A Restaurant Visitors dataset.The data is about daily visitors to four restaurants located in the United States,subject to American holidays.We will see how the exogenous variable (holidays) affects patronage.

source: (https://www.kaggle.com/c/recruit-restaurant-visitors-forecasting)

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
df=pd.read_csv(r'C:\Users\chumj\Downloads\RestVisitors.csv',index_col='date',parse_dates=True)
df.index.freq='D'
```

In [3]:

```
df
```

Out[3]:

	weekday	holiday	holiday_name	rest1	rest2	rest3	rest4	total
date								
2016-01-01	Friday	1	New Year's Day	65.0	25.0	67.0	139.0	296.0
2016-01-02	Saturday	0	na	24.0	39.0	43.0	85.0	191.0
2016-01-03	Sunday	0	na	24.0	31.0	66.0	81.0	202.0
2016-01-04	Monday	0	na	23.0	18.0	32.0	32.0	105.0
2016-01-05	Tuesday	0	na	2.0	15.0	38.0	43.0	98.0
...	...	...	...	...	...	...	...	...
2017-05-27	Saturday	0	na	NaN	NaN	NaN	NaN	NaN
2017-05-28	Sunday	0	na	NaN	NaN	NaN	NaN	NaN
2017-05-29	Monday	1	Memorial Day	NaN	NaN	NaN	NaN	NaN
2017-05-30	Tuesday	0	na	NaN	NaN	NaN	NaN	NaN
2017-05-31	Wednesday	0	na	NaN	NaN	NaN	NaN	NaN

517 rows × 8 columns

In [4]:

```
#some of data are missing(NaN),for the restuarants but the Holiday col still show holiday or not.
#Lets drop all the missing data
df_1=df.dropna()
```

In [5]:

```
df_1
```

Out[5]:

	weekday	holiday	holiday_name	rest1	rest2	rest3	rest4	total
date								
2016-01-01	Friday	1	New Year's	65.0	25.0	67.0	139.0	296.0

date	weekday	holiday	holiday_name	rest1	rest2	rest3	rest4	total
2016-01-01	Friday	1	Day	65.0	25.0	67.0	139.0	296.0
2016-01-02	Saturday	0	na	24.0	39.0	43.0	85.0	191.0
2016-01-03	Sunday	0	na	24.0	31.0	66.0	81.0	202.0
2016-01-04	Monday	0	na	23.0	18.0	32.0	32.0	105.0
2016-01-05	Tuesday	0	na	2.0	15.0	38.0	43.0	98.0
...	...	...	...	...	...	...	...	...
2017-04-18	Tuesday	0	na	30.0	30.0	13.0	18.0	91.0
2017-04-19	Wednesday	0	na	20.0	11.0	30.0	18.0	79.0
2017-04-20	Thursday	0	na	22.0	3.0	19.0	46.0	90.0
2017-04-21	Friday	0	na	38.0	53.0	36.0	38.0	165.0
2017-04-22	Saturday	0	na	97.0	20.0	50.0	59.0	226.0

478 rows × 8 columns

In [7]:

```
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 478 entries, 2016-01-01 to 2017-04-22
Freq: D
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   weekday       478 non-null    object
 1   holiday       478 non-null    int64
 2   holiday_name  478 non-null    object
 3   rest1         478 non-null    float64
 4   rest2         478 non-null    float64
 5   rest3         478 non-null    float64
 6   rest4         478 non-null    float64
 7   total         478 non-null    float64
dtypes: float64(5), int64(1), object(2)
memory usage: 33.6+ KB
```

In [8]:

```
df_1.columns
```

Out[8]:

```
Index(['weekday', 'holiday', 'holiday_name', 'rest1', 'rest2', 'rest3',
       'rest4', 'total'],
      dtype='object')
```

In [9]:

```
#In other to make our model work,we need change all float into interger.
x=['rest1', 'rest2', 'rest3','rest4', 'total']
for col in x:
    df_1[col]=df_1[col].astype(int)
```

```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

In [10]:

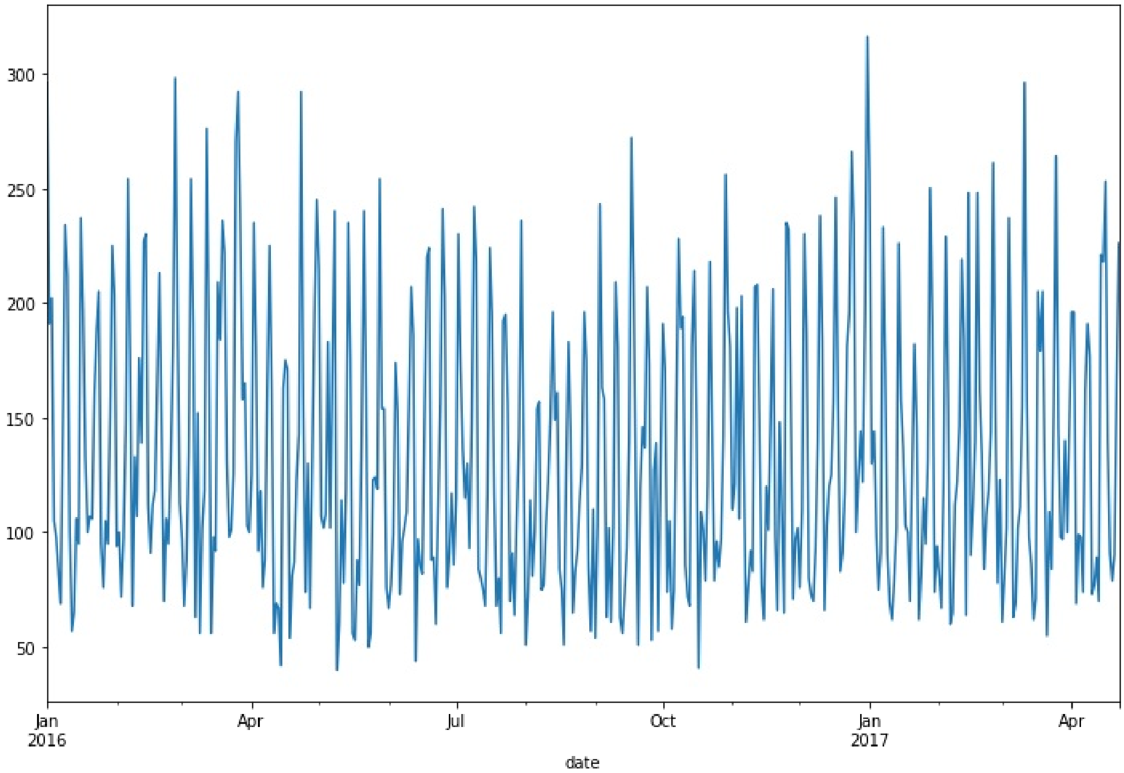
```
df_1.head(3)
```

Out[10]:

date	weekday	holiday	holiday_name	rest1	rest2	rest3	rest4	total
2016-01-01	Friday	1	New Year's Day	65	25	67	139	296
2016-01-02	Saturday	0	na	24	39	43	85	191
2016-01-03	Sunday	0	na	24	31	66	81	202

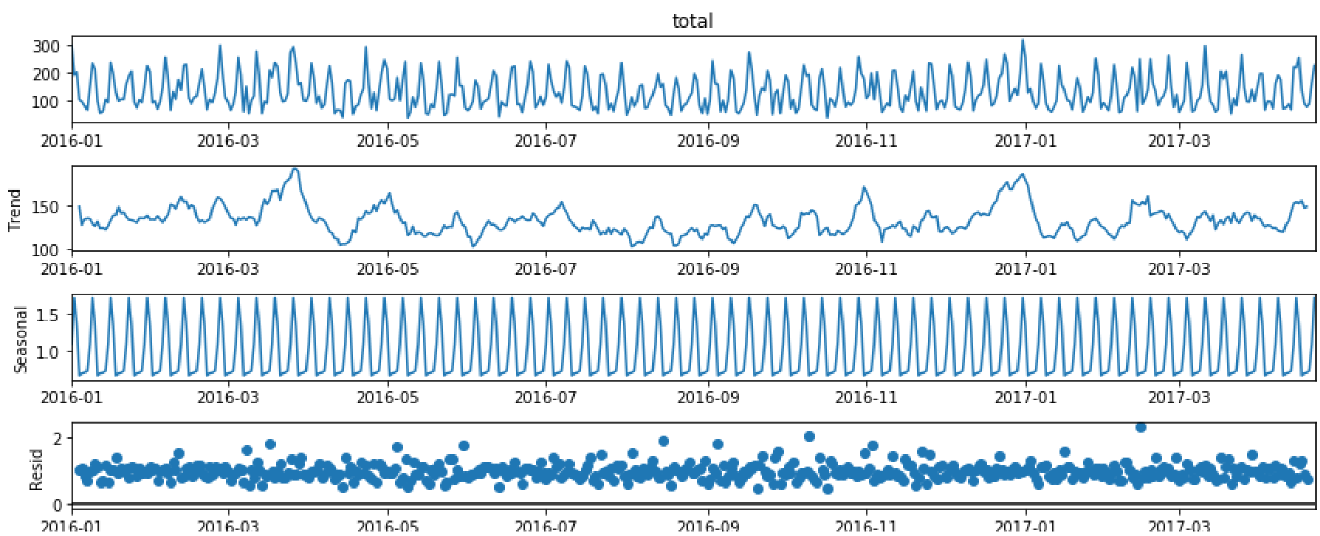
In [11]:

```
#our focus is will mainly be the ['total']
df_1['total'].plot(figsize=(12,8));
```



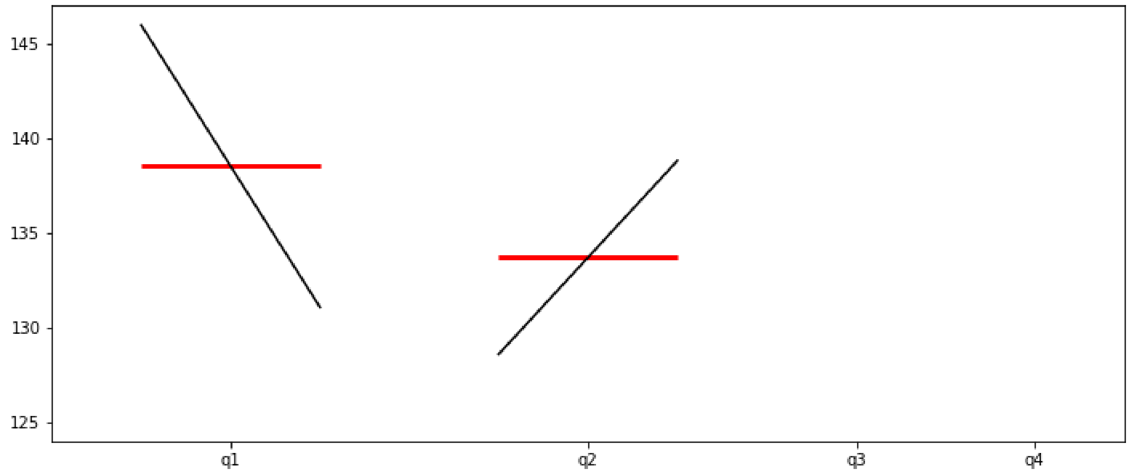
In [12]:

```
#Applying ETS decomposition
from statsmodels.tsa.seasonal import seasonal_decompose
result=seasonal_decompose(df_1['total'],model='mul')
from pylab import rcParams
rcParams['figure.figsize']=12,5
result.plot();
```



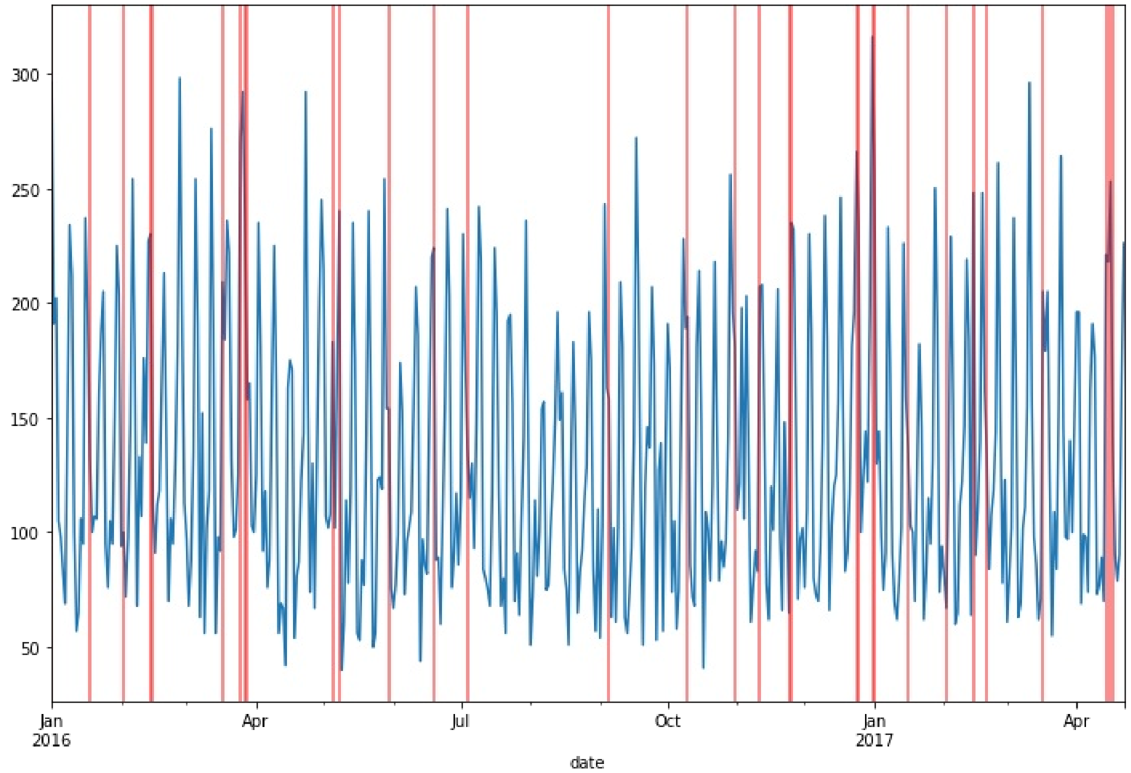
In [13]:

```
#Trying to expose Seasonality with Months and Quarter Plots
from statsmodels.graphics.tsaplots import quarter_plot
dfq=df_1['total'].resample(rule='Q').mean()
quarter_plot(dfq);
```



In [14]:

```
# Since holidays is acting as our Exogenous variable,let fit it into our plot
ax=df_1['total'].plot(figsize=(12,8))
for x in df_1.query('holiday==1').index:
    ax.axvline(x=x,color='r',alpha=0.6);
```



In [15]:

```
#Testing for stationarity using the augmented Dickey Fuller Test
from statsmodels.tsa.stattools import adfuller

def adf_test(series,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
```

```
    """
    print(f'Augmented Dickey-Fuller Test: {title}')
    result = adfuller(series.dropna(),autolag='AIC')

    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)

    for key,val in result[4].items():
        out[f'critical value ({key})']=val

    print(out.to_string())

    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis")
        print("Reject the null hypothesis")
        print("Data has no unit root and is stationary")
    else:
        print("Weak evidence against the null hypothesis")
        print("Fail to reject the null hypothesis")
        print("Data has a unit root and is non-stationary")
```

In [16]:

```
adf_test(df_1['total'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic        -5.592497
p-value                    0.000001
# lags used               18.000000
# observations           459.000000
critical value (1%)       -3.444677
critical value (5%)       -2.867857
critical value (10%)      -2.570135
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

In [33]:

```
#Running auto_arima to obtain the best or recommended orders
from pmdarima import auto_arima
au=auto_arima(df_1['total'],seasonl=True,m=7).summary()
import warnings
warnings.filterwarnings('ignore')
```

In [34]:

```
au
```

Out[34]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	478
Model:	SARIMAX(1, 0, [1], 7)	**Log Likelihood**	-2387.105
Date:	Fri, 04 Sep 2020	**AIC**	4782.211
Time:	23:12:15	**BIC**	4798.889
Sample:	0	**HQIC**	4788.768
	- 478		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	5.9554	2.023	2.943	0.003	1.990	9.921
ar.S.L7	0.9543	0.015	62.493	0.000	0.924	0.984
ma.S.L7	-0.7330	0.054	-13.532	0.000	-0.839	-0.627
sigma2	1318.0895	84.030	15.686	0.000	1153.394	1482.785

Ljung-Box (Q):	72.85	**Jarque-Bera (JB):**	58.97
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.86	**Skew:**	0.73
Prob(H) (two-sided):	0.33	**Kurtosis:**	3.91

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [19]:

```
stepwise_fit=auto_arima(df_1['total'],start_p=0,start_q=0,max_p=3,max_q=3,m=7,seasonal=True,
                        trace=True,stepwise=True,error_action='ignore')
import warnings
warnings.filterwarnings('ignore')
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(1,0,1)[7] intercept   : AIC=4782.211, Time=1.79 sec
 ARIMA(0,0,0)(0,0,0)[7] intercept   : AIC=5269.484, Time=0.04 sec
 ARIMA(1,0,0)(1,0,0)[7] intercept   : AIC=4916.749, Time=1.21 sec
 ARIMA(0,0,1)(0,0,1)[7] intercept   : AIC=5049.644, Time=0.94 sec
 ARIMA(0,0,0)(0,0,0)[7]             : AIC=6126.084, Time=0.01 sec
 ARIMA(0,0,0)(0,0,1)[7] intercept   : AIC=5093.130, Time=0.45 sec
 ARIMA(0,0,0)(1,0,0)[7] intercept   : AIC=4926.360, Time=0.89 sec
 ARIMA(0,0,0)(2,0,1)[7] intercept   : AIC=inf, Time=nan sec
 ARIMA(0,0,0)(1,0,2)[7] intercept   : AIC=4991.110, Time=2.88 sec
 ARIMA(0,0,0)(0,0,2)[7] intercept   : AIC=5010.582, Time=1.17 sec
 ARIMA(0,0,0)(2,0,0)[7] intercept   : AIC=4859.638, Time=4.07 sec
 ARIMA(0,0,0)(2,0,2)[7] intercept   : AIC=inf, Time=4.81 sec
 ARIMA(1,0,0)(1,0,1)[7] intercept   : AIC=4833.921, Time=1.97 sec
 ARIMA(0,0,1)(1,0,1)[7] intercept   : AIC=inf, Time=2.01 sec
 ARIMA(1,0,1)(1,0,1)[7] intercept   : AIC=inf, Time=2.59 sec
 ARIMA(0,0,0)(1,0,1)[7]             : AIC=inf, Time=0.57 sec

Best model:  ARIMA(0,0,0)(1,0,1)[7] intercept
Total fit time: 28.692 seconds
```

In [80]:

```
#splitting our data into train and test sets
train=df_1.iloc[:429]
test=df_1.iloc[429:]
```

In [81]:

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

In [82]:

```
model=SARIMAX(train['total'],order=(0,0,0),seasonal_order=(1,0,1,7),enforce_invertibility=False)
results=model.fit()
results.summary()
```

Out[82]:

SARIMAX Results

Dep. Variable:	total	**No. Observations:**	429
Model:	SARIMAX(1, 0, [1], 7)	**Log Likelihood**	-2131.502
Date:	Sat, 05 Sep 2020	**AIC**	4269.003
Time:	00:59:30	**BIC**	4281.187
Sample:	01-01-2016	**HQIC**	4273.815
	- 03-04-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.S.L7	0.9999	9.74e-05	1.03e+04	0.000	1.000	1.000
ma.S.L7	-0.9384	0.024	-38.492	0.000	-0.986	-0.891
sigma2	1115.1152	59.501	18.741	0.000	998.496	1231.734

Ljung-Box (Q):	68.60	Jarque-Bera (JB):	83.49
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.98	Skew:	0.72
Prob(H) (two-sided):	0.88	Kurtosis:	4.61

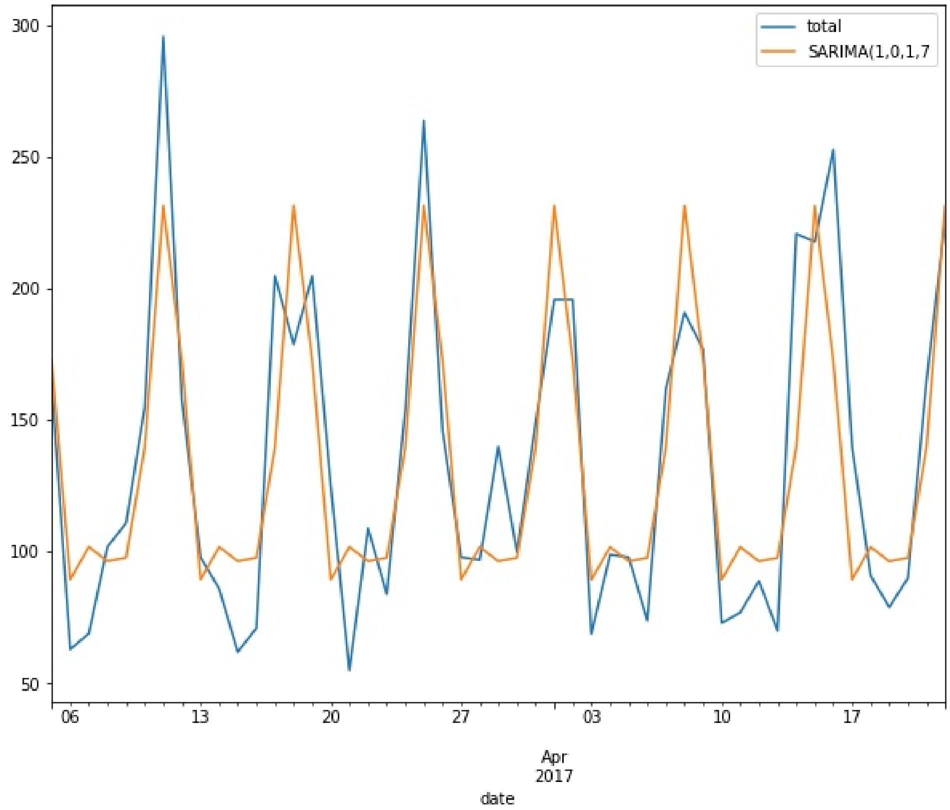
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [83]:

```
start=len(train)
end=len(train)+len(test)-1
predictions=results.predict(start=start,end=end,dynamic=False,typ='levels').rename('SARIMA(1,0,1,7'
)
```

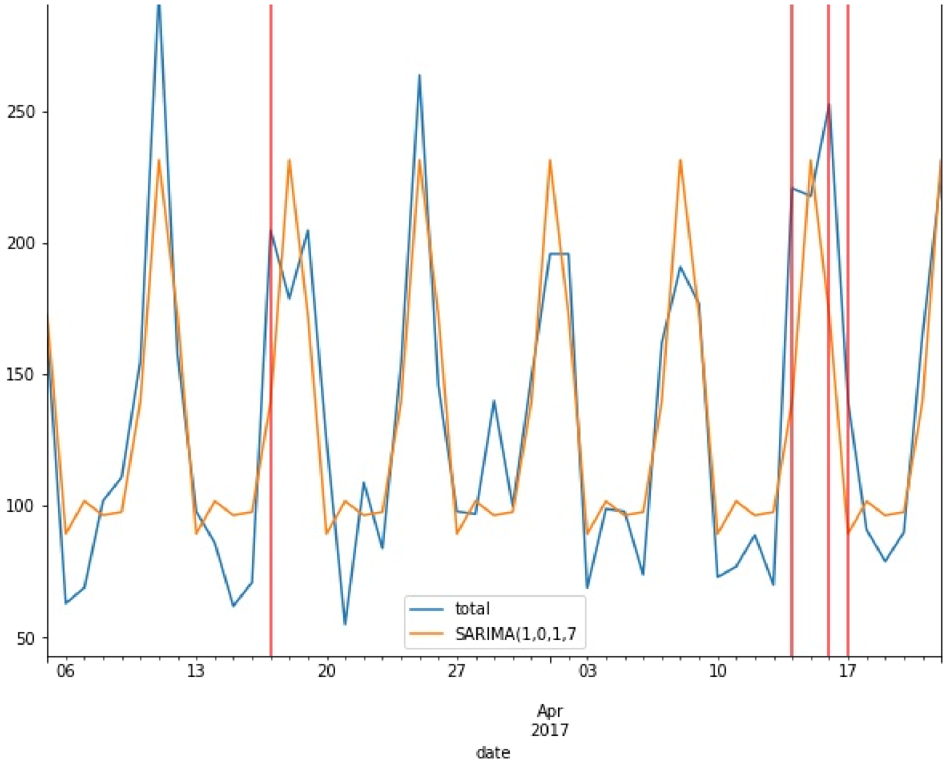
In [84]:

```
test['total'].plot(legend=True,figsize=(10,8))
predictions.plot(legend=True);
```



In [85]:

```
#fitting holidays into our plotted preditions
ax=test['total'].plot(legend=True,figsize=(10,8))
predictions.plot(legend=True)
for x in test.query('holiday==1').index:
    ax.axvline(x=x,color='r',alpha=0.8);
```



In [86]:

```
#evaluating our model without the exogenous variable
from statsmodels.tools.eval_measures import rmse
error=rmse(test['total'],predictions)
```

In [87]:

```
error
```

Out[87]:

```
31.454428561077187
```

In [88]:

```
test['total'].mean()
```

Out[88]:

```
134.6734693877551
```

In [89]:

```
# Including Exogenous variable['total']
au1=auto_arima(df_1['total'],exogenous=df_1[['holiday']],seasonl=True,m=7).summary()
import warnings
warnings.filterwarnings('ignore')
```

In [90]:

```
au1
```

Out[90]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	478
Model:	SARIMAX(0, 0, 1)x(2, 0, [], 7)	**Log Likelihood**	-2348.642
Date:	Sat, 05 Sep 2020	**AIC**	4709.284

Time:	01:00:12	**BIC**	4734.302
Sample:	01-01-2016	**HQIC**	4719.120
	- 04-22-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	11.5526	4.274	2.703	0.007	3.176	19.930
holiday	74.8229	4.545	16.464	0.000	65.916	83.730
ma.L1	0.1781	0.051	3.474	0.001	0.078	0.279
ar.S.L7	0.5061	0.045	11.236	0.000	0.418	0.594
ar.S.L14	0.3843	0.043	8.984	0.000	0.300	0.468
sigma2	1141.9351	80.182	14.242	0.000	984.782	1299.088

Ljung-Box (Q):	85.11	**Jarque-Bera (JB):**	1.77
Prob(Q):	0.00	**Prob(JB):**	0.41
Heteroskedasticity (H):	0.89	**Skew:**	0.12
Prob(H) (two-sided):	0.45	**Kurtosis:**	3.18

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [91]:

```
model=SARIMAX(train['total'],exog=train[['holiday']],order=(0,0,1),seasonal_order=(2,0,0,7),enforce
_invertibility=False)
results=model.fit()
results.summary()
```

Out[91]:

SARIMAX Results

Dep. Variable:	total	**No. Observations:**	429
Model:	SARIMAX(0, 0, 1)x(2, 0, [], 7)	**Log Likelihood**	-2125.435
Date:	Sat, 05 Sep 2020	**AIC**	4260.869
Time:	01:00:32	**BIC**	4281.176
Sample:	01-01-2016	**HQIC**	4268.889
	- 03-04-2017		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
holiday	67.9339	4.305	15.781	0.000	59.497	76.371
ma.L1	0.2028	0.051	4.012	0.000	0.104	0.302
ar.S.L7	0.5230	0.042	12.424	0.000	0.440	0.605
ar.S.L14	0.4501	0.042	10.749	0.000	0.368	0.532
sigma2	1124.5165	74.421	15.110	0.000	978.653	1270.380

Ljung-Box (Q):	102.21	**Jarque-Bera (JB):**	1.50
Prob(Q):	0.00	**Prob(JB):**	0.47
Heteroskedasticity (H):	0.90	**Skew:**	0.12
Prob(H) (two-sided):	0.55	**Kurtosis:**	3.15

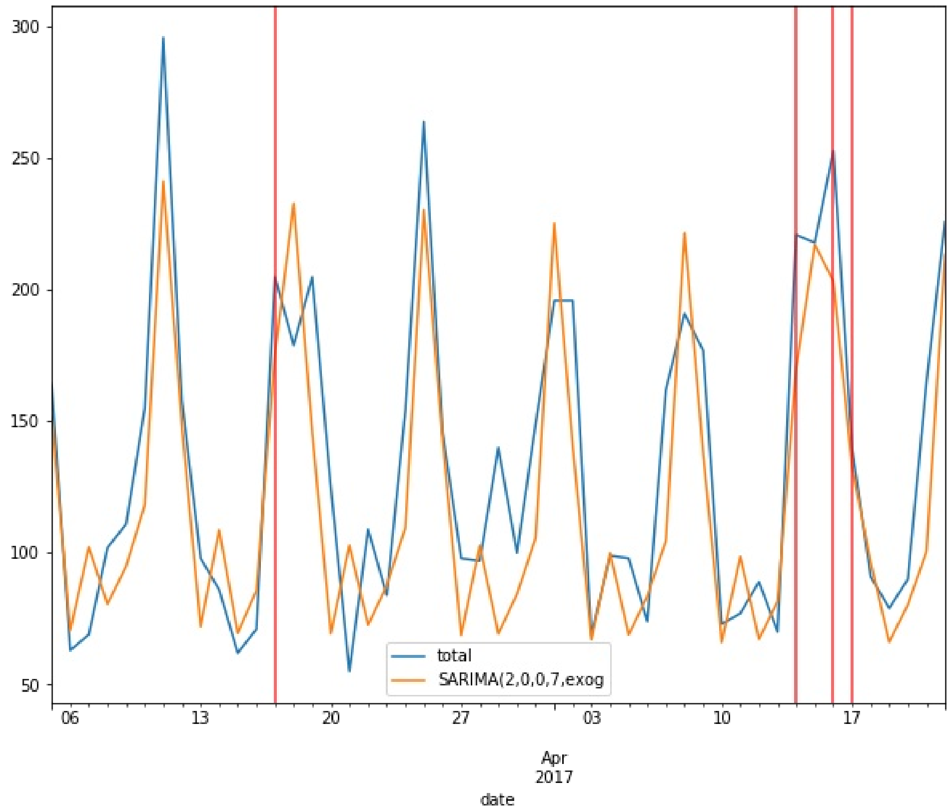
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [92]:

```
# obtaining predictions with exogenous variable
start=len(train)
end=len(train)+len(test)-1
exog_forecast=test[['holiday']]
pred_exogenous=results.predict(start=start,end=end,exog=exog_forecast).rename('SARIMA(2,0,0,7,exog'
)
```

In [93]:

```
#fitting holidays into our plotted preditions
ax=test['total'].plot(legend=True,figsize=(10,8))
pred_exogenous.plot(legend=True)
for x in test.query('holiday==1').index:
    ax.axvline(x=x,color='r',alpha=0.8);
```



In [94]:

```
error_exog=(rmse(test['total'],pred_exogenous))
```

In [95]:

```
error
```

Out[95]:

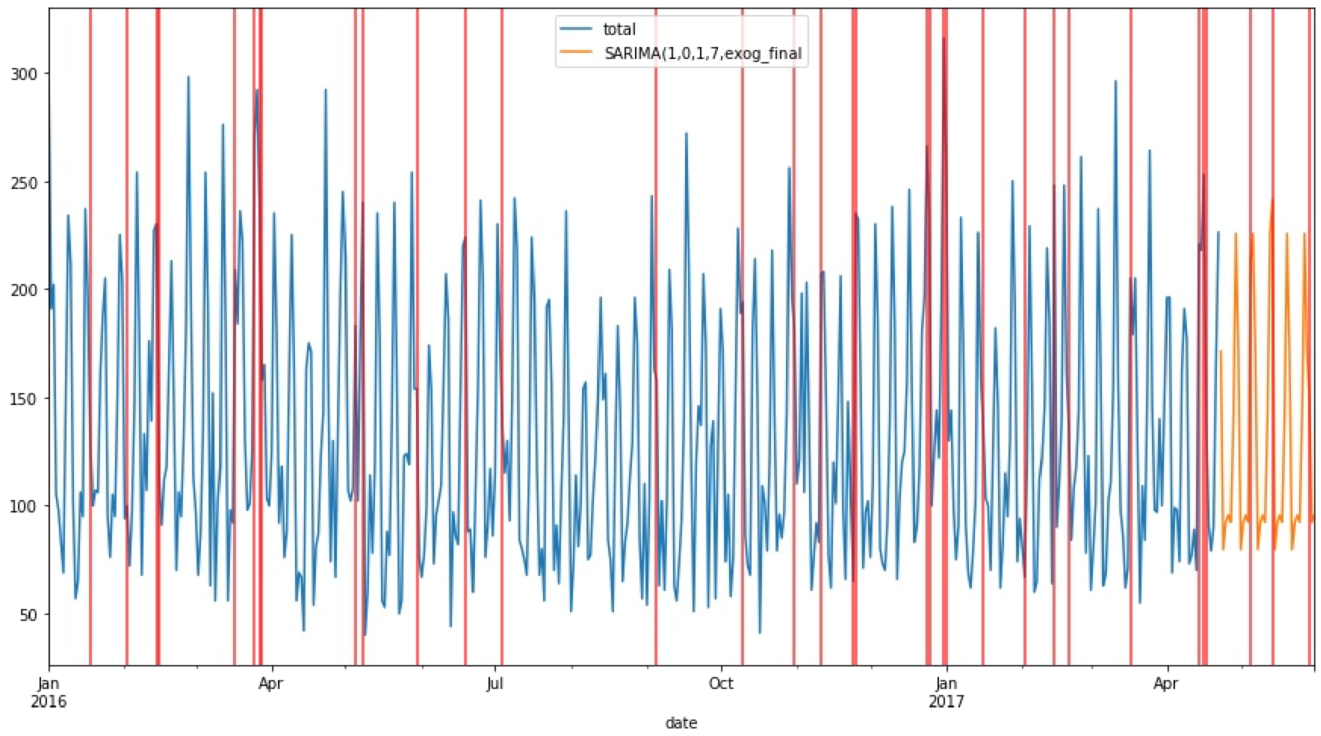
```
31.454428561077187
```

In [98]:

```
#retrain the model on the entire data and forecast for the future
model3=SARIMAX(df_1['total'],exog=df_1[['holiday']],order=(0,0,0),seasonal_order=(1,0,1,7),enforce_
invertibility=False)
result=model3.fit()
exog_forecast=df[478:][['holiday']]
final_forecast=result.predict(len(df_1),len(df_1)+38,exog=exog_forecast).rename('SARIMA(1,0,1,7,exo
g_final')
```

In [99]:

```
#final forcast with our original data
ax=df_1['total'].plot(legend=True,figsize=(15,8))
final_forecast.plot(legend=True)
for x in df.query('holiday==1').index:
    ax.axvline(x=x,color='r',alpha=0.8);
```



In [75]:

```
len(df_1)
```

Out[75]:

```
478
```

In [76]:

```
len(df)
```

Out[76]:

```
517
```

In [ ]:

```

```