

NA_res,_ = adf_test(df['NA'])
print(f'Stationary Status for NA Language: {NA_res}')
```

```
Stationary Status for ZH Language: TS is non stationary
Stationary Status for RU Language: TS is stationary
Stationary Status for JA Language: TS is non stationary
Stationary Status for FR Language: TS is stationary
Stationary Status for ES Language: TS is stationary
Stationary Status for EN Language: TS is non stationary
Stationary Status for DE Language: TS is non stationary
Stationary Status for _DE Language: TS is non stationary
Stationary Status for NA Language: TS is stationary
```

Observation

Below Mentioned Language trends are not stationary

1. ZH
2. JA
3. EN
4. DE
5. _DE

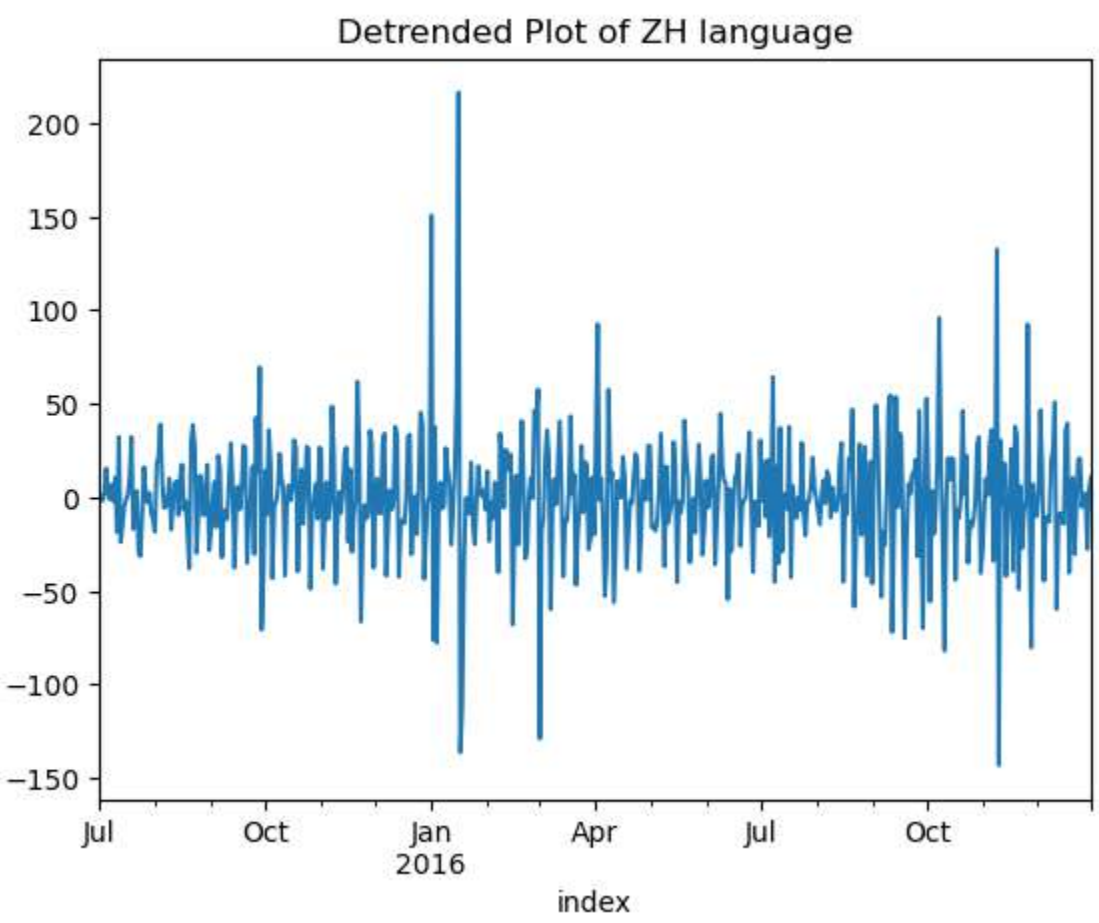
`Need to Perform Difference and de-trending to make it stationary.`

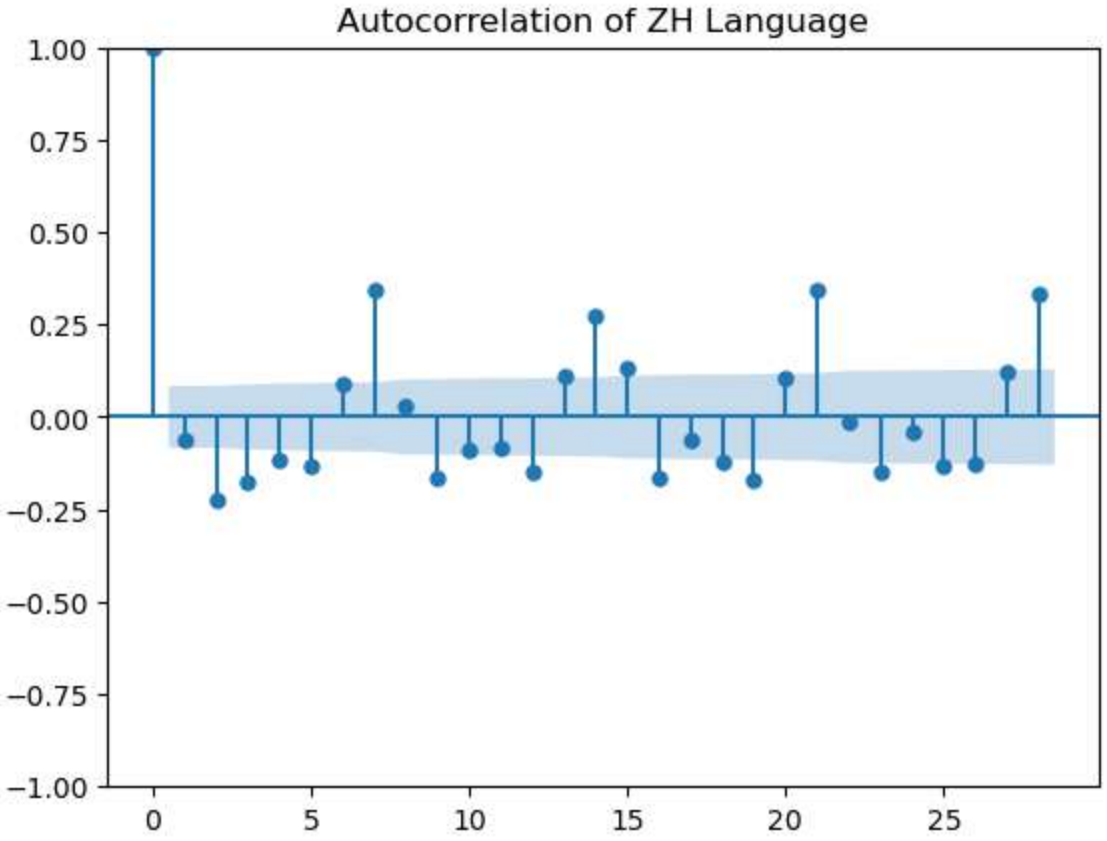
In [105…
```
def plot_acfs(val,title):
    """
    This function will Autocorrelation plot for provided language code.
    """
    plot_acf(val,title=f'Autocorrelation of {title} Language')
    plt.show()
```

In [106…
```
def get_detrended_deseasonal_data(val,title,m_val=None):
    detrended_val = val.diff()
    detrended_val.plot(title=f'Detrended Plot of {title} language')
    print(f'ADF Test after Detrending {title} language:{adf_test(detrended_val.dropna())
    plot_acfs(detrended_val.dropna(),title)
    return detrended_val
#     de_seasoned = detrended_val.diff(m_val).dropna()
#     plot_acfs(de_seasoned.dropna(),title=f'detrened & Deseasoned {title} language')
#     print(f'ADF Test after Deseasoned {title} Language:{adf_test(de_seasoned.dropna())
#     return de_seasoned
```

In [107…
```
deTrended_ZH_lang = get_detrended_deseasonal_data(df['zh'],'ZH')
```

```
ADF Test after Detrending ZH language:('TS is stationary', 1.0822615719686195e-11)
```

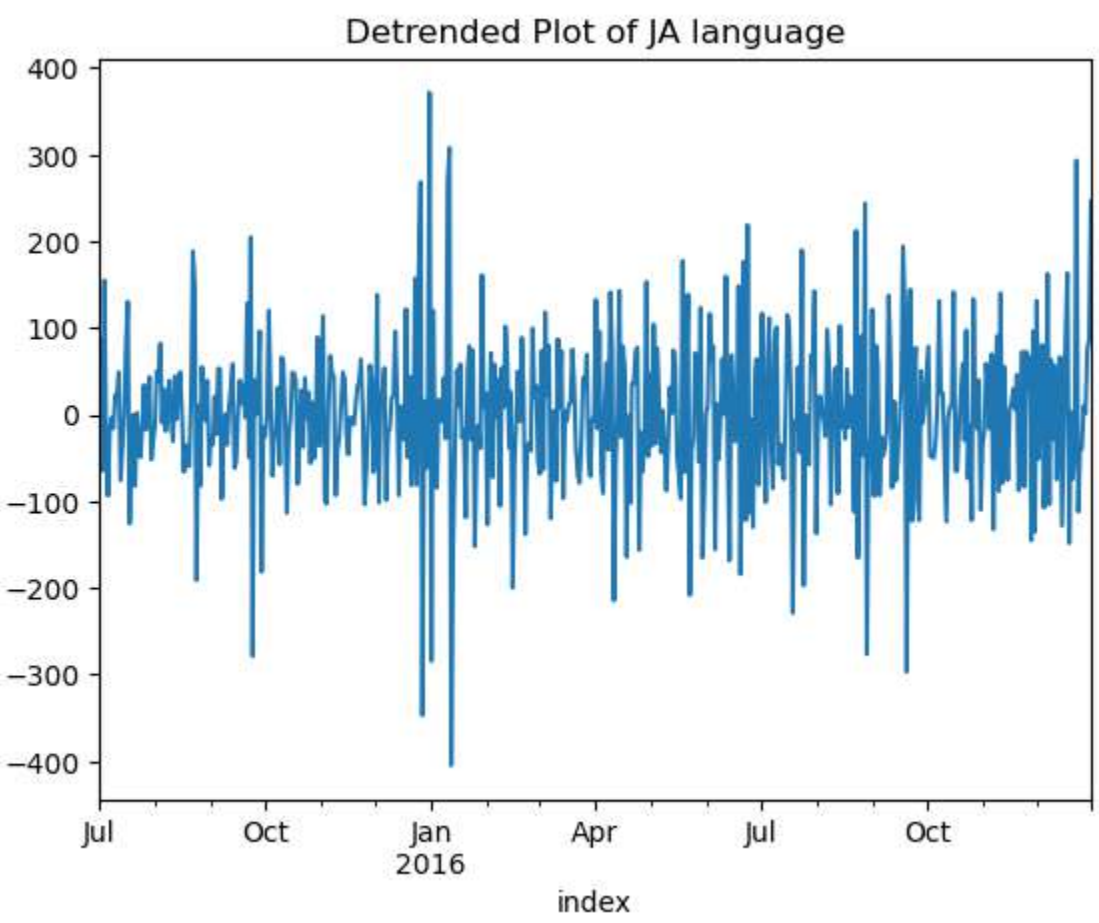


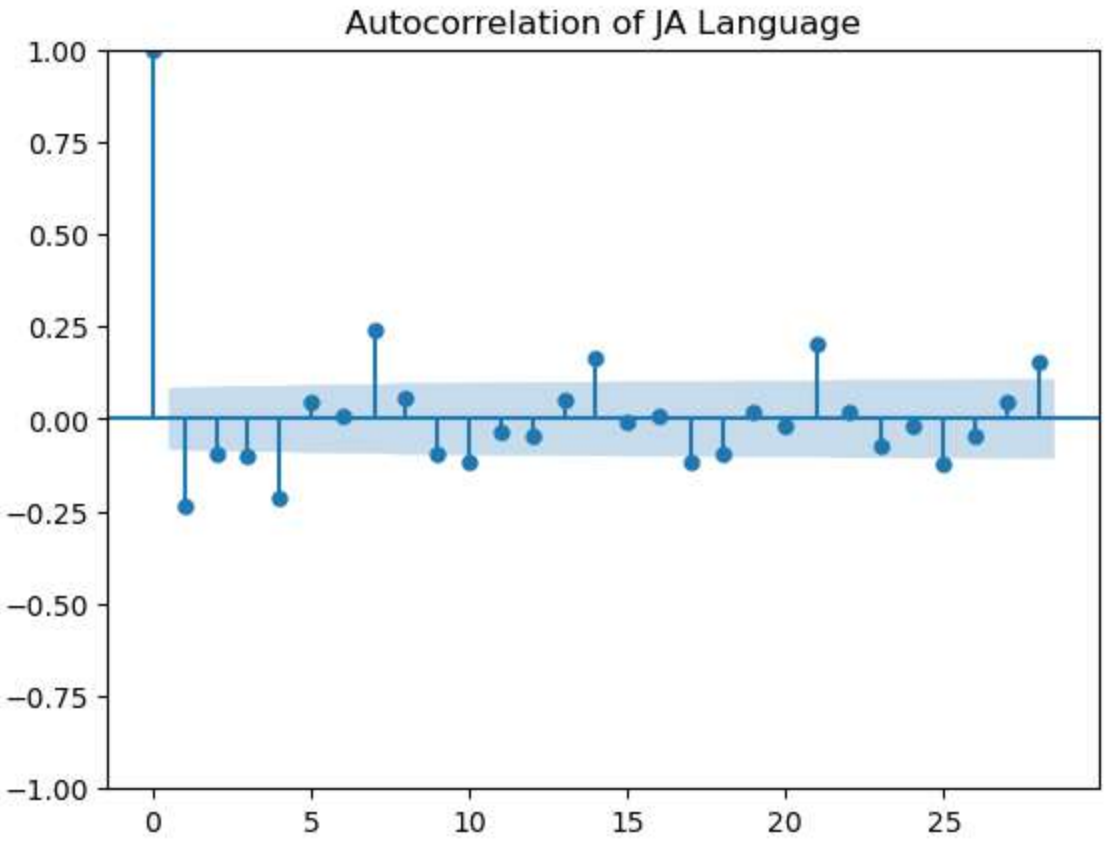


In [108…
```
# ZH -> Peak at 7 ,14,21
# p -> 8 q-> 5
```

In [109…
```
deTrended_JA_lang = get_detrended_deseasonal_data(df['ja'],'JA')
```

```
ADF Test after Detrending JA language:('TS is stationary', 5.950183790262341e-20)
```

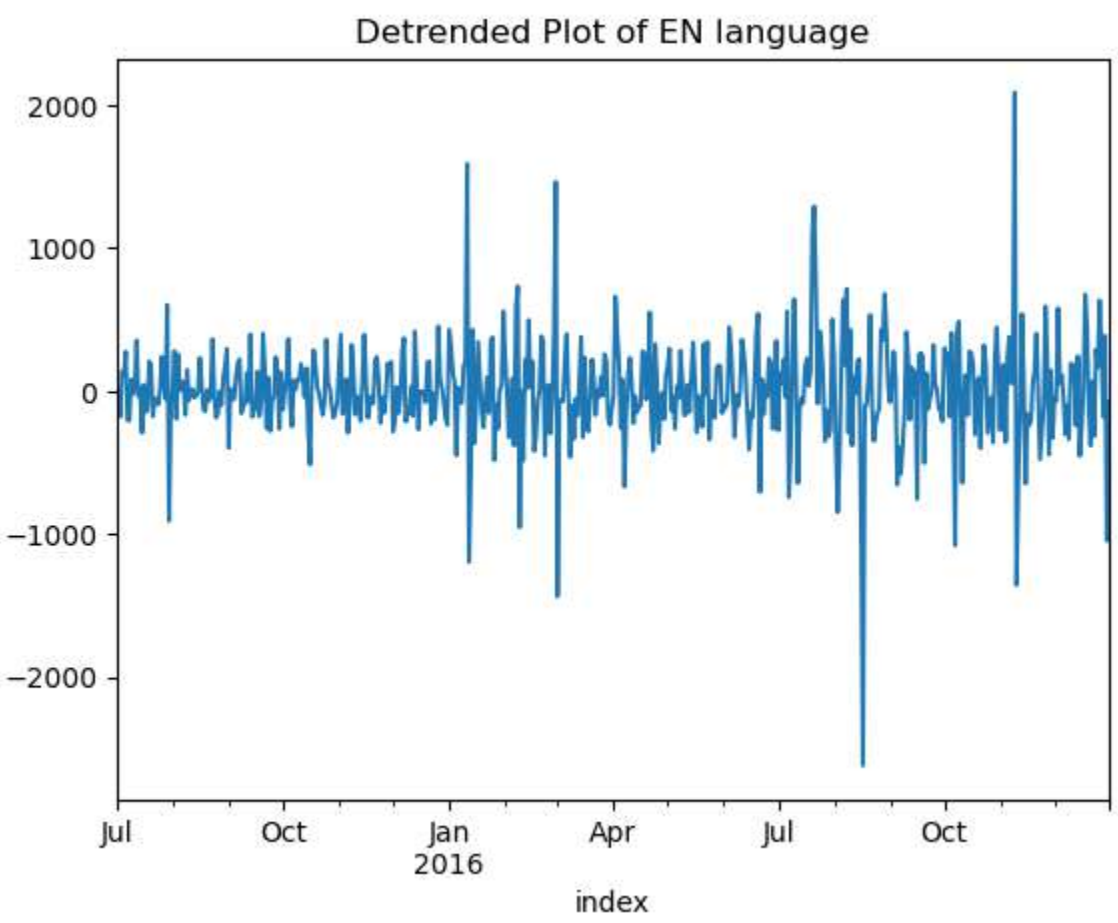


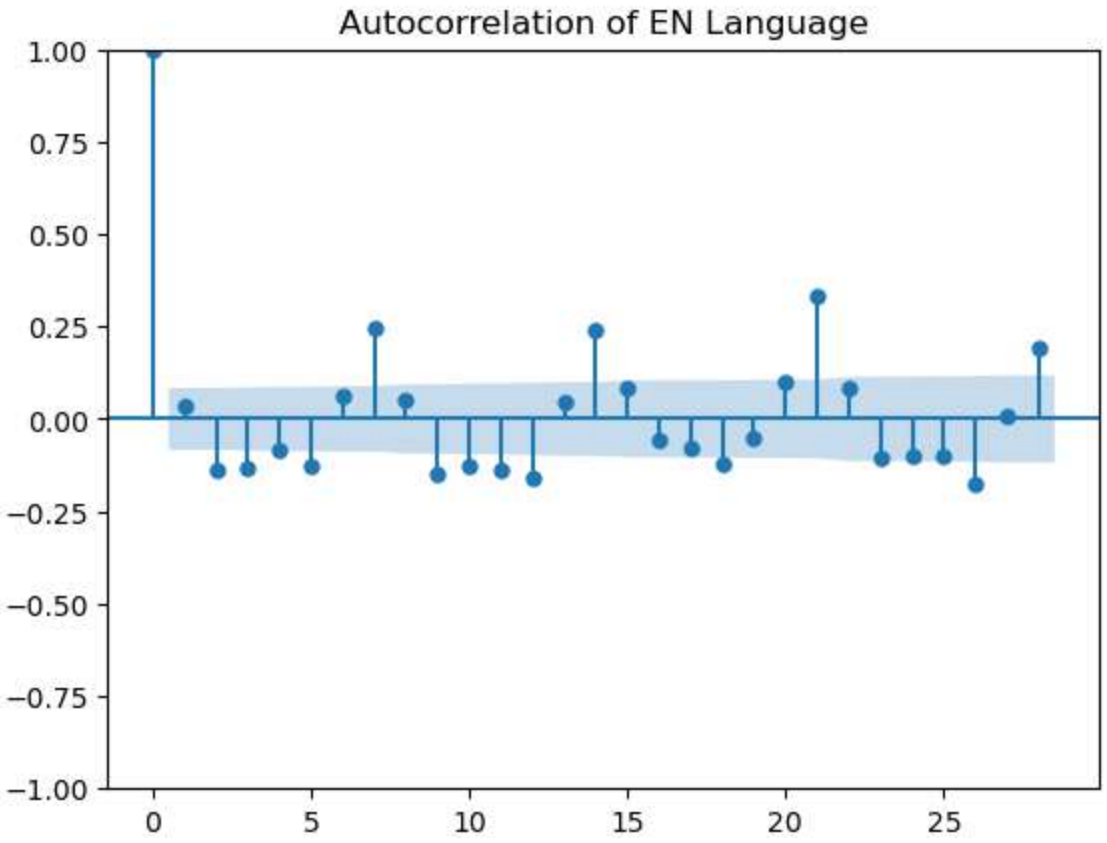


In [110…
```
# JA -> Peak at 7 ,14,21
# P -> 6 Q -> 5
```

In [111…
```
deTrended_EN_lang = get_detrended_deseasonal_data(df['en'],'EN')
```

```
ADF Test after Detrending EN language:('TS is stationary', 5.292042536116286e-13)
```

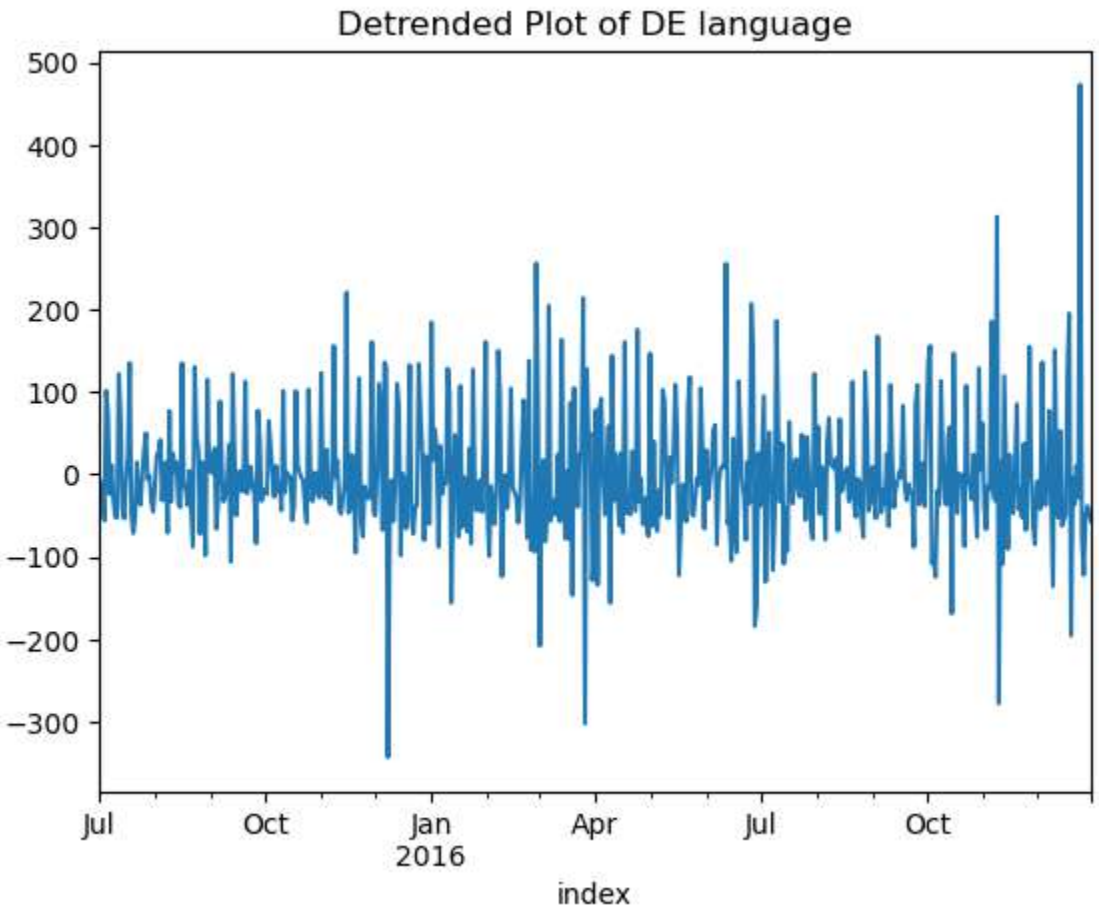


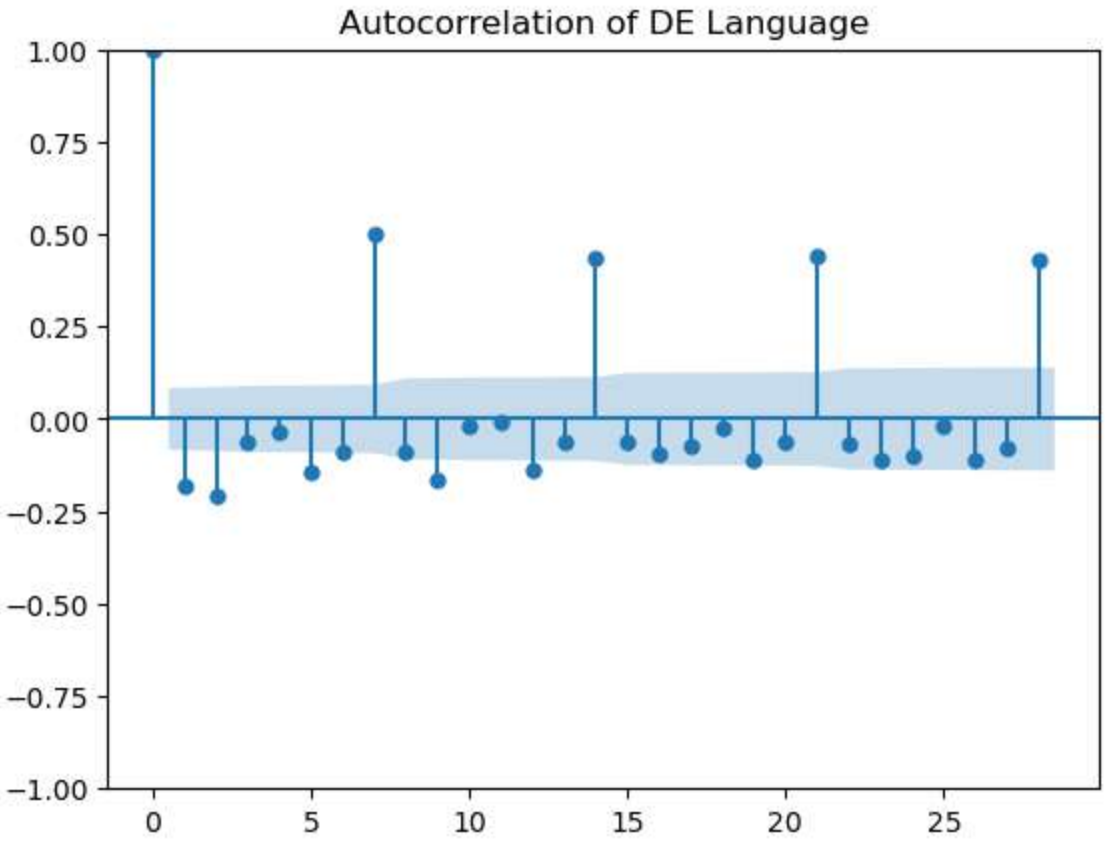


In [112…
```
# EN -> Peak at 7 ,14,21
# P -> 6 Q -> 5
```

In [113…
```
deTrended_DE_lang = get_detrended_deseasonal_data(df['de'],'DE')
```

```
ADF Test after Detrending DE language:('TS is stationary', 2.0718405278634737e-10)
```

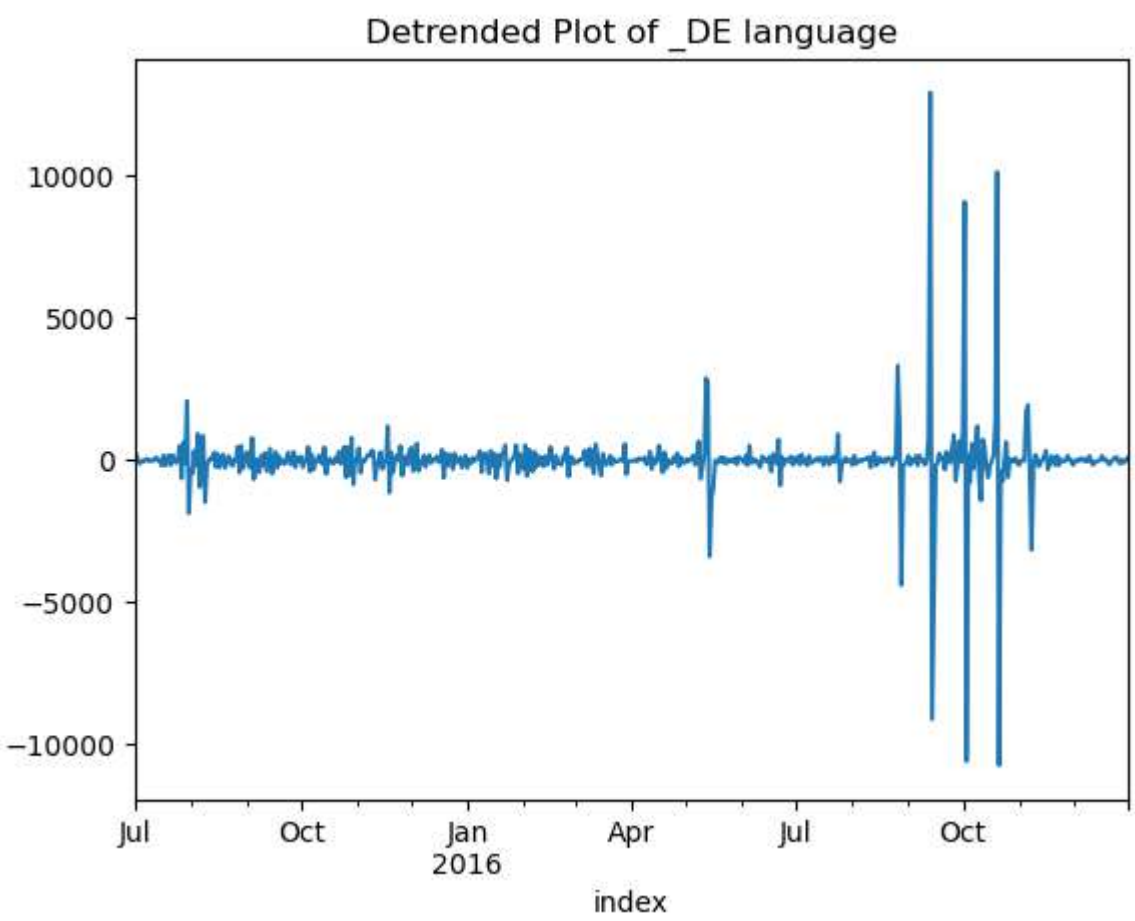


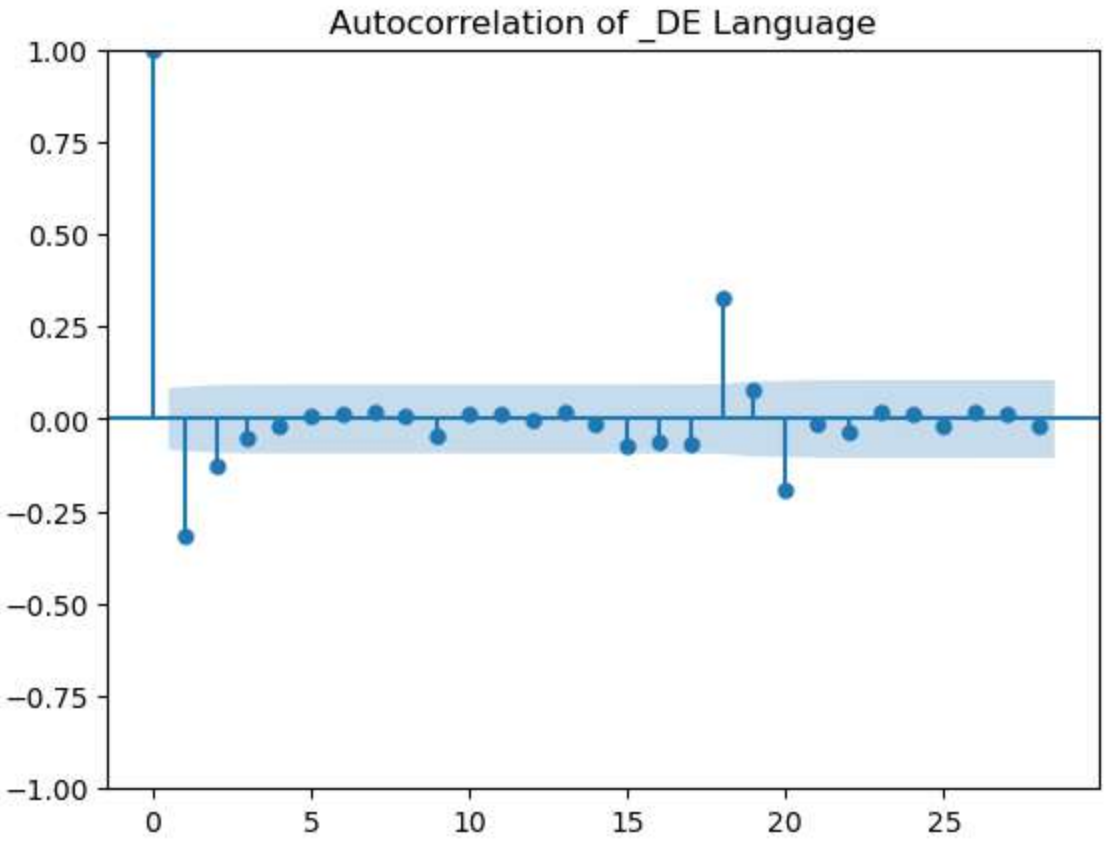


In [114…
```
# DE -> Peak at 7 ,14,21
# P -> 10 Q -> 5
```

In [115…
```
deTrended__DE_lang = get_detrended_deseasonal_data(df['_de'],'_DE')
```

```
ADF Test after Detrending _DE language:('TS is stationary', 1.5961380159824795e-20)
```

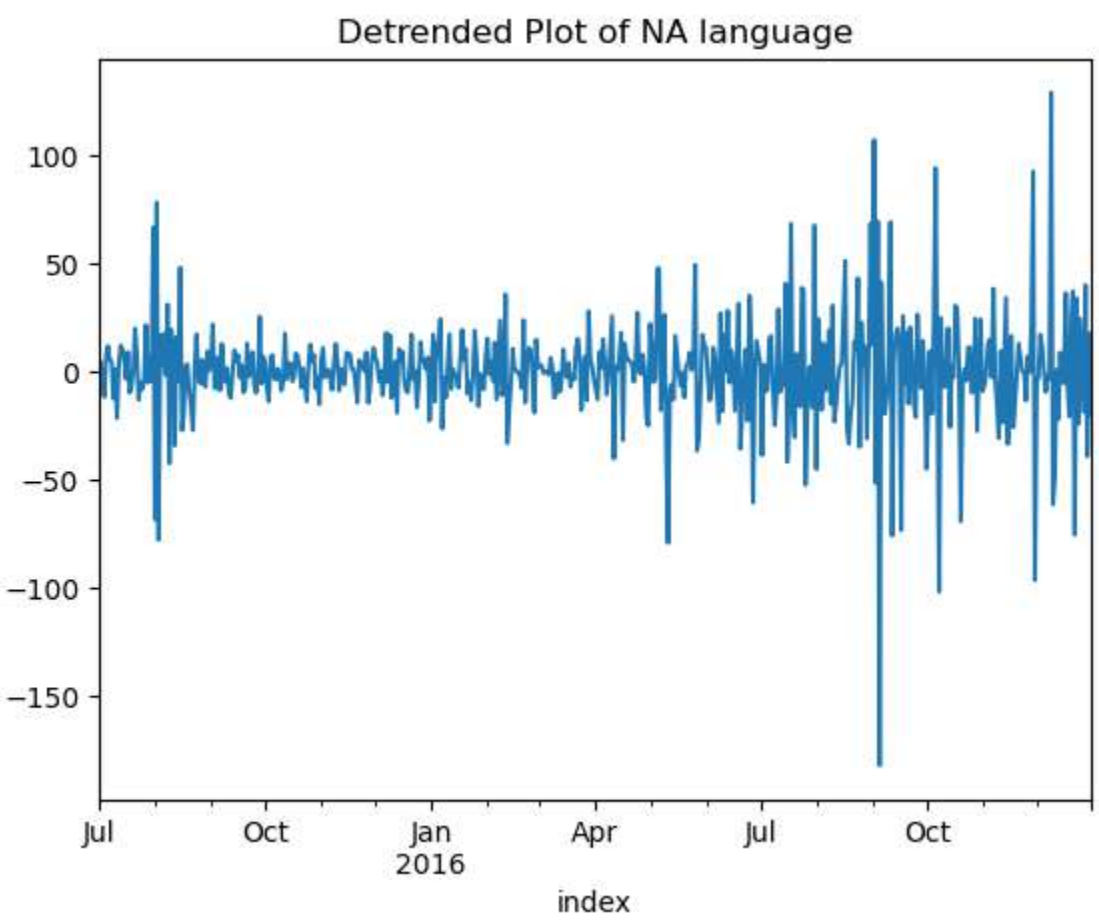


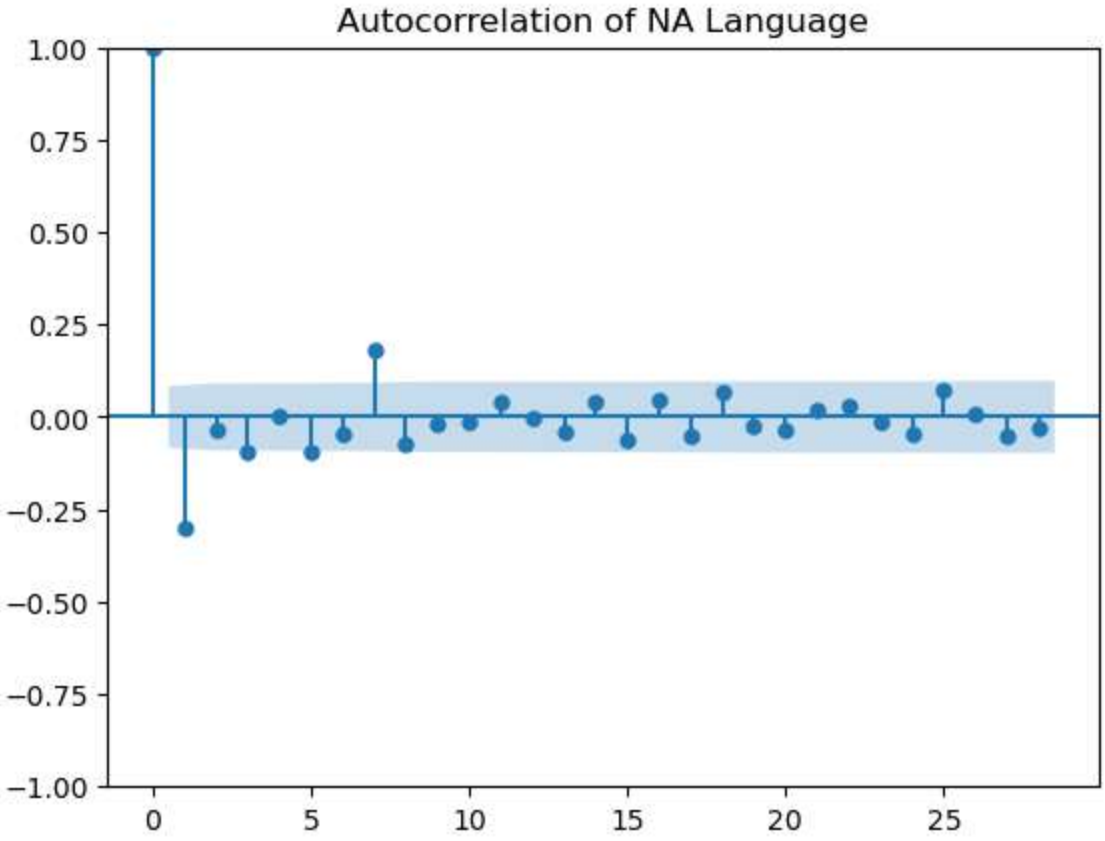


In [116…
```
# _DE -> Peak at 7 ,14,21
# P -> 6 Q -> 2
```

In [117…
```
deTrended_NA_lang = get_detrended_deseasonal_data(df['NA'],'NA')
```

```
ADF Test after Detrending NA language:('TS is stationary', 6.074796247844382e-19)
```





In [118…

```
# NA -> Peak at 7 ,14,21
# P -> 5 Q -> 2
```

Observation

It was observed that Seasonality of data is occuring in every `7` month.

Training model for different languages Using ARIMA Model

In [119…
```
df.head()
```

Out[119]:

index	NA	_de	de	en	es	fr	ja	ru	z
2015-07-01	124.284493	11.0	748.269078	3699.986672	1124.820419	518.275576	623.080703	691.852568	300.51192
2015-07-02	128.301077	192.0	738.514997	3688.549667	1076.750516	521.232645	709.462561	702.874850	300.81690
2015-07-03	123.393474	192.0	709.930981	3509.763578	993.822199	503.030516	644.695993	655.785690	298.91779
2015-07-04	111.330127	80.0	653.911157	3647.536381	934.806317	534.642391	799.631563	620.469613	301.84295
2015-07-05	119.047298	72.0	755.402831	3761.088692	1013.400292	525.623191	768.694764	656.572081	317.26109

In [120…
```
df.shape
```

Out[120]:
```
(550, 9)
```

In [ ]:
```
```

In [121…
```
zh_df = pd.DataFrame(df['zh'])
zh_train_x = zh_df.loc[zh_df.index < zh_df.index[-200]].copy()
zh_test_x = zh_df.loc[zh_df.index >= zh_df.index[-200]].copy()
```

In [122…
```
zh_test_x.shape
```

Out[122]:
```
(200, 1)
```

In [123…
```
zh_train_x.head()
```

Out[123]:

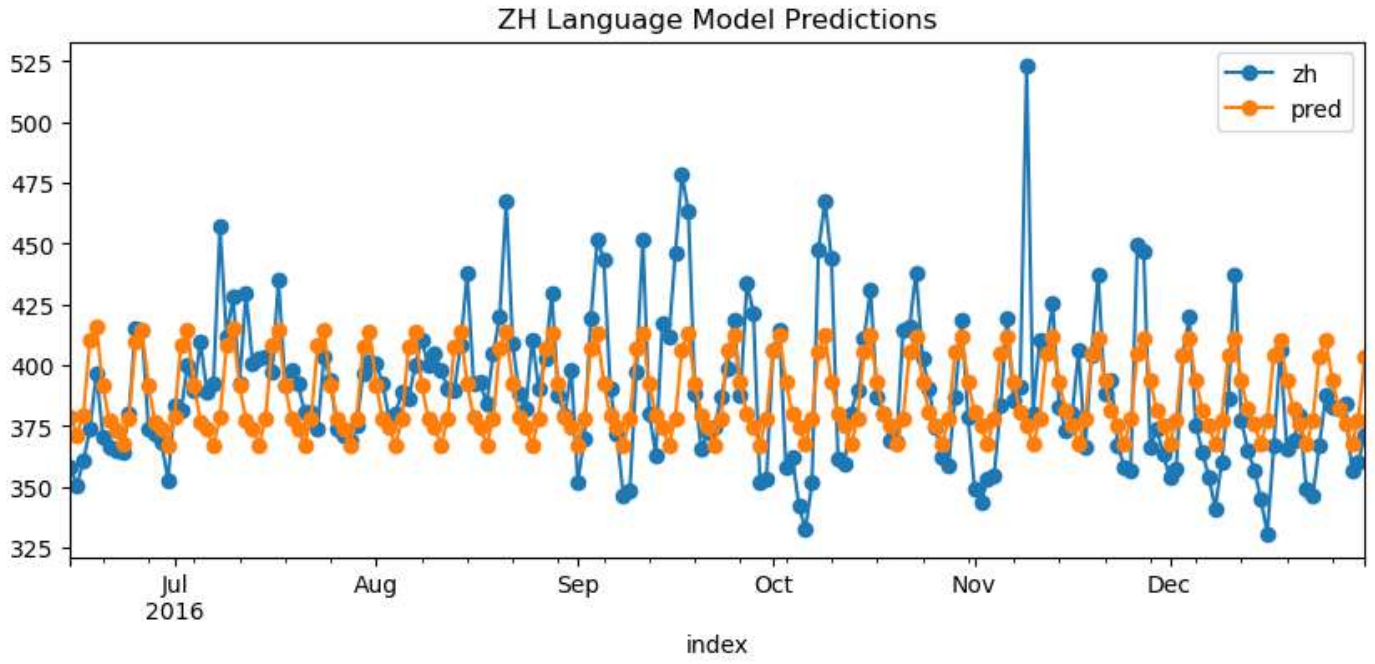
index	zh
2015-07-01	300.511928
2015-07-02	300.816903
2015-07-03	298.917792
2015-07-04	301.842952
2015-07-05	317.261095

In [124…
```
def performance(actual, predicted):
    mape_value = mape(actual, predicted)
    print(f'MAPE :{round(mape_value,3)}')
```

In [125…
```
model = SARIMAX(zh_train_x['zh'],order=[8,1,8])
.fit()
```

```
zh_test_x['pred'] = model.forecast(200)
zh_test_x.plot(style='-o',title='ZH Language Model Predictions',figsize=(10,4))
performance(zh_test_x['zh'],zh_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.048
```



Model for JA Language Code

In [126…
```
ja_df = pd.DataFrame(df['ja'])
ja_train_x = ja_df.loc[ja_df.index < ja_df.index[-200]].copy()
ja_test_x = ja_df.loc[ja_df.index >= ja_df.index[-200]].copy()
```

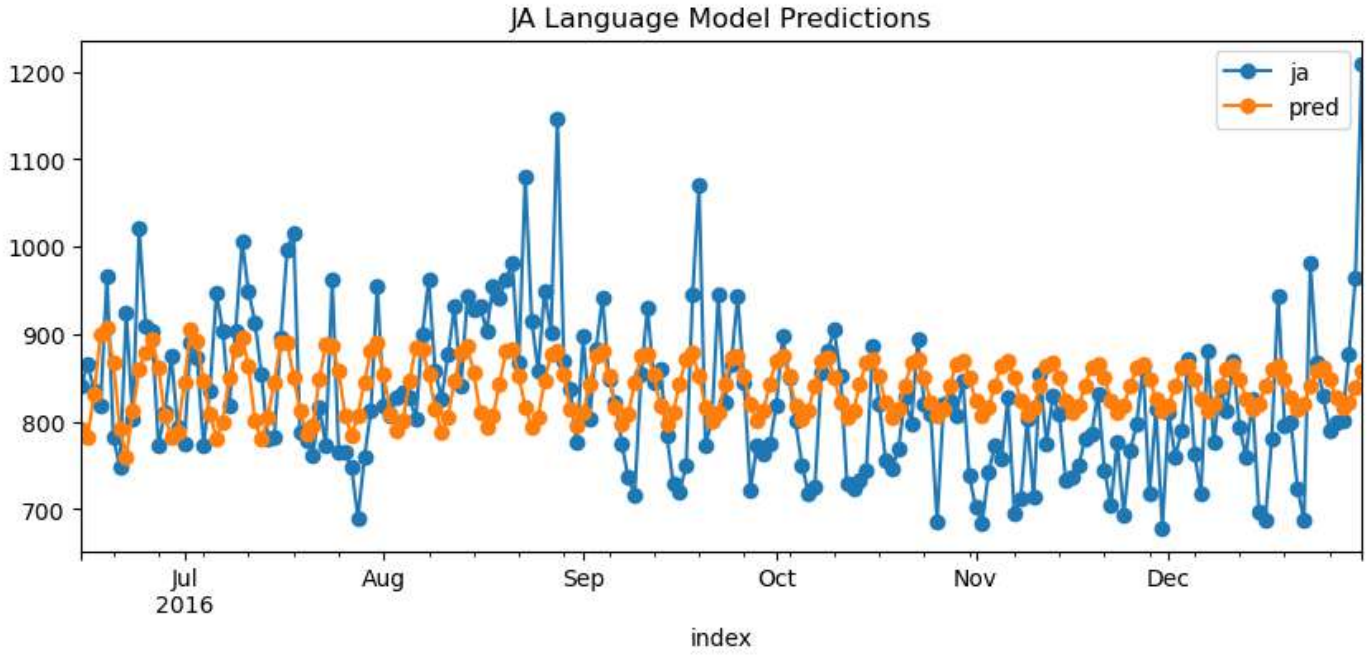
In [127…
```
ja_train_x.shape,ja_test_x.shape
```

Out[127]:
```
((350, 1), (200, 1))
```

In [128…
```
model = SARIMAX(ja_train_x,order=(8,1,8))
model = model.fit()
ja_test_x['pred'] = model.forecast(200)
ja_test_x.plot(style='-o',title='JA Language Model Predictions',figsize=(10,4))
performance(ja_test_x['ja'],ja_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

MAPE :0.079



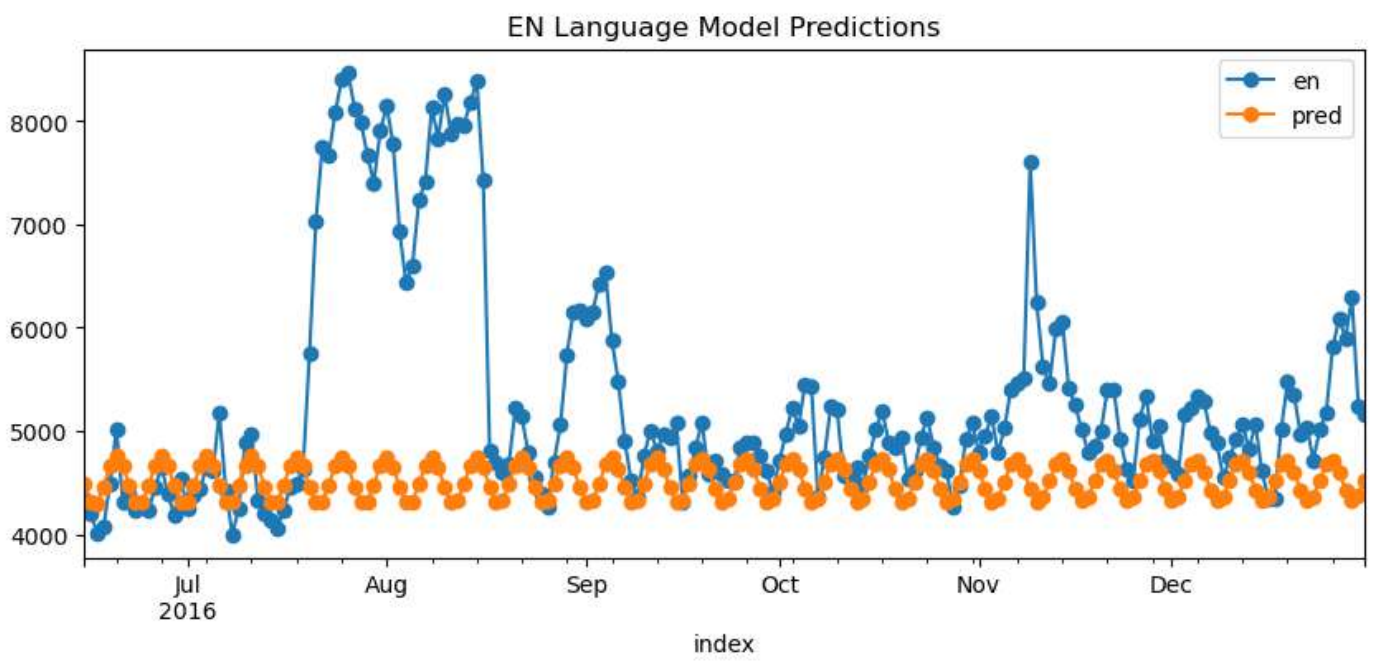
Model EN Language Code

In [129…
```
en_df = pd.DataFrame(df['en'])
en_train_x = en_df.loc[en_df.index < en_df.index[-200]].copy()
en_test_x = en_df.loc[en_df.index >= en_df.index[-200]].copy()
```

In [130…
```
model = SARIMAX(en_train_x,order=(3,1,3))
model = model.fit()
en_test_x['pred'] = model.forecast(200)
en_test_x.plot(style='-o',title='EN Language Model Predictions',figsize=(10,4))
performance(en_test_x['en'],en_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

MAPE :0.133

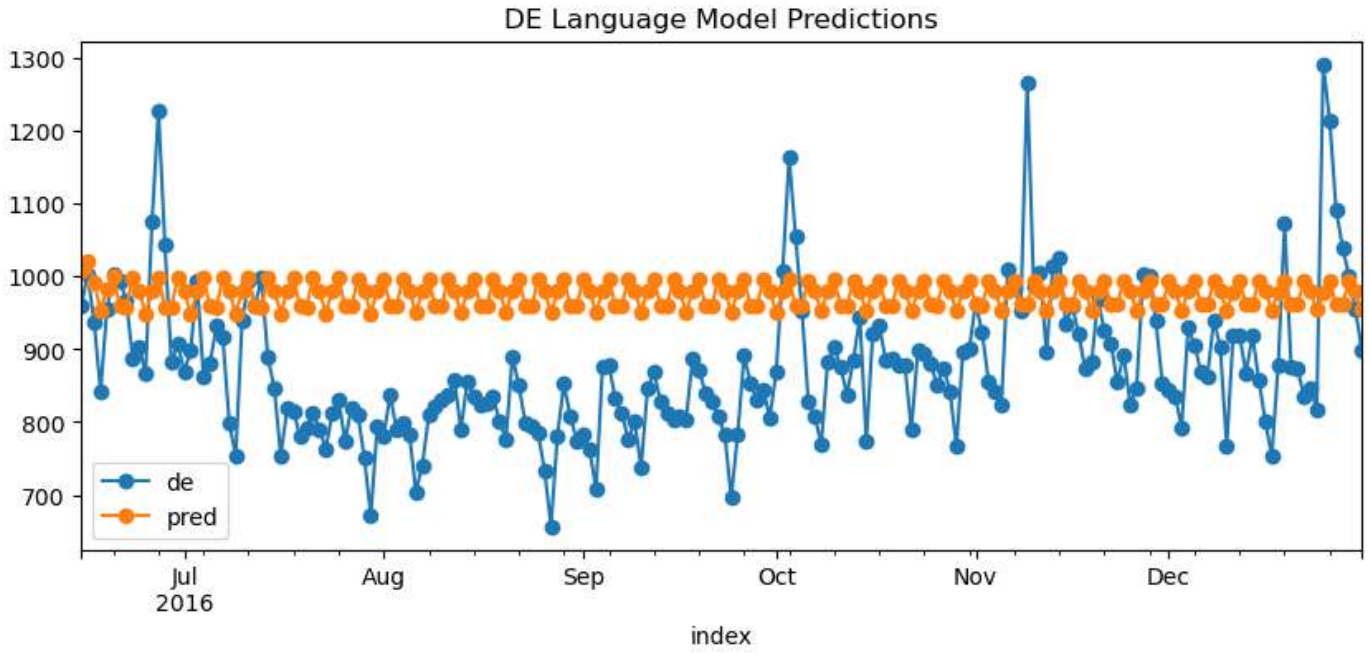


Model DE Language Code

In [131…
```
de_df = pd.DataFrame(df['de'])
de_train_x = de_df.loc[de_df.index < de_df.index[-200]].copy()
de_test_x = de_df.loc[de_df.index >= de_df.index[-200]].copy()
```

In [132…
```
model = SARIMAX(de_train_x,order=(4,1,4))
model = model.fit()
de_test_x['pred'] = model.forecast(200)
de_test_x.plot(style='-o',title='DE Language Model Predictions',figsize=(10,4))
performance(de_test_x['de'],de_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.145
```

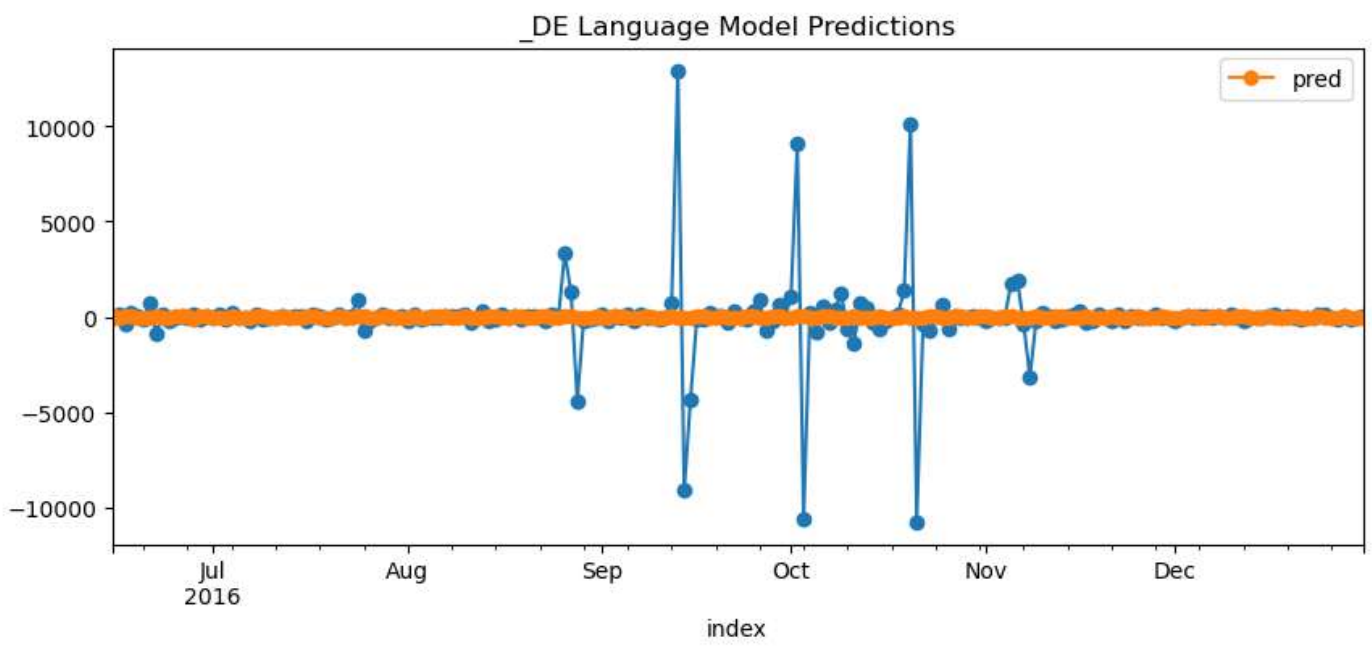


Model _DE Language Code

In [133…
```
_de_df = pd.DataFrame(deTrended__DE_lang)
_de_train_x = _de_df.loc[_de_df.index < _de_df.index[-200]].copy()
_de_test_x  = _de_df.loc[_de_df.index >= _de_df.index[-200]].copy()
```

In [134…
```
model = SARIMAX(_de_train_x,order=(8,1,8))
model = model.fit()
_de_test_x['pred'] = model.forecast(200)
_de_test_x.plot(style='-o',title='_DE Language Model Predictions',figsize=(10,4))
performance(_de_test_x['_de'],_de_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning: Non-invertible starting MA parameters found. Using zeros as starting paramete
rs.
  warn('Non-invertible starting MA parameters found.'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
C:\Users\gaura\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\core.py:807: User
Warning: The label '_de' of <matplotlib.lines.Line2D object at 0x00000295B91B9690> start
s with '_'. It is thus excluded from the legend.
  ax.legend(handles, labels, loc="best", title=title)
MAPE :5787166219623565.0
```

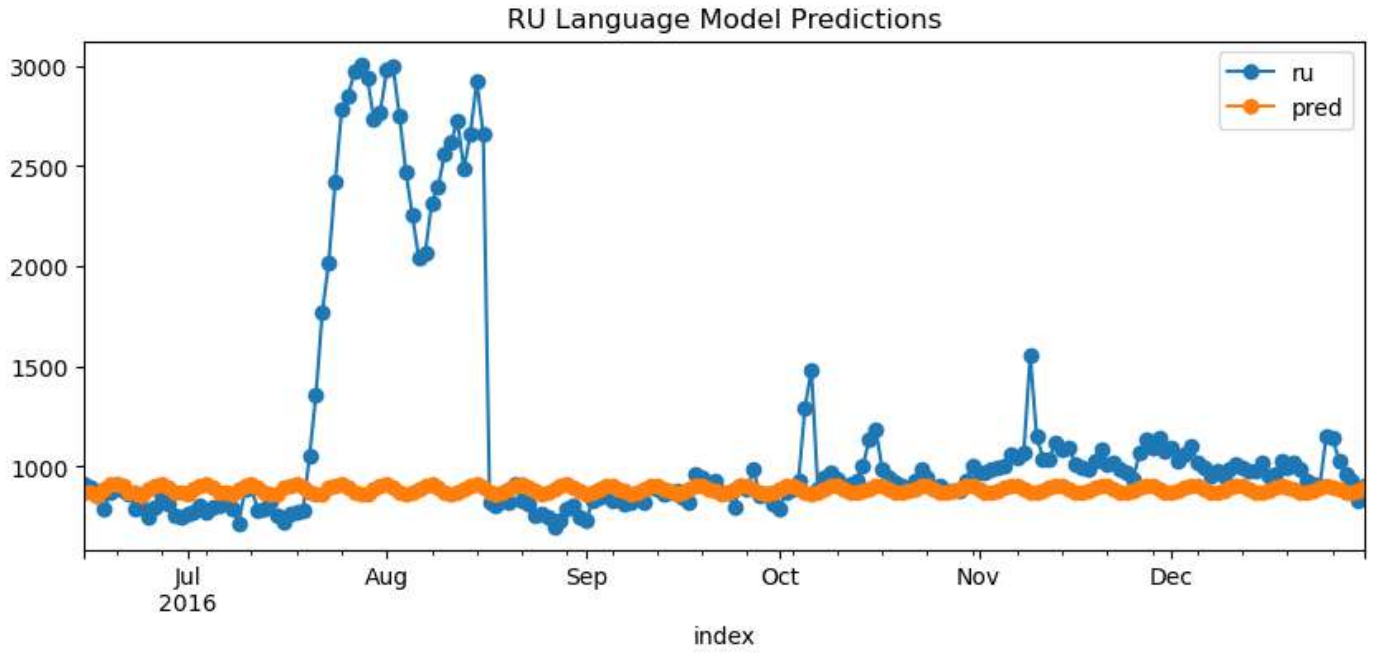


Model RU Language Code

In [135…
```
ru_df = pd.DataFrame(df['ru'])
ru_train_x = ru_df.loc[ru_df.index < ru_df.index[-200]].copy()
ru_test_x  = ru_df.loc[ru_df.index >= ru_df.index[-200]].copy()
```

In [147…
```
model = SARIMAX(ru_train_x,order=(7,1,7))
model = model.fit()
ru_test_x['pred'] = model.forecast(200)
ru_test_x.plot(style='-o',title='RU Language Model Predictions',figsize=(10,4))
performance(ru_test_x['ru'],ru_test_x['pred'])
```

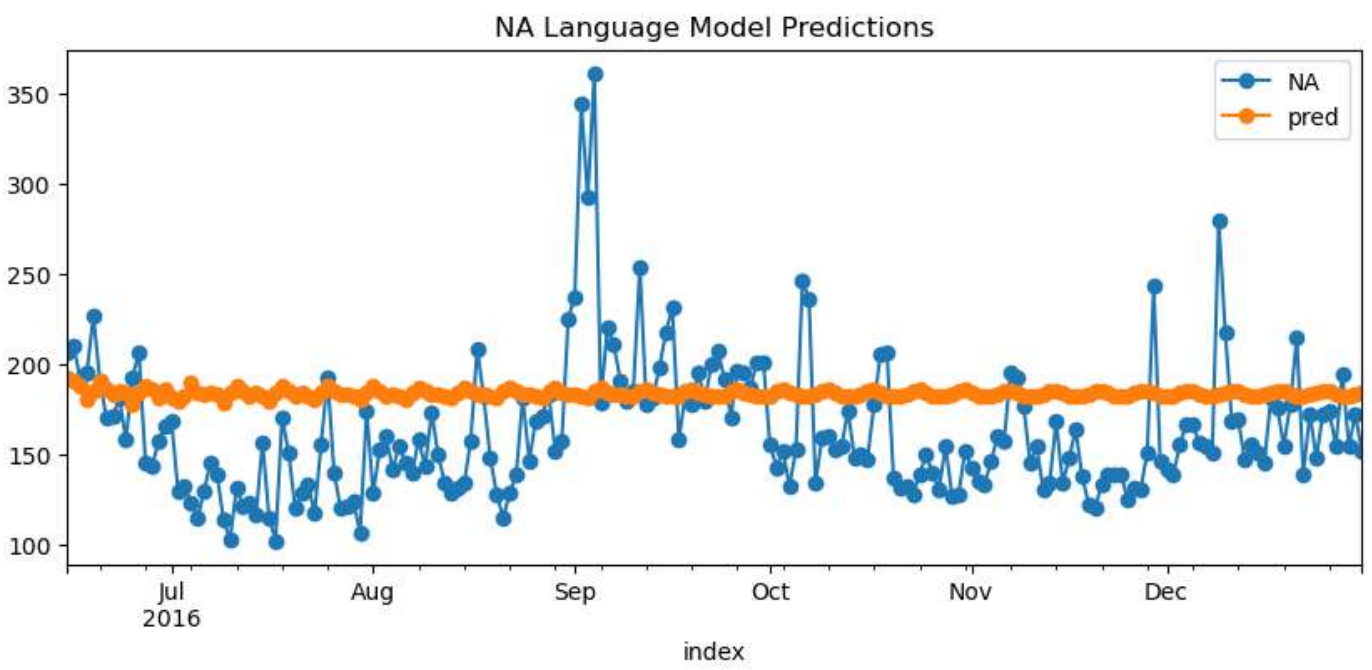
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: Us
erWarning: Non-stationary starting autoregressive parameters found. Using zeros as start
ing parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning: Non-invertible starting MA parameters found. Using zeros as starting paramete
rs.
  warn('Non-invertible starting MA parameters found.'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.174
```



In [137…
```
NA_df = pd.DataFrame(df['NA'])
NA_train_x = NA_df.loc[NA_df.index < NA_df.index[-200]].copy()
NA_test_x  = NA_df.loc[NA_df.index >= NA_df.index[-200]].copy()
```

In [138…
```
model = SARIMAX(NA_train_x,order=(7,1,7))
model = model.fit()
NA_test_x['pred'] = model.forecast(200)
NA_test_x.plot(style='-o',title='NA Language Model Predictions',figsize=(10,4))
performance(NA_test_x['NA'],NA_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: Us
erWarning: Non-stationary starting autoregressive parameters found. Using zeros as start
ing parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning: Non-invertible starting MA parameters found. Using zeros as starting paramete
rs.
  warn('Non-invertible starting MA parameters found.'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.238
```

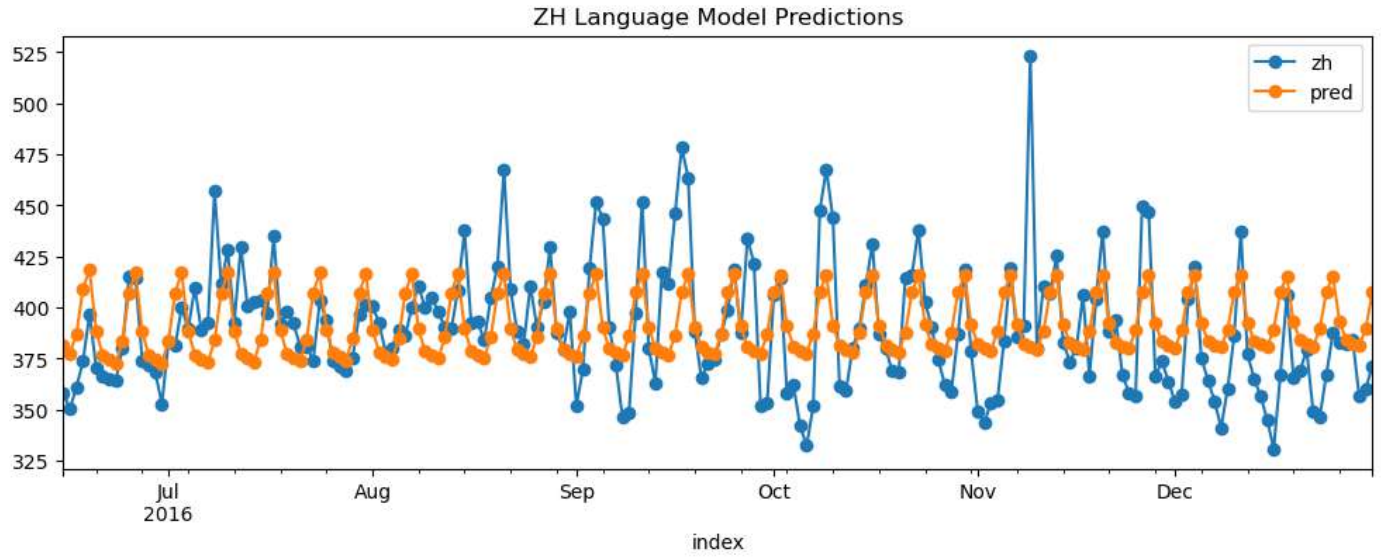


Training model for different languages Using SARIMAX Model

In [139…
```
# Model for ZH Language Code
```

In [140…
```
model = SARIMAX(zh_train_x['zh'],order=(2,1,1),seasonal_order=(2,0,1,7)) # (p,d,q) (P,D,
model = model.fit()
zh_test_x['pred'] = model.forecast(200)
zh_test_x.plot(style='-o',title='ZH Language Model Predictions',figsize=(12,4))
performance(zh_test_x['zh'],zh_test_x['pred'])
```

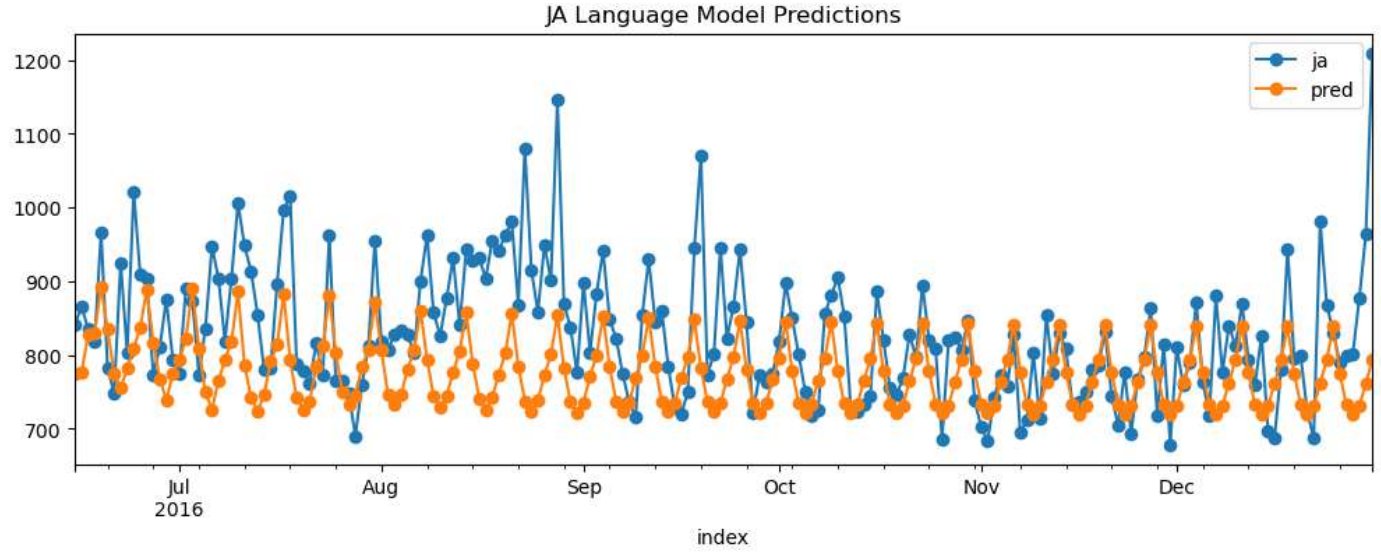
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning: Non-invertible starting MA parameters found. Using zeros as starting paramete
rs.
  warn('Non-invertible starting MA parameters found.'
MAPE :0.05
```



In [141…
```
# Model for JA Language Code
```

In [142…
```
model = SARIMAX(ja_train_x['ja'],order=(2,0,0),seasonal_order=(7,1,1,7)) # (p,d,q) (P,D,
model = model.fit()
ja_test_x['pred'] = model.forecast(200)
ja_test_x.plot(style='-o',title='JA Language Model Predictions',figsize=(12,4))
performance(ja_test_x['ja'],ja_test_x['pred'])
```

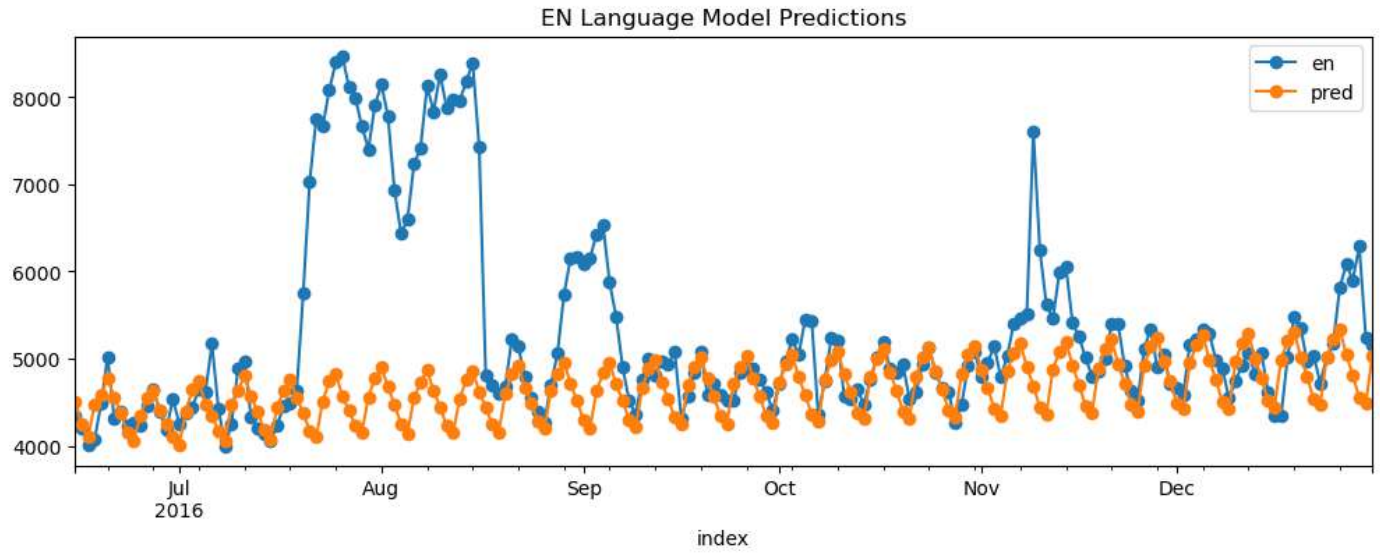
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
MAPE :0.079
```



In [143…
```
# Model for EN Language Code
```

In [144…
```
model = SARIMAX(en_train_x['en'],order=(2,0,3),seasonal_order=(7,2,1,7)) # (p,d,q) (P,D,
model = model.fit()
en_test_x['pred'] = model.forecast(200)
en_test_x.plot(style='-o',title='EN Language Model Predictions',figsize=(12,4))
performance(en_test_x['en'],en_test_x['pred'])
```

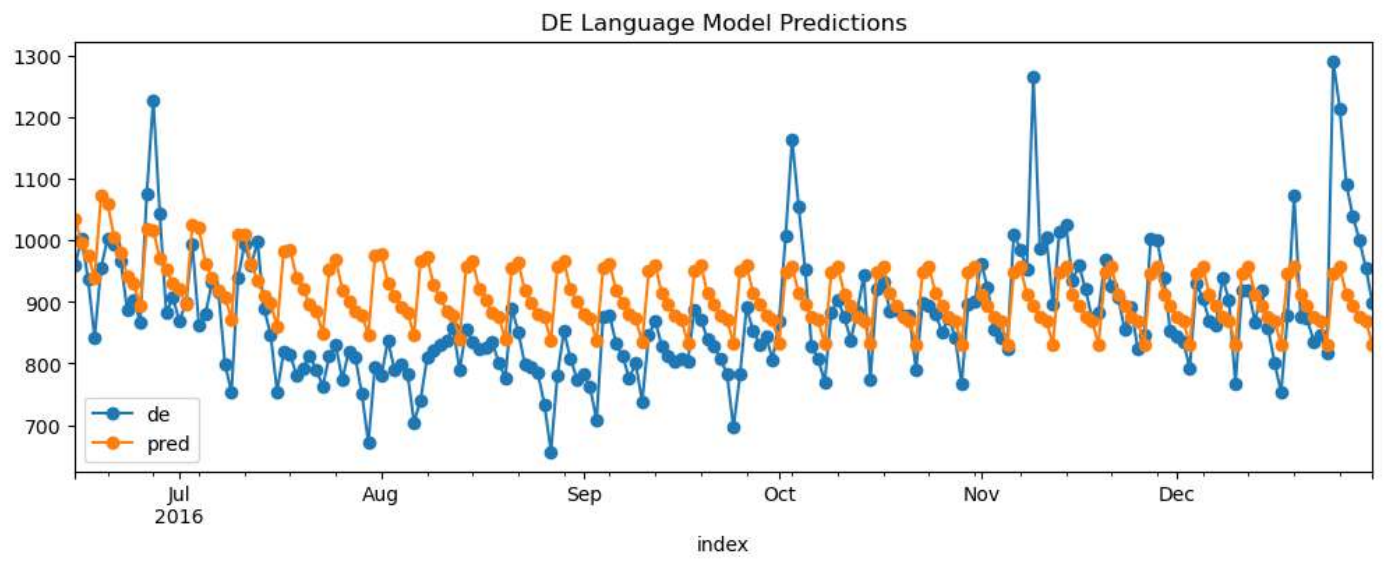
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: Us
erWarning: Non-stationary starting autoregressive parameters found. Using zeros as start
ing parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning: Non-invertible starting MA parameters found. Using zeros as starting paramete
rs.
  warn('Non-invertible starting MA parameters found.'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.112
```



In [148…
```
# Model for DE Language Code
```

In [146…
```
model = SARIMAX(de_train_x['de'],order=(3,0,0),seasonal_order=(7,1,1,7)) # (p,d,q) (P,D,
model = model.fit()
de_test_x['pred'] = model.forecast(200)
de_test_x.plot(style='-o',title='DE Language Model Predictions',figsize=(12,4))
performance(de_test_x['de'],de_test_x['pred'])
```

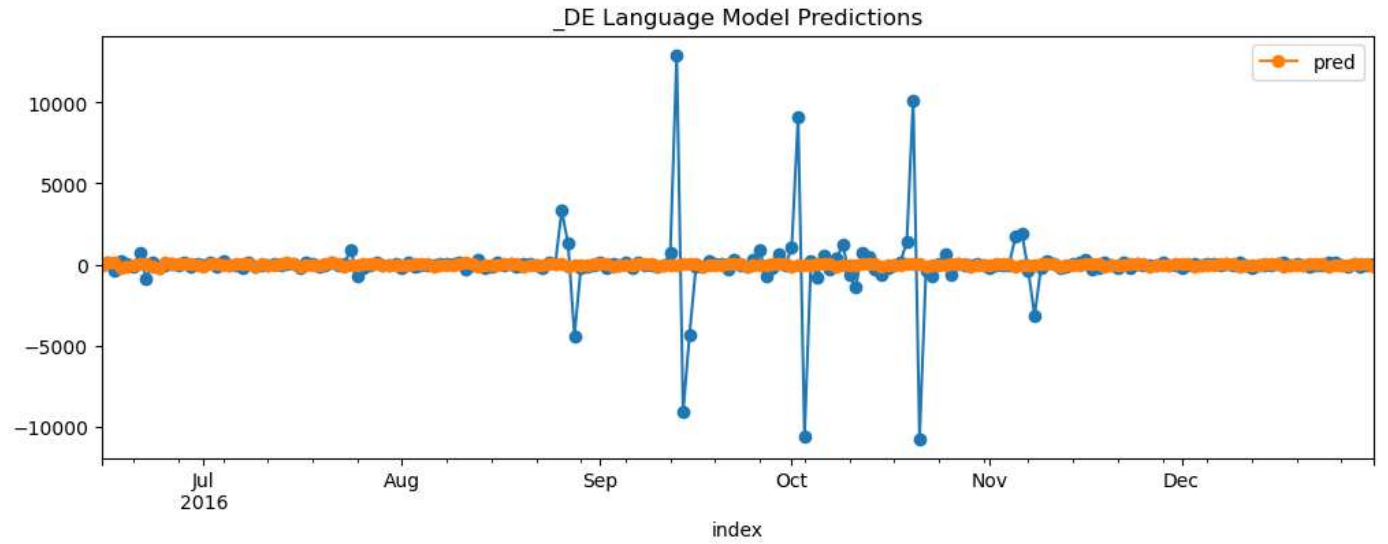
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
MAPE :0.087
```



In [151…
```
# Model for _DE Language Code
```

In [156…
```
model = SARIMAX(_de_train_x['_de'],order=(3,0,0),seasonal_order=(7,1,1,7)) # (p,d,q) (P,
model = model.fit()
_de_test_x['pred'] = model.forecast(200)
_de_test_x.plot(style='-o',title='_DE Language Model Predictions',figsize=(12,4))
performance(_de_test_x['_de'],_de_test_x['pred'])
```

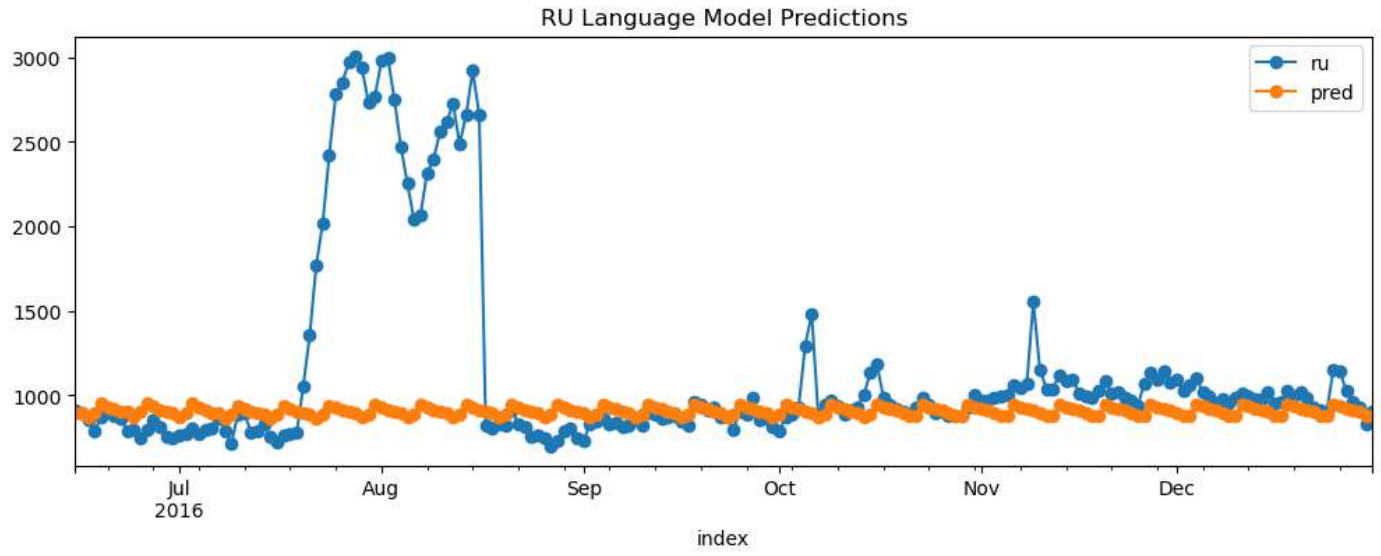
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\core.py:807: User
Warning: The label '_de' of <matplotlib.lines.Line2D object at 0x00000295E7C87550> start
s with '_'. It is thus excluded from the legend.
  ax.legend(handles, labels, loc="best", title=title)
MAPE :7218950574980631.0
```



In [157…
```
# Model RU Language Code
```

In [165…
```
model = SARIMAX(ru_train_x['ru'],order=(2,0,1),seasonal_order=(7,1,1,7)) # (p,d,q) (P,D,
model = model.fit()
ru_test_x['pred'] = model.forecast(200)
ru_test_x.plot(style='-o',title='RU Language Model Predictions',figsize=(12,4))
performance(ru_test_x['ru'],ru_test_x['pred'])
```

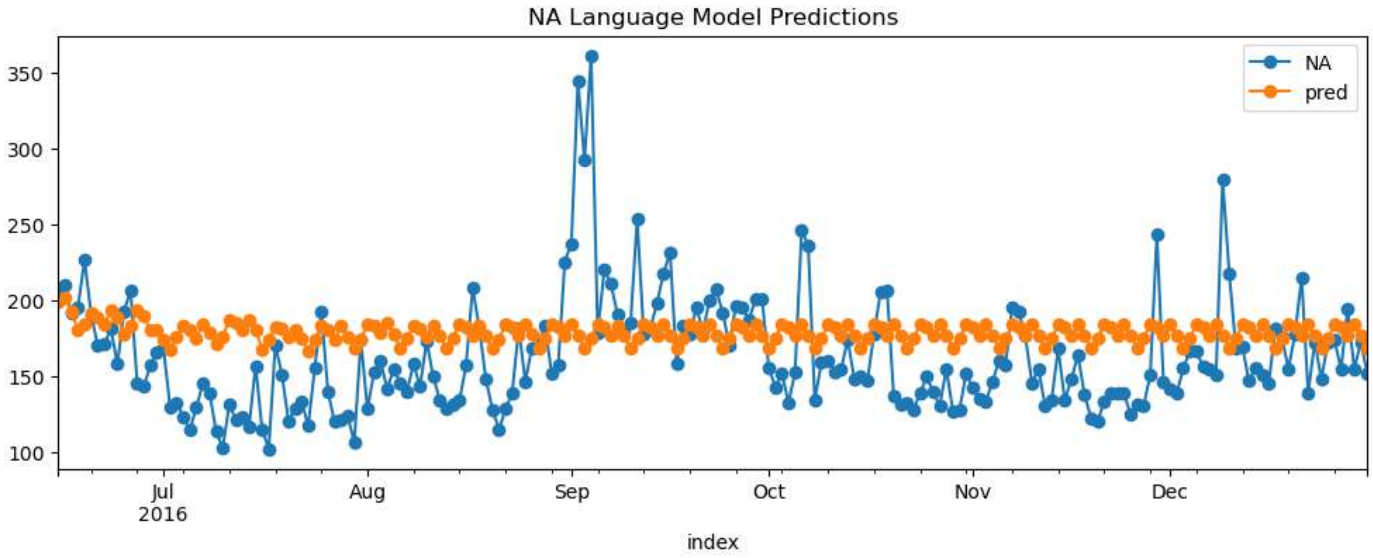
```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: Us
erWarning: Non-stationary starting autoregressive parameters found. Using zeros as start
ing parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: Us
erWarning: Non-invertible starting MA parameters found. Using zeros as starting paramete
rs.
  warn('Non-invertible starting MA parameters found.'
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.172
```



In [171…
```
# Model NA Language Code
```

In [183…
```
model = SARIMAX(NA_train_x['NA'],order=(2,0,0),seasonal_order=(7,1,3,7)) # (p,d,q) (P,D,
model = model.fit()
NA_test_x['pred'] = model.forecast(200)
NA_test_x.plot(style='-o',title='NA Language Model Predictions',figsize=(12,4))
performance(NA_test_x['NA'],NA_test_x['pred'])
```

```
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueW
arning: No frequency information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
C:\Users\gaura\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWar
ning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MAPE :0.214
```



Questions

1. Defining the problem statements and where can this and modifications of this be used?

Ans : The problem statement was all about getting the forecast of pageview for different language wikipedia page. This can be utilize to understand which language type page is prefered more depends on region.

1. Write 3 inferences you made from the data visualizations ?

Ans : Inference is as folows:

1. Most prefered language is english as it was observed that avg pageview are more as compare to other languages.
2. It was observed that there is significant decrease in pageviews for Spanish language in between January to March 2016.
3. It was observed that there is significant Increase in pageviews for French language in between April to May 2016.

1. What does the decomposition of series do?

Ans: Decompositions of series will showcase below mentioned points.

```
1. Trend
2. Seasonality
3. Residual
```

1. What level of differencing gave you a stationary series?

Ans : Level of differencing is contextual in this case it gave me staitionary series with just single differencing.

1. Difference between arima, sarima & sarimax.

Ans : The Basic difference between arima and Sarima is that arima does'nt take care of Seasonal part but re of it. The differnce between arima, sarima and Sarimax is both arima and sarima does'nt

deal with external variable but sarimax does.

1. What other methods other than grid search would be suitable to get the model for all languages?

Ans: We can utilize iterative approach instead of grid search.

In [ ]: