Data Set provided by Informa B.V.

Pharmacy Sales

Goal: Forecast Sales

Column 01: Delivery date
This is the date when the product was delivered to the customer.

Column 02: Delivery time
This is the time when the product was delivered to the customer.

Column 03: Pharmacy number
This is the internal system number from the pharmacy. This number can only be used to group data from the same pharmacy.

Column 04: Pharmacy Postcode (2)
This is a part the postcode from the pharmacy. It contains the first 2 numbers from the postcode.

Column 05: Year of birth
Contains the year of birth of the customer

Column 06: Gender
Contains the gender of the customer
(1=male,2=female)

Column 07: CNK
The CNK is the unique product code which is standardized within Belgium.

Column 08: Product name
This is the name of the product in the Dutch language.

Column 09: ATC code
This is the ATC code which is an international standardized. Every medicine has a unique code. This code is built out of several portions.

Column 10: Units
Number of product units in a package

Column 11: Price
The price of the delivery

Column 12: Contribution
The contribution of the customer for this delivery

Resources:
https://www.kite.com/python/answers/how-to-set-column-names-when-importing-a-csv-into-a-pandas-dataframe-in-python (https://www.kite.com/python/answers/how-to-set-column-names-when-importing-a-csv-into-a-pandas-dataframe-in-python)
https://stackoverflow.com/questions/35277075/python-pandas-counting-the-occurrences-of-a-specific-value (https://stackoverflow.com/questions/35277075/python-pandas-counting-the-occurrences-of-a-specific-value)
https://stackoverflow.com/questions/27060098/replacing-few-values-in-a-pandas-dataframe-column-with-another-value (https://stackoverflow.com/questions/27060098/replacing-few-values-in-a-pandas-dataframe-column-with-another-value)
https://thispointer.com/python-pandas-how-to-drop-rows-in-dataframe-by-conditions-on-column-

values/ (https://thispointer.com/python-pandas-how-to-drop-rows-in-dataframe-by-conditions-on-column-values/) https://medium.com/@szabo.bibor/how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e (https://medium.com/@szabo.bibor/how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e)
https://stackoverflow.com/questions/17978092/combine-date-and-time-columns-using-python-pandas (https://stackoverflow.com/questions/17978092/combine-date-and-time-columns-using-python-pandas)
https://www.geeksforgeeks.org/convert-the-column-type-from-string-to-datetime-format-in-pandas-dataframe/ (https://www.geeksforgeeks.org/convert-the-column-type-from-string-to-datetime-format-in-pandas-dataframe/)
https://www.dataquest.io/blog/tutorial-time-series-analysis-with-pandas/ (https://www.dataquest.io/blog/tutorial-time-series-analysis-with-pandas/)
https://stackoverflow.com/questions/60214194/error-in-reading-stock-data-datetimeproperties-object-has-no-attribute-week (https://stackoverflow.com/questions/60214194/error-in-reading-stock-data-datetimeproperties-object-has-no-attribute-week)
https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b (https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b)

In [1]:
```
import pandas as pd
import numpy as np

# Ignore competability warnings
import warnings
warnings.filterwarnings('ignore')

# Option to show all the DataFrame columns
pd.options.display.max_columns = None
```

In [2]:
```
header_list = ["Delivery Date", "Delivery Time", "Pharmacy Number", "Pharmac
               "YOB", "Gender", "CNK Code", "Product Name", "ATC Code", "Uni
df_pharmacy = pd.read_csv("data_2020/data_2020.csv", names=header_list)
```

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

Out[3]:
```
(5072146, 12)
```

In [4]:
```
df_pharmacy.head(10)
```

Out[4]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	C
0	01/01/2020	00:00	9105972	10	1925	2	5520465	HONORARIUM PER WEEK PER RUSTOORDBEWONER ROB-RVT	
1	01/01/2020	00:00	9105972	10	1923	2	5520465	HONORARIUM PER WEEK PER RUSTOORDBEWONER ROB-RVT	
2	01/01/2020	00:00	9105972	10	1924	2	736165	BURINEX COMP 1 X 5 MG	C03C
3	01/01/2020	00:00	9105972	10	1921	2	750695	XARELTO COMP 1 X 15 MG	B01A
4	01/01/2020	00:00	9105972	10	1924	2	750695	XARELTO COMP 1 X 15 MG	B01A
5	01/01/2020	00:00	9105972	10	1921	2	7706310	ESCITALOPRAM TEVA COMP 1 X 10 MG	N06A
6	01/01/2020	00:00	9105972	10	1921	2	7706310	ESCITALOPRAM TEVA COMP 1 X 10 MG	N06A
7	01/01/2020	00:00	9105972	10	1924	2	743732	L THYROXINE CHRISTIAENS COMP 1 X 50 MCG	H03A
8	01/01/2020	00:00	9105972	10	1923	2	743732	L THYROXINE CHRISTIAENS COMP 1 X 50 MCG	H03A
9	01/01/2020	00:00	9105972	10	1923	2	789537	ASAFLOW COMP 1 X 80 MG	B01A

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5072146 entries, 0 to 5072145
Data columns (total 12 columns):
 #   Column             Dtype  
---  ------             -----  
 0   Delivery Date      object 
 1   Delivery Time      object 
 2   Pharmacy Number    int64  
 3   Pharmacy Postcode  int64  
 4   YOB                int64  
 5   Gender             int64  
 6   CNK Code           int64  
 7   Product Name       object 
 8   ATC Code           object 
 9   Units              int64  
 10  Price              float64
 11  Contribution       float64
dtypes: float64(2), int64(6), object(4)
memory usage: 464.4+ MB
```

Data Wrangling

In [6]:
```
# Replace first empty placeholders - not all spaces are NaNs
df_pharmacy.replace(' ', '', inplace=True)
```

In [7]:
```
# Check for missing values in each column
df_pharmacy.isnull().sum()
```

Out[7]:
```
Delivery Date             0
Delivery Time        972059
Pharmacy Number           0
Pharmacy Postcode         0
YOB                       0
Gender                    0
CNK Code                  0
Product Name              0
ATC Code                  0
Units                     0
Price                     0
Contribution              0
dtype: int64
```

In [8]:
```
missing_value_formats = ["n.a.", "?", "NA", "n/a", "na", "--", "", 0, 0.00]

df_pharmacy.replace(
    to_replace=missing_value_formats,
    value=np.nan,
    inplace=True
)
```

In [9]:
```
# Check for missing values in each column
df_pharmacy.isnull().sum()
```

Out[9]:
```
Delivery Date             0
Delivery Time        972059
Pharmacy Number           0
Pharmacy Postcode     37962
YOB                    1580
Gender                12150
CNK Code                  0
Product Name              0
ATC Code             168154
Units                599669
Price                187438
Contribution         848781
dtype: int64
```

Drop additional Gender value == 3

In [10]:
```
df_pharmacy.Gender.unique()
```

Out[10]:
```
array([ 2.,  1., nan,  3.])
```

In [11]:
```
indexes = df_pharmacy[df_pharmacy['Gender'] == 3].index
df_pharmacy.drop(indexes , inplace=True)
```

In [12]:
```
df_pharmacy.Gender.unique()
```

Out[12]:
```
array([ 2.,  1., nan])
```

In [ ]:
```
# Other checks - To be removed

# Orders with 0 units
#df_pharmacy[df_pharmacy.Units == 0].shape[0]

# Orders with 0 units and no contribution
#df_pharmacy[(df_pharmacy.Units == 0) & (df_pharmacy.Contribution == 0.00)]

#data = df_pharmacy[(df_pharmacy.Units != 0) & (df_pharmacy.Contribution !=
#data.shape

#check = df_pharmacy.shape[0] - data.shape[0]
#check
```

Unique values per column, types and nans

In [13]:
```
df_pharmacy[150:200]
```

Out[13]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Proc
150	01/01/2020	NaN	7056069	42.0	NaN	NaN	5520937	BESCHIKBAARHEIDSHON
151	01/01/2020	NaN	7668522	28.0	NaN	NaN	5520937	BESCHIKBAARHEIDSHON
152	01/01/2020	NaN	7807692	12.0	NaN	NaN	5520937	BESCHIKBAARHEIDSHON
153	01/01/2020	NaN	8112792	60.0	NaN	NaN	5520937	BESCHIKBAARHEIDSHON
154	01/01/2020	03:52	9099141	20.0	2003.0	1.0	1715127	AMOXICLAV SANDOZ 8 MG
155	01/01/2020	03:52	9099141	20.0	2005.0	1.0	1715127	AMOXICLAV SANDOZ 8 MG
156	01/01/2020	03:52	9099141	20.0	2002.0	1.0	5520523	WACHTHON
157	01/01/2020	03:52	9099141	20.0	2003.0	1.0	5520523	WACHTHON
158	01/01/2020	06:58	9099141	20.0	2009.0	1.0	2202372	AMOXICILLINE SANDO TAE
159	01/01/2020	06:58	9099141	20.0	2009.0	1.0	5520523	WACHTHON
160	01/01/2020	06:58	9099141	20.0	2007.0	1.0	5520523	WACHTHON
161	01/01/2020	08:51	9099141	20.0	1977.0	2.0	5520523	WACHTHON
162	01/01/2020	08:51	9099141	20.0	1976.0	2.0	5520523	WACHTHON
163	01/01/2020	08:51	9099141	20.0	1976.0	2.0	86470	EXACYL AMP PER OS 10
164	01/01/2020	08:51	9099141	20.0	1977.0	2.0	86470	EXACYL AMP PER OS 10
165	01/01/2020	08:52	9051816	10.0	1989.0	2.0	104596	BRUFEN 400 MG FILM 100
166	01/01/2020	08:52	9051816	10.0	1990.0	2.0	1458736	AUGMENTIN 875
167	01/01/2020	08:52	9051816	10.0	1989.0	2.0	1458736	AUGMENTIN 875
168	01/01/2020	08:52	9051816	10.0	1990.0	2.0	5520523	WACHTHON
169	01/01/2020	08:52	9051816	10.0	1990.0	2.0	5520523	WACHTHON
170	01/01/2020	08:18	7068483	90.0	1996.0	1.0	2744761	AMOXICILLINE EC FILMOMH TABL 24 X
171	01/01/2020	09:32	7815132	30.0	1986.0	2.0	3172467	BUFOMIX 160MC EASYHALER INHAL PDI
172	01/01/2020	09:32	7815132	30.0	1986.0	2.0	5520523	WACHTHON
173	01/01/2020	09:05	7684719	20.0	1970.0	2.0	321596	FRAXIPARINE SER 10 X 0
174	01/01/2020	09:05	7684719	20.0	1971.0	2.0	5520523	WACHTHON
175	01/01/2020	09:09	7684719	20.0	1940.0	1.0	1199058	FELDENE LYOTABS
176	01/01/2020	09:09	7684719	20.0	1938.0	1.0	5520523	WACHTHON
177	01/01/2020	09:14	8112792	60.0	1945.0	2.0	2214906	TRADONAL ODIS ORODIS

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Proc
178	01/01/2020	09:14	8112792	60.0	1945.0	2.0	278192	CLEXANE SPUIT INJ 10 X
179	01/01/2020	09:14	8112792	60.0	1948.0	2.0	278192	CLEXANE SPUIT INJ 10 X
180	01/01/2020	09:14	8112792	60.0	1946.0	2.0	5520523	WACHTHON
181	01/01/2020	09:14	8112792	60.0	1947.0	2.0	5520523	WACHTHON
182	01/01/2020	10:26	9090330	10.0	1946.0	2.0	2697670	AMOXICILLINE SANDO TAE
183	01/01/2020	10:26	9090330	10.0	1947.0	2.0	5520523	WACHTHON
184	01/01/2020	10:26	9090330	10.0	1946.0	2.0	5520523	WACHTHON
185	01/01/2020	10:26	7807692	12.0	1984.0	2.0	1132885	IBUPROFEN EG 400 MG TABL 30
186	01/01/2020	10:26	7807692	12.0	1984.0	2.0	5520523	WACHTHON
187	01/01/2020	10:34	7668522	28.0	1948.0	1.0	2697670	AMOXICILLINE SANDO TAE
188	01/01/2020	10:34	7668522	28.0	1950.0	1.0	2697670	AMOXICILLINE SANDO TAE
189	01/01/2020	10:34	7668522	28.0	1950.0	1.0	5520523	WACHTHON
190	01/01/2020	10:35	9090330	10.0	1942.0	1.0	117572	LASIX COMP 5
191	01/01/2020	10:35	9090330	10.0	1942.0	1.0	117572	LASIX COMP 5
192	01/01/2020	10:35	9090330	10.0	1939.0	1.0	1677863	DUOVENT HFA AERO D
193	01/01/2020	10:35	9090330	10.0	1941.0	1.0	1677863	DUOVENT HFA AERO D
194	01/01/2020	10:35	9090330	10.0	1943.0	1.0	2650703	TRITACE COMP 56 X
195	01/01/2020	10:35	9090330	10.0	1941.0	1.0	5520523	WACHTHON
196	01/01/2020	10:35	9090330	10.0	1943.0	1.0	5520523	WACHTHON
197	01/01/2020	10:36	7338390	10.0	1970.0	2.0	2308443	TRANSTEC 35,0MCG/U
198	01/01/2020	10:36	7338390	10.0	1969.0	2.0	5520523	WACHTHON
199	01/01/2020	10:37	7056069	42.0	1992.0	2.0	5520523	WACHTHON

In [14]:
```
df_pharmacy["Pharmacy Postcode"].unique()
```

Out[14]:
```
array([10., 41., 84., 30., 61., 11., 15., 28., 42., 12., 60., 20., 90.,
       75., 68., 13., 89., 53., 87., 92., 94., 86., 21., 40., 44., 17.,
       39., 14., 55., 50., 26., 25., 93., 48., 70., 22., 99., 29., 88.,
       35., 16., 71., 56., 23., 45., 36., 80., 24., 38., 91., nan, 66.,
       51., 65., 83., 81., 96., 43., 97., 85.])
```

In [15]:
```
df_pharmacy.dtypes
```

Out[15]:
```
Delivery Date          object
Delivery Time          object
Pharmacy Number         int64
Pharmacy Postcode     float64
YOB                   float64
Gender                float64
CNK Code                int64
Product Name           object
ATC Code               object
Units                 float64
Price                 float64
Contribution          float64
dtype: object
```

In [16]:
```
df = df_pharmacy.dropna()
df = df.reset_index(drop=True)
df.shape
```

Out[16]:
```
(3309068, 12)
```

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
Delivery Date         0
Delivery Time         0
Pharmacy Number       0
Pharmacy Postcode     0
YOB                   0
Gender                0
CNK Code              0
Product Name          0
ATC Code              0
Units                 0
Price                 0
Contribution          0
dtype: int64
```

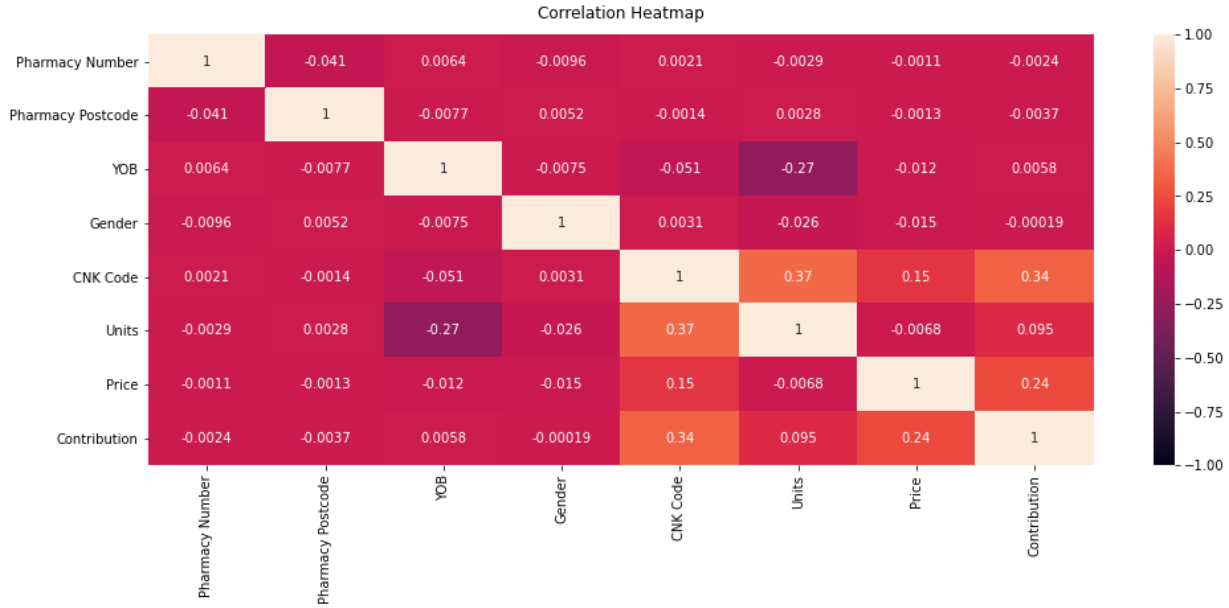
In [18]:
```
df = df.astype({'Pharmacy Postcode': 'int', 'YOB': 'int', 'Gender': 'int',
df.dtypes
```

Out[18]:
```
Delivery Date          object
Delivery Time          object
Pharmacy Number         int64
Pharmacy Postcode       int32
YOB                     int32
Gender                  int32
CNK Code                int64
Product Name           object
ATC Code               object
Units                   int32
Price                 float64
Contribution          float64
dtype: object
```

EDA

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [65]:
```
# Increase the size of the heatmap.
plt.figure(figsize=(16, 6))
# Store heatmap object in a variable to easily access it when you want to i
# Set the range of values to be displayed on the colormap from -1 to 1, and
# to display the correlation values on the heatmap.
heatmap = sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True)
# Give a title to the heatmap. Pad defines the distance of the title from t
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



In [66]:
```
# Not all relationships are symetrical, please read the following article
# https://towardsdatascience.com/rip-correlation-introducing-the-predictive-
# The Predictive Power Score PPS may give us a better understanding
# of the categorical columns and their relationships

# If you dont have the lib - pip install ppscore
import ppscore as pps

# Create a matrix with all the results
pps_matrix = pps.matrix(df)
pps_matrix
```

Out[66]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	
Delivery Date	1.000000	0.000531	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.0(
Delivery Time	0.000000	1.000000	0.232195	0.038308	0.000198	0.0	0.000000	0.000000	0.0(
Pharmacy Number	0.000000	0.000000	1.000000	0.009950	0.000000	0.0	0.000000	0.000000	0.0(
Pharmacy Postcode	0.000000	0.000000	0.998296	1.000000	0.000000	0.0	0.000000	0.000000	0.0(
YOB	0.000000	0.000000	0.000000	0.000000	1.000000	0.0	0.070448	0.100616	0.12
Gender	0.000000	0.000000	0.000000	0.000000	0.000000	1.0	0.090678	0.092824	0.12
CNK Code	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	1.000000	0.857167	0.54
Product Name	0.000000	0.000476	0.001504	0.000000	0.003673	0.0	0.860582	1.000000	0.3(
ATC Code	0.004144	0.000000	0.002247	0.000000	0.007923	0.0	0.896220	0.880725	1.0(
Units	0.000000	0.000000	0.000000	0.000000	0.043735	0.0	0.866590	0.848453	0.64
Price	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.594811	0.653348	0.59
Contribution	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.672497	0.640323	0.52

Time Series Data

In [20]:
```
df[10333:10339]
```

Out[20]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	ATC Cod
10333	01/02/2020	14:43	7084071	20	1937	2	2557213	PANTOMED 20 MG TABL 100	A02BC0
10334	01/02/2020	14:43	7084071	20	1939	2	3181757	L THYROXINE CHRISTIAENS COMP 112X0,075MG	H03AA0
10335	01/02/2020	14:43	4065450	20	1954	1	3154549	ALGOTRA COMP OMHULDE 30 X 325MG/37,5MG	N02AJ1
10336	01/02/2020	14:43	4065450	20	1950	1	3154549	ALGOTRA COMP OMHULDE 30 X 325MG/37,5MG	N02AJ1
10337	01/02/2020	14:43	7668522	28	1955	1	1677798	TRAMADOL EG TABL 60 X 50 MG	N02AX0
10338	01/02/2020	14:43	7668522	28	1955	1	1677798	TRAMADOL EG TABL 60 X 50 MG	N02AX0

In [21]:
```
df['delivery_date']= pd.to_datetime(df['Delivery Date'])
```

In [22]:
```
df.dtypes
```

Out[22]:
```
Delivery Date                  object
Delivery Time                  object
Pharmacy Number                 int64
Pharmacy Postcode               int32
YOB                             int32
Gender                          int32
CNK Code                        int64
Product Name                   object
ATC Code                       object
Units                           int32
Price                        float64
Contribution                 float64
delivery_date         datetime64[ns]
dtype: object
```

In [23]:
```
df[10333:10339]
```

Out[23]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	ATC Cod
10333	01/02/2020	14:43	7084071	20	1937	2	2557213	PANTOMED 20 MG TABL 100	A02BC0
10334	01/02/2020	14:43	7084071	20	1939	2	3181757	L THYROXINE CHRISTIAENS COMP 112X0,075MG	H03AA0
10335	01/02/2020	14:43	4065450	20	1954	1	3154549	ALGOTRA COMP OMHULDE 30 X 325MG/37,5MG	N02AJ1
10336	01/02/2020	14:43	4065450	20	1950	1	3154549	ALGOTRA COMP OMHULDE 30 X 325MG/37,5MG	N02AJ1
10337	01/02/2020	14:43	7668522	28	1955	1	1677798	TRAMADOL EG TABL 60 X 50 MG	N02AX0
10338	01/02/2020	14:43	7668522	28	1955	1	1677798	TRAMADOL EG TABL 60 X 50 MG	N02AX0

In [24]:
```
df['delivery_date_time'] = pd.to_datetime(df['Delivery Date'] + ' ' + df['De
```

In [25]:
```
df[10333:10339]
```

Out[25]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	ATC Cod
10333	01/02/2020	14:43	7084071	20	1937	2	2557213	PANTOMED 20 MG TABL 100	A02BC0
10334	01/02/2020	14:43	7084071	20	1939	2	3181757	L THYROXINE CHRISTIAENS COMP 112X0,075MG	H03AA0
10335	01/02/2020	14:43	4065450	20	1954	1	3154549	ALGOTRA COMP OMHULDE 30 X 325MG/37,5MG	N02AJ1
10336	01/02/2020	14:43	4065450	20	1950	1	3154549	ALGOTRA COMP OMHULDE 30 X 325MG/37,5MG	N02AJ1
10337	01/02/2020	14:43	7668522	28	1955	1	1677798	TRAMADOL EG TABL 60 X 50 MG	N02AX0
10338	01/02/2020	14:43	7668522	28	1955	1	1677798	TRAMADOL EG TABL 60 X 50 MG	N02AX0

In [26]:
```
df_time = df.set_index('delivery_date_time')
df_time.head(5)
```

Out[26]:

delivery_date_time	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Produ
2020-01-01	01/01/2020	00:00	9105972	10	1945	2	2218279	DUROG E 25MCG/HEU
2020-01-01	01/01/2020	00:00	9123051	61	1955	2	2622264	AMOC 875MC FILMOMH
2020-01-01	01/01/2020	00:00	9123051	61	1957	2	2622264	AMOC 875MC FILMOMH
2020-01-01	01/01/2020	00:00	9123051	61	1956	2	867556	BRUFEN FO MG FILMOI 30 X
2020-01-01	01/01/2020	00:00	9123051	61	1959	2	867556	BRUFEN FO MG FILMOI 30 X

In [27]:
```
df_time.index
```

Out[27]:
```
DatetimeIndex(['2020-01-01 00:00:00', '2020-01-01 00:00:00',
               '2020-01-01 00:00:00', '2020-01-01 00:00:00',
               '2020-01-01 00:00:00', '2020-01-01 00:00:00',
               '2020-01-01 00:00:00', '2020-01-01 00:00:00',
               ...
               '2020-07-31 23:19:00', '2020-07-31 23:19:00',
               '2020-07-31 23:19:00', '2020-07-31 23:19:00',
               '2020-07-31 23:19:00', '2020-07-31 23:19:00',
               '2020-07-31 23:19:00', '2020-07-31 23:19:00'],
              dtype='datetime64[ns]', name='delivery_date_time', length=3
309068, freq=None)
```

In [28]:
```
# Add columns with year, month, and weekday name
df_time['Year'] = df_time.index.year
df_time['Month'] = df_time.index.month
df_time['Weekday Name'] = df_time['delivery_date'].dt.day_name()
# Display a random sampling of 5 rows
df_time.sample(5, random_state=0)
```

Out[28]:

delivery_date_time	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product N
2020-05-25 10:32:00	05/25/2020	10:32	7726050	28	1942	1	16832	ALLOPURI EG COMP 30(
2020-04-08 17:56:00	04/08/2020	17:56	7056069	42	1974	2	119172	MAXI COLLYRE (
2020-01-17 00:00:00	01/17/2020	00:00	7666512	29	1958	1	117572	LASIX COM X 4(
2020-02-26 13:57:00	02/26/2020	13:57	7840239	30	2019	2	2322436	ROTAT TUBE 1 DO
2020-02-10 00:00:00	02/10/2020	00:00	7993260	35	1936	1	1092857	LEDERTREX C(30X2,

In [82]:
```
# Now you can slice data based on time periods
df_time.loc['2020-01-05':'2020-02-05']
```

Out[82]:

delivery_date_time	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Pro
2020-02-01 09:18:00	02/01/2020	09:18	7612332	80	1956	2	1132885	IBUPROF MG FILMOI
2020-02-01 10:56:00	02/01/2020	10:56	7612332	80	1961	1	1184027	ZESTR
2020-02-01 10:56:00	02/01/2020	10:56	7612332	80	1959	1	2695864	ZANIDIP CC
2020-02-01 11:04:00	02/01/2020	11:04	7612332	80	1946	1	2555159	PANTOPF MAAGSAPR
2020-01-05 00:00:00	01/05/2020	00:00	8101971	10	1936	2	113399	FURAD/ CAPS 5
...	...	...	...	...	...	...	...	
2020-01-07 16:53:00	01/07/2020	16:53	7695600	43	1935	2	3049905	INUVAIF NEXTHAL
2020-01-07 17:23:00	01/07/2020	17:23	7612332	80	1980	2	40428	ETUMIN COMP
2020-01-07 17:10:00	01/07/2020	17:10	7695600	43	1947	2	3000718	MONOPRC COLLYRE
2020-01-07 17:13:00	01/07/2020	17:13	7695600	43	1968	1	2116945	CITAL COMP
2020-01-07 17:13:00	01/07/2020	17:13	7695600	43	1967	1	2116945	CITAL COMP

677063 rows × 16 columns

In [83]:
```
# Or slice the entire month rows
df_time.loc['2020-03']
```

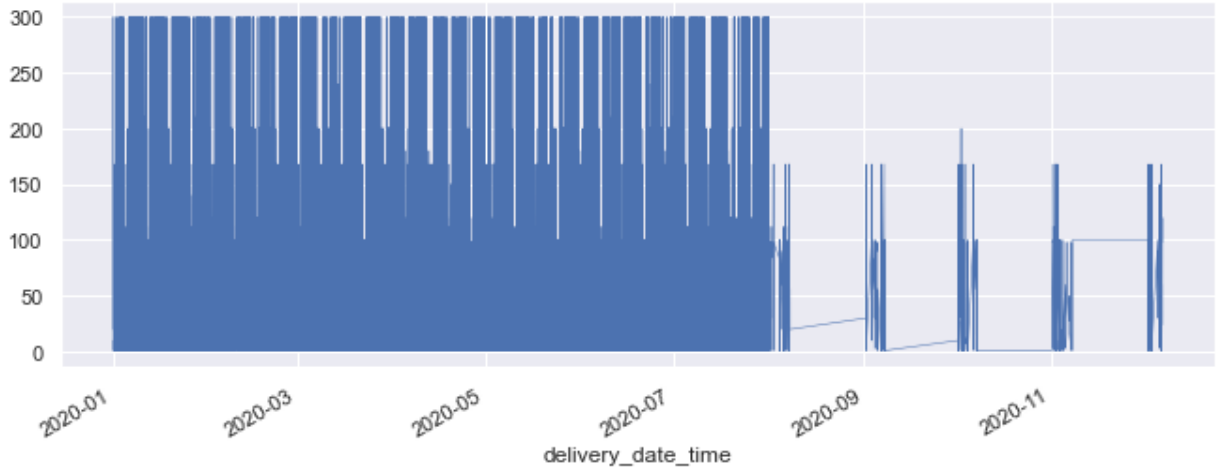
Out[83]:

delivery_date_time	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product I
2020-03-01 09:52:00	03/01/2020	09:52	7612332	80	1995	2	2666352	MONTELU SANDOZ 1 FILMOMH 28 X 1
2020-03-01 09:54:00	03/01/2020	09:54	7612332	80	1941	2	2990760	ATORSTAT 20 MG FILM TAB
2020-03-01 09:59:00	03/01/2020	09:59	7612332	80	1966	1	2351740	AZITHROMY 500 MG SAN TABL OMH 6
2020-03-01 09:06:00	03/01/2020	09:06	7612332	80	1935	1	1715127	AMOXI SANDO MG/12 CON
2020-03-01 09:06:00	03/01/2020	09:06	7612332	80	1937	1	1715127	AMOXI SANDO MG/12 CON
...	...	...	...	...	...	...	...	
2020-03-07 18:47:00	03/07/2020	18:47	7695600	43	1950	2	891416	UTROGE CAPS ORAL 10
2020-03-07 18:47:00	03/07/2020	18:47	7695600	43	1949	2	891416	UTROGE CAPS ORAL 10
2020-03-07 18:47:00	03/07/2020	18:47	7695600	43	1953	2	891416	UTROGE CAPS ORAL 10
2020-03-07 18:47:00	03/07/2020	18:47	7695600	43	1951	2	891416	UTROGE CAPS ORAL 10
2020-03-07 18:47:00	03/07/2020	18:47	7695600	43	1950	2	891416	UTROGE CAPS ORAL 10

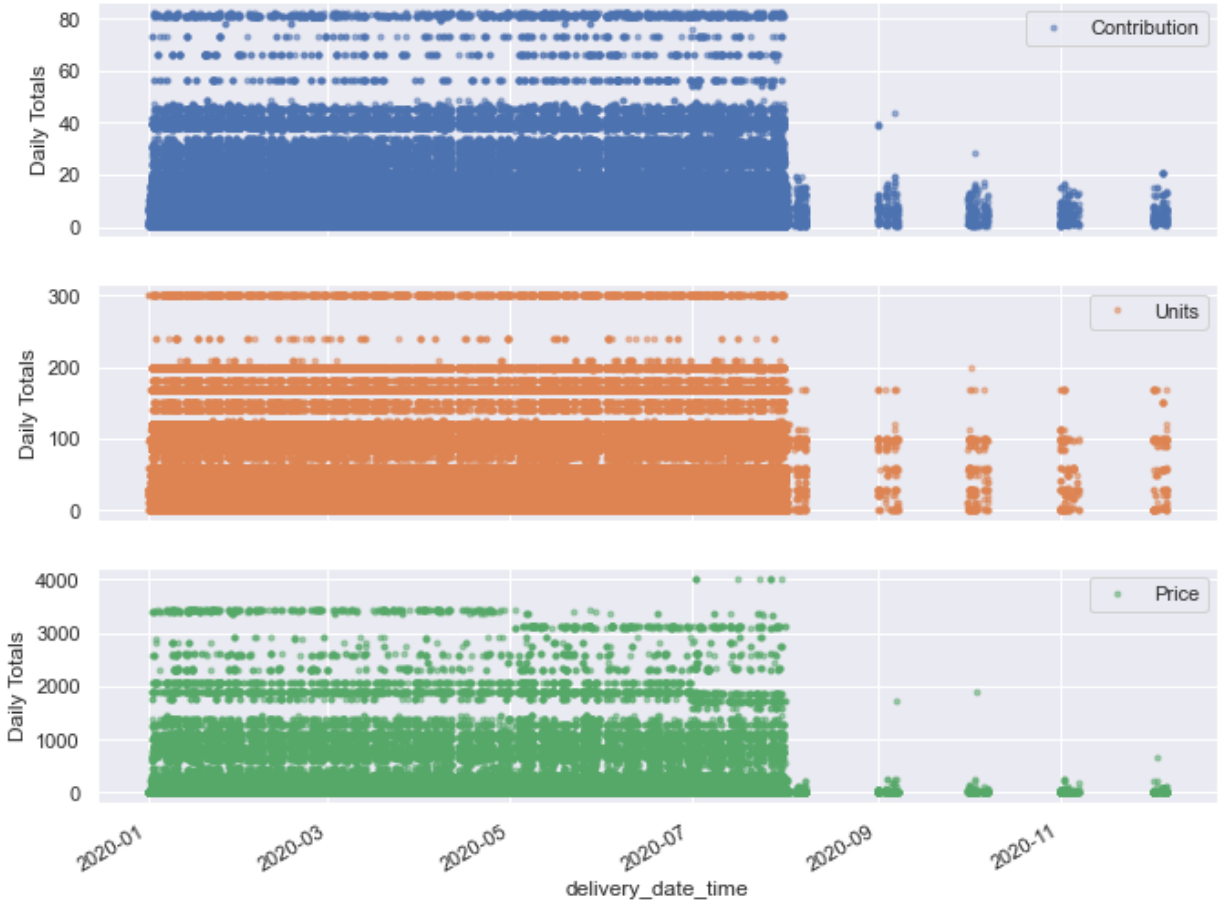
435911 rows × 16 columns

In [84]:
```
import matplotlib.pyplot as plt
# Display figures inline in Jupyter notebook
import seaborn as sns
# Use seaborn style defaults and set the default figure size
sns.set(rc={'figure.figsize':(11, 4)})

df_time['Contribution'].plot(linewidth=0.5);
```

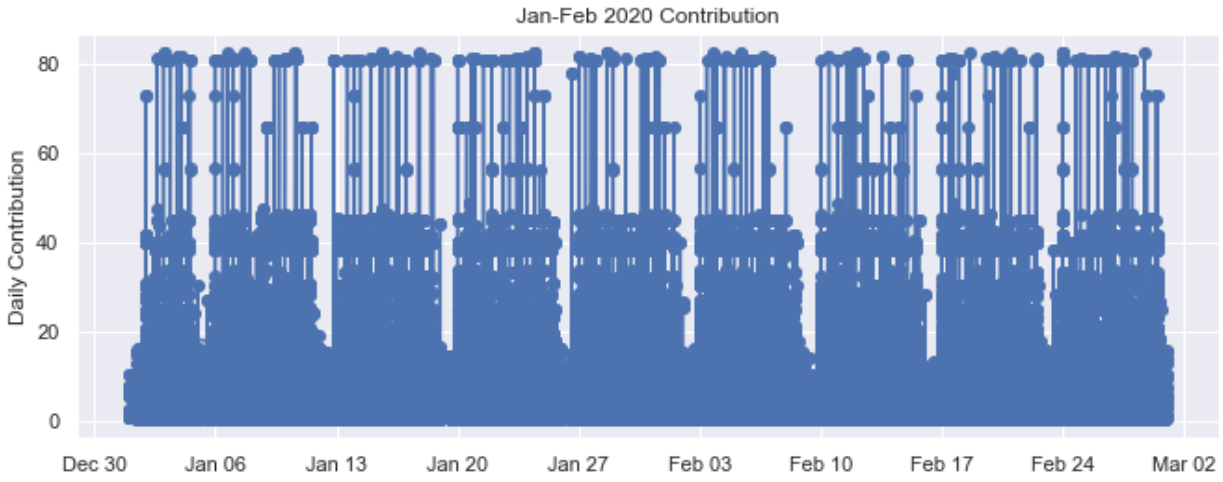
In [87]:
```
df_time['Units'].plot(linewidth=0.5);
```



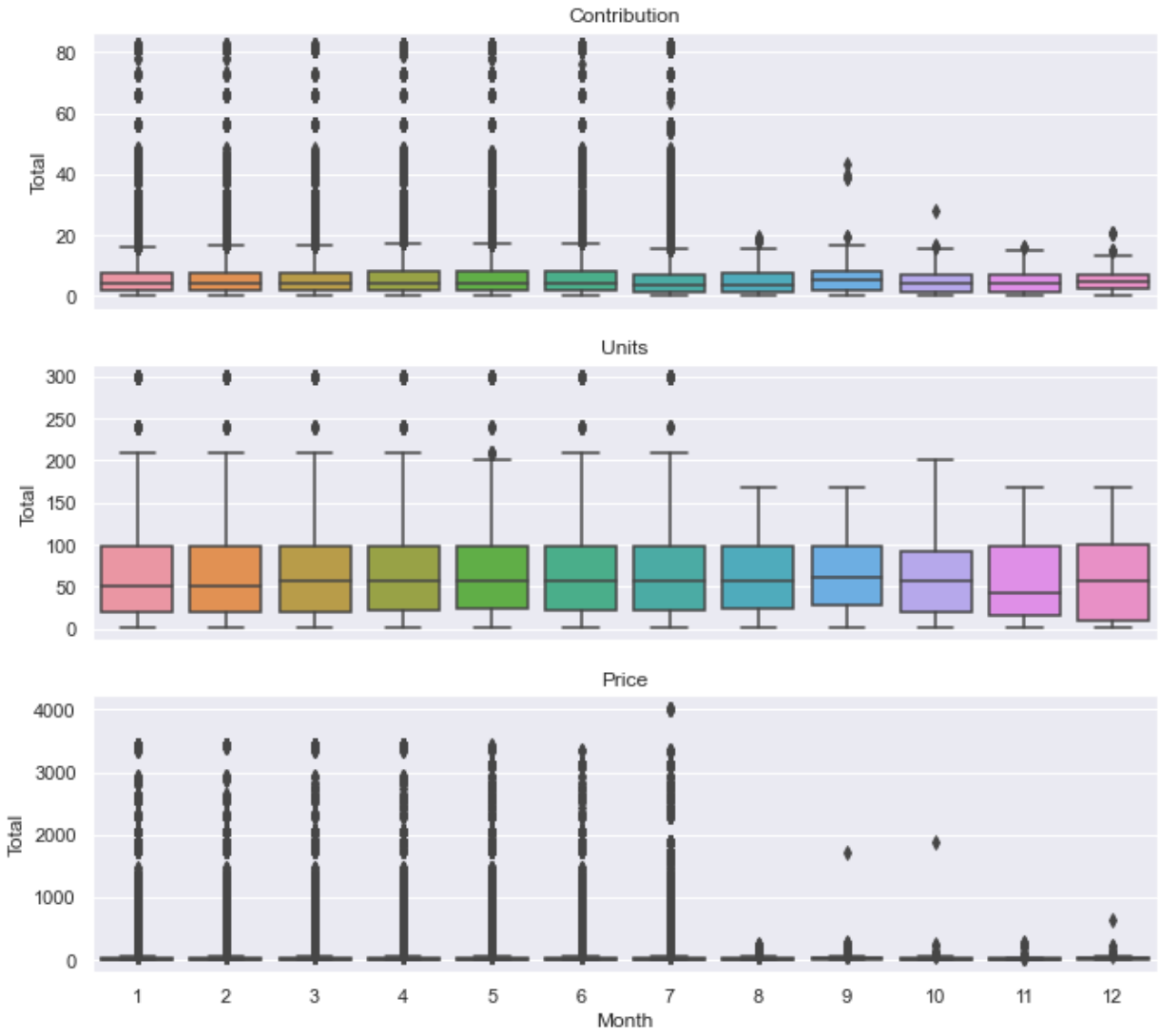
In [88]:
```
cols_plot = ['Contribution', 'Units', 'Price']
axes = df_time[cols_plot].plot(marker='.', alpha=0.5, linestyle='None', figs
for ax in axes:
    ax.set_ylabel('Daily Totals')
```



In [90]:
```
import matplotlib.dates as mdates

fig, ax = plt.subplots()
ax.plot(df_time.loc['2020-01':'2020-02', 'Contribution'], marker='o', linest
ax.set_ylabel('Daily Contribution')
ax.set_title('Jan-Feb 2020 Contribution')

# Set x-axis major ticks to weekly interval, on Mondays
ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MONDAY))
# Format x-tick labels as 3-letter month name and day number
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %d'));
```



In [92]:
```
fig, axes = plt.subplots(3, 1, figsize=(11, 10), sharex=True)
for name, ax in zip(['Contribution', 'Units', 'Price'], axes):
    sns.boxplot(data=df_time, x='Month', y=name, ax=ax)
    ax.set_ylabel('Total')
    ax.set_title(name)
    # Remove the automatic x-axis label from all but the bottom subplot
    if ax != axes[-1]:
        ax.set_xlabel('')
```



In [36]:
```
# Here more visualizations from https://www.dataquest.io/blog/tutorial-time-
# Please if you want to re-create the visualizations in the project folder t
# https://www.jmp.com/en_gb/home.html
# Software for data analisis from SAS capable to deal with big amounts of da
```

In [ ]:
```

```

In [ ]:
```

```

In [31]:
```
df_time.head()
```

Out[31]:

delivery_date_time	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Produ
2020-01-01	01/01/2020	00:00	9105972	10	1945	2	2218279	DUROG E 25MCG/HEU
2020-01-01	01/01/2020	00:00	9123051	61	1955	2	2622264	AMOC 875MC FILMOMH
2020-01-01	01/01/2020	00:00	9123051	61	1957	2	2622264	AMOC 875MC FILMOMH
2020-01-01	01/01/2020	00:00	9123051	61	1956	2	867556	BRUFEN FO MG FILMOI 30 X
2020-01-01	01/01/2020	00:00	9123051	61	1959	2	867556	BRUFEN FO MG FILMOI 30 X

In [29]:
```
df_time.shape
```

Out[29]:
```
(3309068, 16)
```

In [38]:
```
group_by_date = df_time.groupby('delivery_date')['Contribution'].sum()
print(group_by_date)
```

```
delivery_date
2020-01-01     1547.21
2020-01-02    73568.54
2020-01-03    93811.55
2020-01-04    40186.05
2020-01-05     2119.99
                 ...
2020-11-07       41.63
2020-12-02      177.73
2020-12-03      105.68
2020-12-05      329.84
2020-12-06      197.68
Name: Contribution, Length: 242, dtype: float64
```

In [39]:
```
group_by_price = df_time.groupby('delivery_date')['Price'].sum()
print(group_by_price)
```

```
delivery_date
2020-01-01      5381.65
2020-01-02    376623.48
2020-01-03    440875.85
2020-01-04    206925.18
2020-01-05      6499.61
                 ...
2020-11-07       352.88
2020-12-02       881.09
2020-12-03      1305.32
2020-12-05      1612.12
2020-12-06       770.85
Name: Price, Length: 242, dtype: float64
```

In [40]:
```
type(group_by_price)
```

Out[40]:
```
pandas.core.series.Series
```

In [41]:
```
df_test = group_by_price.to_frame().reset_index()
```

In [44]:
```
df_test.shape
```

Out[44]:
```
(242, 2)
```

In [45]:
```
df_test.head()
```

Out[45]:

	delivery_date	Price
0	2020-01-01	5381.65
1	2020-01-02	376623.48
2	2020-01-03	440875.85
3	2020-01-04	206925.18
4	2020-01-05	6499.61

In [30]:
```
# Contribution per pharmacy
group_class = df_time.groupby('Pharmacy Number')['Contribution'].sum()
print(group_class)
```

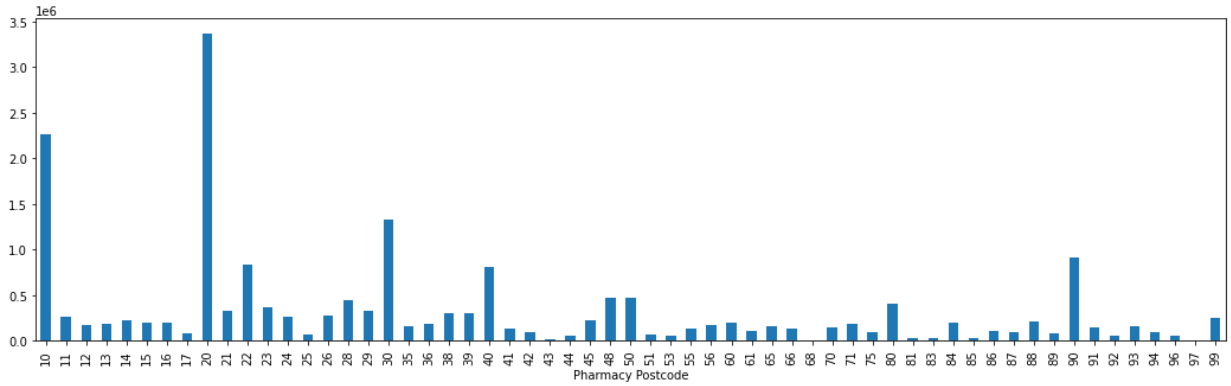
```
Pharmacy Number
3790968       44711.57
4003923         297.72
4038423       97057.11
4065450     132936.25
4306971       83310.01
                ...
12403926    114463.97
13876641    164522.58
14017410      46324.47
18932010      21789.00
30522741      83020.48
Name: Contribution, Length: 258, dtype: float64
```

In [32]:
```
# Contribution per pharmacy
group_class = df_time.groupby('Pharmacy Postcode')['Contribution'].sum()
print(group_class)
```

```
Pharmacy Postcode
10    2.264715e+06
11    2.655302e+05
12    1.723234e+05
13    1.860746e+05
14    2.178520e+05
15    1.942671e+05
16    1.933811e+05
17    7.862356e+04
20    3.365417e+06
21    3.240315e+05
22    8.356629e+05
23    3.620533e+05
24    2.633554e+05
25    6.335779e+04
26    2.741504e+05
28    4.362255e+05
29    3.206213e+05
30    1.329792e+06
35    1.533233e+05
36    1.768392e+05
38    2.934660e+05
39    2.987792e+05
40    8.013931e+05
41    1.348908e+05
42    9.430006e+04
43    1.082642e+04
44    5.440064e+04
45    2.251044e+05
48    4.669108e+05
50    4.693072e+05
51    6.478158e+04
53    5.272083e+04
55    1.366398e+05
56    1.639595e+05
60    1.911768e+05
61    9.861874e+04
65    1.526182e+05
66    1.332580e+05
68    4.013000e+02
70    1.492732e+05
71    1.822786e+05
75    9.648078e+04
80    4.010323e+05
81    2.413576e+04
83    3.067345e+04
84    1.985886e+05
85    2.970761e+04
86    9.885555e+04
87    8.552189e+04
88    2.025448e+05
89    8.495005e+04
90    9.100946e+05
```

```
91    1.382438e+05
92    4.719162e+04
93    1.619310e+05
94    9.812250e+04
96    4.921750e+04
97    9.874800e+02
99    2.442643e+05
Name: Contribution, dtype: float64
```

In [34]:
```
group_class.plot.bar(figsize=(18,5))

plt.show()
```



In [37]:
```
# Export the data to csv
# df_time.to_csv('df_time.csv')
```

In [96]:
```
gk = df_time.groupby('Pharmacy Number')
```

In [100]:
```
type(gk)
```

Out[100]:
```
pandas.core.groupby.generic.DataFrameGroupBy
```

In [98]:
```
gk.first()
```

Out[98]:

Pharmacy Number	Delivery Date	Delivery Time	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	ATC Code
3790968	01/02/2020	10:59	21	2019	1	2663532	ROTARIX SUSP BUV 1,5 ML	J07BH01
4003923	03/17/2020	19:27	10	2016	1	2622314	AMOCLANEEG 250MG/62,5MG/5ML DRINKB.SUSP 100ML	J01CR02
4038423	01/02/2020	08:57	20	1966	2	2582435	ZANICOMBO 10 MG/10 MG FILMOMH TABL 98	C09BB02
4065450	01/02/2020	09:50	20	1966	2	3926664	ALPHARIX TETRA OPL INJ VOORGEV.SPUIT 0,5 ML ...	J07BB02
4306971	01/02/2020	09:51	22	1954	1	3641040	ROSUVASTATINE MYLAN 10MG FILMOMH TABL 98	C10AA07
...	...	...	...	...	...	...	...	...
12403926	01/03/2020	00:00	11	1922	1	119065	MARCOUMAR COMP 25 X 3 MG	B01AA04
13876641	01/02/2020	09:37	39	1971	1	1499185	VALTRAN GUTT BUV 1 X 60 ML	N02AX01
14017410	01/02/2020	09:39	10	1947	1	2622264	AMOCLANEEG 875MG/125MG FILMOMH TABL 20	J01CR02
18932010	01/02/2020	12:45	20	1972	2	639880	MONURIL DOS PULV OR 1 X 3 G	J01XX01
30522741	01/02/2020	00:00	20	1927	2	100974	ALDACTONE COMP 50 X 25 MG	C03DA01

258 rows × 15 columns

Forecasting

In [46]:
```
df_test.shape
```

Out[46]:
```
(242, 2)
```

In [47]:
```
df_test.head()
```

Out[47]:

	delivery_date	Price
0	2020-01-01	5381.65
1	2020-01-02	376623.48
2	2020-01-03	440875.85
3	2020-01-04	206925.18
4	2020-01-05	6499.61

In [48]:
```
df_test.dtypes
```

Out[48]:
```
delivery_date    datetime64[ns]
Price                   float64
dtype: object
```

In [50]:
```
df_forecast = df_test.set_index('delivery_date')
df_forecast.index
```

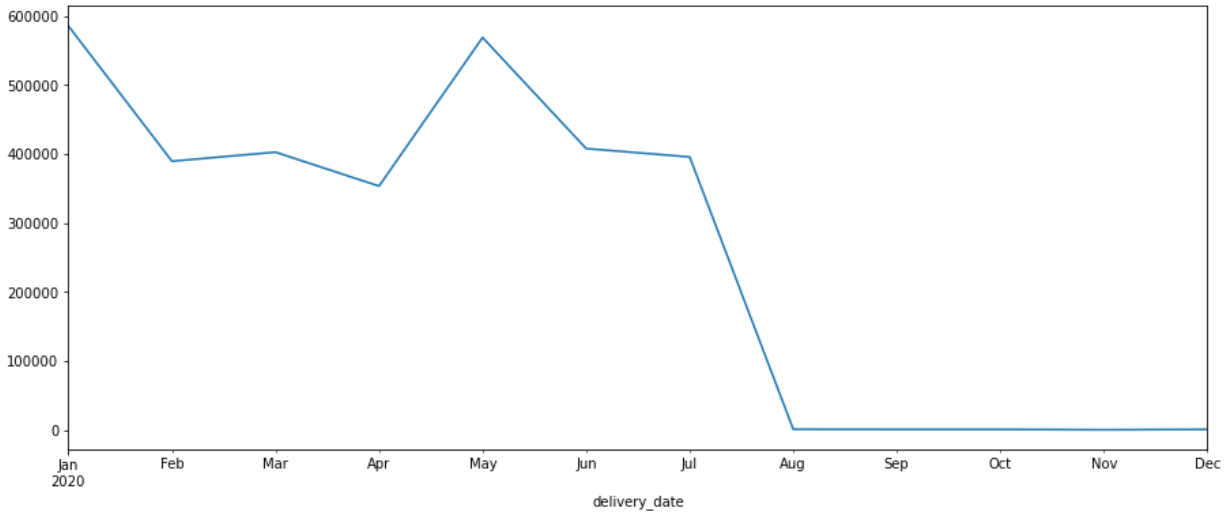
Out[50]:
```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06', '2020-01-07', '2020-01-08',
               '2020-01-09', '2020-01-10',
               ...
               '2020-11-02', '2020-11-03', '2020-11-04', '2020-11-05',
               '2020-11-06', '2020-11-07', '2020-12-02', '2020-12-03',
               '2020-12-05', '2020-12-06'],
              dtype='datetime64[ns]', name='delivery_date', length=242, f
req=None)
```

In [51]:
```
y = df_forecast['Price'].resample('MS').mean()
```

In [52]:
```
y
```

Out[52]:
```
delivery_date
2020-01-01    585833.325484
2020-02-01    389790.941724
2020-03-01    402866.722581
2020-04-01    353722.499000
2020-05-01    569010.294839
2020-06-01    408099.771667
2020-07-01    395956.810968
2020-08-01      1331.538333
2020-09-01      1155.503333
2020-10-01      1165.675000
2020-11-01       641.664286
2020-12-01      1142.345000
Freq: MS, Name: Price, dtype: float64
```

In [53]:
```
y.plot(figsize=(15, 6))
plt.show()
```



In [56]:
```
# This will work is you use the big data set because it needs more than 24 r

from pylab import rcParams
import statsmodels.api as sm

rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```

```
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
<ipython-input-56-f9c2fe04efd4> in <module>
      5
      6 rcParams['figure.figsize'] = 18, 8
----> 7 decomposition = sm.tsa.seasonal_decompose(y, model='additive')
      8 fig = decomposition.plot()
      9 plt.show()

~\Anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*arg
s, **kwargs)
    197                 else:
    198                     kwargs[new_arg_name] = new_arg_value
--> 199             return func(*args, **kwargs)
    200
    201         return cast(F, wrapper)

~\Anaconda3\lib\site-packages\statsmodels\tsa\seasonal.py in seasonal_dec
ompose(x, model, filt, period, two_sided, extrapolate_trend)
    146         raise ValueError('x must have 2 complete cycles requires
 {0} '
    147                          'observations. x only has {1} '
--> 148                          'observation(s)'.format(2 * pfreq, x.sha
pe[0]))
    149
    150     if filt is None:

ValueError: x must have 2 complete cycles requires 24 observations. x onl
y has 12 observation(s)
```

ARIMA = Autoregressive Integrated Moving Average

In [58]:
```
import itertools

p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d,

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [63]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
            results = mod.fit()
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, resu
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:311.8750677797765
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:4.0
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:2.0
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:4.0
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:4.0
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:6.0
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:4.0
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:6.0
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:278.2665274372251
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:6.0
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:4.0
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:6.0
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:6.0
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:8.0

C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)


ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:6.0
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:8.0
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:269.00096984877376
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:4.0
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:2.0
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:4.0
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:4.0
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:6.0
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:4.0
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:6.0
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:245.0772324273814
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:6.0
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:4.0

C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)
C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)

ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:6.0
```

```
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:6.0
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:8.0
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:6.0
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:8.0
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:295.2718209029121
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:6.0
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:4.0
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:6.0
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:6.0
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:8.0

C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)
C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)
C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)

ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:6.0
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:8.0
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:271.2773812614031
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:8.0
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:6.0
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:8.0
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:8.0
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:10.0
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:8.0
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:10.0
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:270.71409197432547
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:6.0

C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)
C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)
C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)
C:\Users\Angel\Anaconda3\lib\site-packages\statsmodels\base\model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. C
heck mle_retvals
  ConvergenceWarning)

ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:4.0
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:6.0
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:6.0
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:8.0
```

```
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:6.0
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:8.0
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:240.02370245680052
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:8.0
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:6.0
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:8.0
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:8.0
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:10.0
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:8.0
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:10.0
```

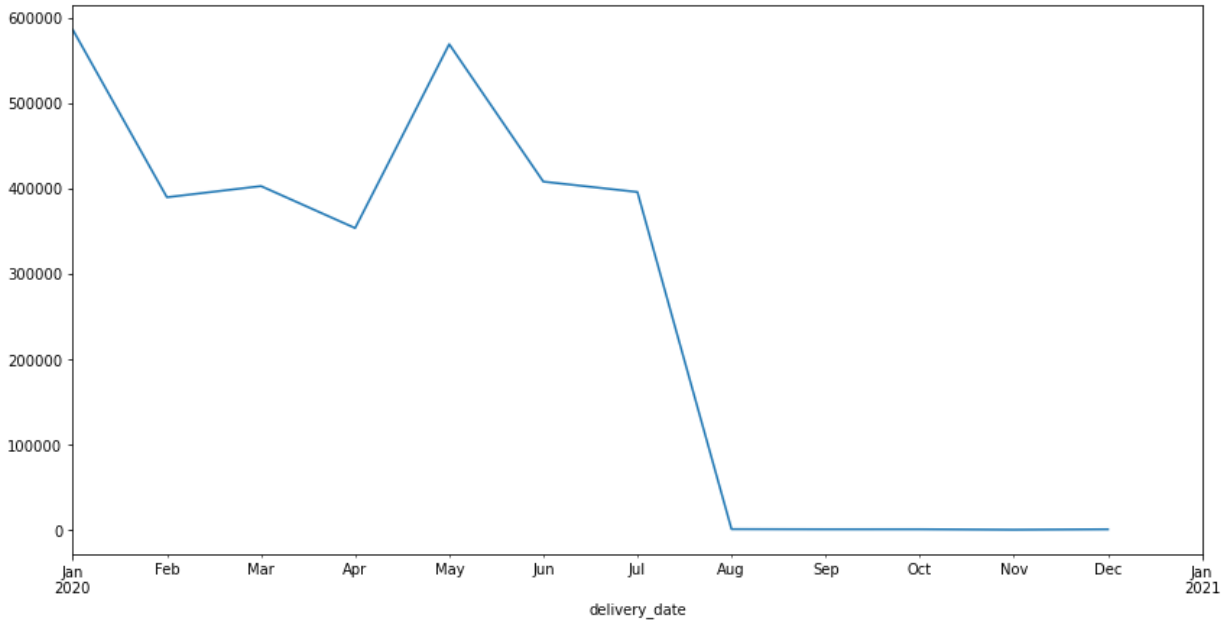
In [64]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(0, 0, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
=====
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----
ar.L1         -0.6688      0.165     -4.043      0.000      -0.993      -
0.345
ma.L1          1.5097      0.764      1.976      0.048       0.012
3.007
sigma2      2.002e+10   1.72e-11   1.16e+21      0.000       2e+10
2e+10
==============================================================================
=====
```

In [66]:
```
pred = results.get_prediction(start=pd.to_datetime('2021-01-01'), dynamic=Fa
pred_ci = pred.conf_int()
ax = y['2020':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, f
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Sales')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-66-73ed5768935d> in <module>
      5 ax.fill_between(pred_ci.index,
      6                 pred_ci.iloc[:, 0],
----> 7                 pred_ci.iloc[:, 1], color='k', alpha=.2)
      8 ax.set_xlabel('Date')
      9 ax.set_ylabel('Sales')

~\AppData\Roaming\Python\Python37\site-packages\matplotlib\__init__.py in
inner(ax, data, *args, **kwargs)
   1429     def inner(ax, *args, data=None, **kwargs):
   1430         if data is None:
-> 1431             return func(ax, *map(sanitize_sequence, args), **kwar
gs)
   1432
   1433         bound = new_sig.bind(ax, *args, **kwargs)

~\AppData\Roaming\Python\Python37\site-packages\matplotlib\axes\_axes.py
 in fill_between(self, x, y1, y2, where, interpolate, step, **kwargs)
   5301         return self._fill_between_x_or_y(
   5302             "x", x, y1, y2,
-> 5303             where=where, interpolate=interpolate, step=step, **kw
args)
   5304
   5305     fill_between.__doc__ = _fill_between_x_or_y.__doc__.format(

~\AppData\Roaming\Python\Python37\site-packages\matplotlib\axes\_axes.py
 in _fill_between_x_or_y(self, ind_dir, ind, dep1, dep2, where, interpola
te, step, **kwargs)
   5288
   5289         # now update the datalim and autoscale
-> 5290         pts = np.row_stack([np.column_stack([ind[where], dep1[whe
re]]),
   5291                             np.column_stack([ind[where], dep2[whe
re]])])
   5292         if ind_dir == "y":

<__array_function__ internals> in column_stack(*args, **kwargs)

~\Anaconda3\lib\site-packages\numpy\lib\shape_base.py in column_stack(tu
p)
    654             arr = array(arr, copy=False, subok=True, ndmin=2).T
```

```
    655         arrays.append(arr)
--> 656     return _nx.concatenate(arrays, 1)
    657
    658

<__array_function__ internals> in concatenate(*args, **kwargs)

TypeError: invalid type promotion
```



Adding data

In [68]:
```
header_list = ["Delivery Date", "Delivery Time", "Pharmacy Number", "Pharmac
               "YOB", "Gender", "CNK Code", "Product Name", "ATC Code", "Uni
df_pharmacy_all = pd.read_csv("data_all/ds.csv", names=header_list)
```

In [69]:
```
df_pharmacy_all.shape
```

Out[69]: (22705348, 12)

In [70]:
```
df_pharmacy_all.head(10)
```

Out[70]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	A C
0	01/01/2017	00:00	7341765	21	1923	1	1715119	AMOXICLAV SANDOZ 500MG/125 MG COMP 30	J01C
1	01/01/2017	00:00	7341765	21	1923	1	5520523	WACHTHONORARIUM	
2	01/01/2017	00:00	7341765	21	1925	1	1799931	ZALDIAR 37,5 MG/325 MG FILMOMH TABL 20	N02A
3	01/01/2017	00:00	8272695	16	1930	2	1719400	VASEXTEN CAPS BLIST 28 X 10 MG	C08C
4	01/01/2017	00:00	8272695	16	1933	2	5520523	WACHTHONORARIUM	
5	01/01/2017	00:00	9111423	10	1931	1	1750132	AACIDEXAM 5MG/ML OPL INJ FL INJ 1 X 1ML	H02A
6	01/01/2017	00:00	9111423	10	1935	1	1750132	AACIDEXAM 5MG/ML OPL INJ FL INJ 1 X 1ML	H02A
7	01/01/2017	00:00	8272695	16	1937	2	1715127	AMOXICLAV SANDOZ 875 MG/125 MG COMP 20	J01C
8	01/01/2017	00:00	8272695	16	1937	2	5520523	WACHTHONORARIUM	
9	01/01/2017	00:00	7341765	21	1939	1	5520523	WACHTHONORARIUM	

In [71]:
```
df_pharmacy_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22705348 entries, 0 to 22705347
Data columns (total 12 columns):
 #   Column             Dtype  
---  ------             -----  
 0   Delivery Date      object 
 1   Delivery Time      object 
 2   Pharmacy Number    int64  
 3   Pharmacy Postcode  int64  
 4   YOB                int64  
 5   Gender             int64  
 6   CNK Code           int64  
 7   Product Name       object 
 8   ATC Code           object 
 9   Units              int64  
 10  Price              float64
 11  Contribution       float64
dtypes: float64(2), int64(6), object(4)
memory usage: 2.0+ GB
```

In [74]:
```
# Replace first empty placeholders - not all spaces are NaNs
df_pharmacy_all.replace(' ', '', inplace=True)
# Check for missing values in each column
df_pharmacy_all.isnull().sum()
```

Out[74]:
```
Delivery Date              0
Delivery Time        4941181
Pharmacy Number            0
Pharmacy Postcode          0
YOB                        0
Gender                     0
CNK Code                   0
Product Name               0
ATC Code                   0
Units                      0
Price                      0
Contribution               0
dtype: int64
```

In [75]:
```
missing_value_formats = ["n.a.", "?", "NA", "n/a", "na", "--", "", 0, 0.00]

df_pharmacy_all.replace(
    to_replace=missing_value_formats,
    value=np.nan,
    inplace=True
)
# Check for missing values in each column
df_pharmacy_all.isnull().sum()
```

Out[75]:
```
Delivery Date              0
Delivery Time        4941181
Pharmacy Number            0
Pharmacy Postcode     157072
YOB                     5219
Gender                 78181
CNK Code                   0
Product Name               0
ATC Code              893665
Units                3592232
Price                 904353
Contribution         4399804
dtype: int64
```

In [77]:
```
df = df_pharmacy_all.dropna()
df = df.reset_index(drop=True)
df.isnull().sum()
```

Out[77]:
```
Delivery Date        0
Delivery Time        0
Pharmacy Number      0
Pharmacy Postcode    0
YOB                  0
Gender               0
CNK Code             0
Product Name         0
ATC Code             0
Units                0
Price                0
Contribution         0
dtype: int64
```

In [78]:
```
df.shape
```

Out[78]:
```
(13565923, 12)
```

In [79]:
```
df.dtypes
```

Out[79]:
```
Delivery Date          object
Delivery Time          object
Pharmacy Number         int64
Pharmacy Postcode     float64
YOB                   float64
Gender                float64
CNK Code                int64
Product Name           object
ATC Code               object
Units                 float64
Price                 float64
Contribution          float64
dtype: object
```

In [80]:
```
df = df.astype({'Pharmacy Postcode': 'int', 'YOB': 'int', 'Gender': 'int',
df.dtypes
```

Out[80]:
```
Delivery Date          object
Delivery Time          object
Pharmacy Number         int64
Pharmacy Postcode       int32
YOB                     int32
Gender                  int32
CNK Code                int64
Product Name           object
ATC Code               object
Units                   int32
Price                 float64
Contribution          float64
dtype: object
```

In [81]:
```
df['delivery_date']= pd.to_datetime(df['Delivery Date'])
df['delivery_date_time'] = pd.to_datetime(df['Delivery Date'] + ' ' + df['De
df.head()
```

Out[81]:

	Delivery Date	Delivery Time	Pharmacy Number	Pharmacy Postcode	YOB	Gender	CNK Code	Product Name	ATC Code	Unit
0	01/01/2017	00:00	7341765	21	1923	1	1715119	AMOXICLAV SANDOZ 500MG/125 MG COMP 30	J01CR02	3
1	01/01/2017	00:00	7341765	21	1925	1	1799931	ZALDIAR 37,5 MG/325 MG FILMOMH TABL 20	N02AJ13	2
2	01/01/2017	00:00	8272695	16	1930	2	1719400	VASEXTEN CAPS BLIST 28 X 10 MG	C08CA12	2
3	01/01/2017	00:00	9111423	10	1931	1	1750132	AACIDEXAM 5MG/ML OPL INJ FL INJ 1 X 1ML	H02AB02	
4	01/01/2017	00:00	9111423	10	1935	1	1750132	AACIDEXAM 5MG/ML OPL INJ FL INJ 1 X 1ML	H02AB02	

In [82]:
```
group_by_price = df.groupby('delivery_date')['Price'].sum()
print(group_by_price)
```

```
delivery_date
2017-01-01      5255.78
2017-01-02    290603.19
2017-01-03    433230.18
2017-01-04    402280.87
2017-01-05    432374.46
                 ...
2019-12-27    383708.80
2019-12-28    170061.09
2019-12-29      9705.90
2019-12-30    427991.49
2019-12-31    296502.03
Name: Price, Length: 1095, dtype: float64
```

In [83]:
```
df_test = group_by_price.to_frame().reset_index()
df_test.shape
```

Out[83]:
```
(1095, 2)
```

In [84]:
```
df_test.dtypes
```

Out[84]:
```
delivery_date    datetime64[ns]
Price                   float64
dtype: object
```

In [85]:
```
df_test.head()
```

Out[85]:

	delivery_date	Price
0	2017-01-01	5255.78
1	2017-01-02	290603.19
2	2017-01-03	433230.18
3	2017-01-04	402280.87
4	2017-01-05	432374.46

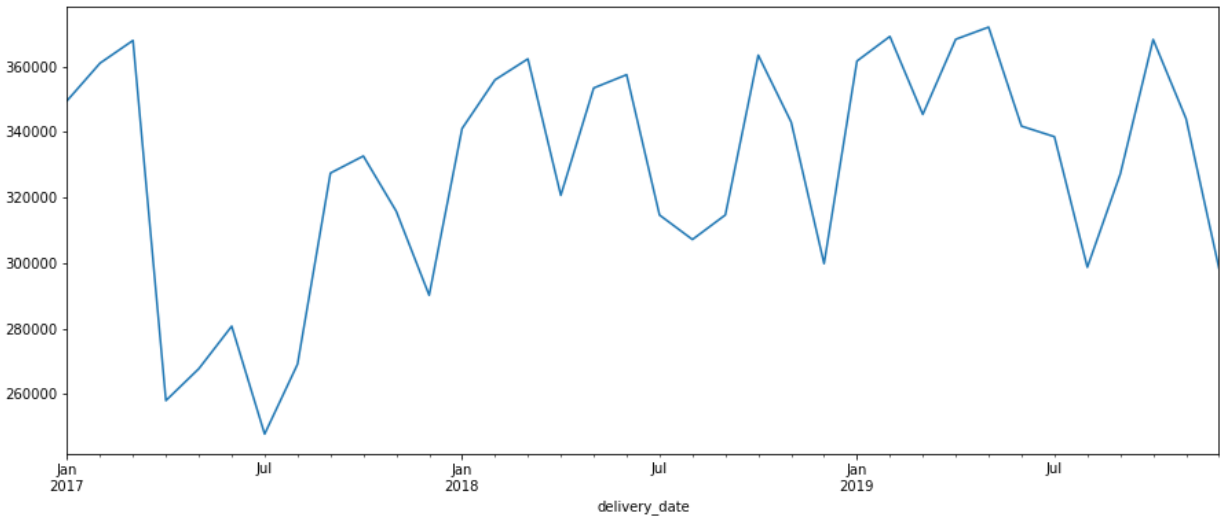
In [86]:
```
df_forecast = df_test.set_index('delivery_date')
df_forecast.index
```

Out[86]:
```
DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03', '2017-01-04',
               '2017-01-05', '2017-01-06', '2017-01-07', '2017-01-08',
               '2017-01-09', '2017-01-10',
               ...
               '2019-12-22', '2019-12-23', '2019-12-24', '2019-12-25',
               '2019-12-26', '2019-12-27', '2019-12-28', '2019-12-29',
               '2019-12-30', '2019-12-31'],
              dtype='datetime64[ns]', name='delivery_date', length=1095,
freq=None)
```

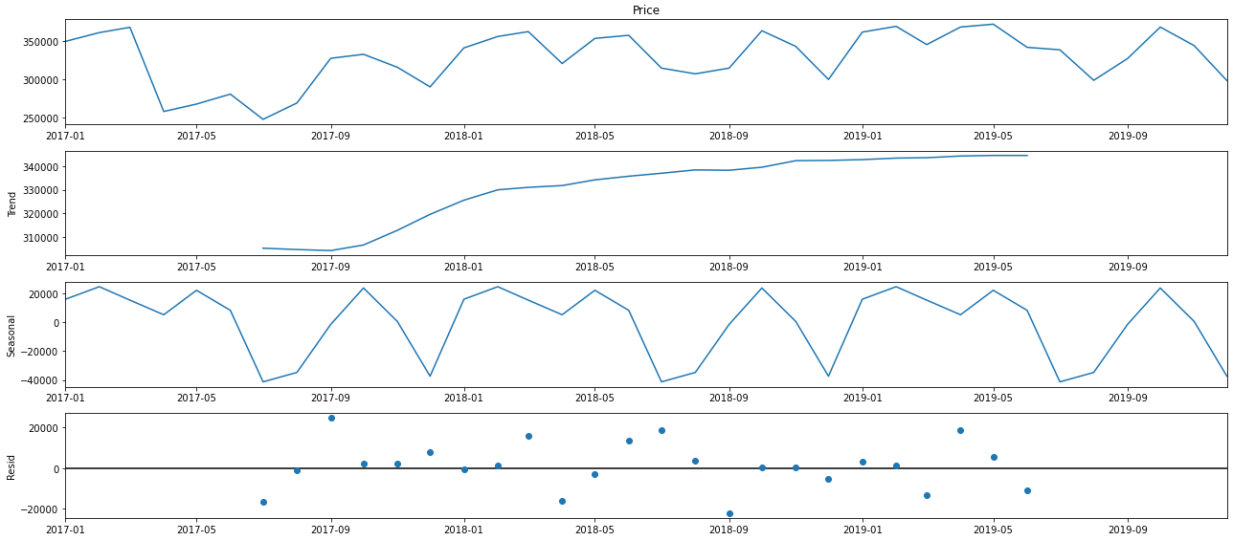
In [87]:
```
y = df_forecast['Price'].resample('MS').mean()
y
```

Out[87]:
```
delivery_date
2017-01-01    349618.564194
2017-02-01    361065.277857
2017-03-01    367987.873548
2017-04-01    257990.435333
2017-05-01    267707.791935
2017-06-01    280700.431667
2017-07-01    247755.005484
2017-08-01    269096.572903
2017-09-01    327476.461000
2017-10-01    332681.516452
2017-11-01    315815.267000
2017-12-01    290164.121290
2018-01-01    341072.110000
2018-02-01    355941.773214
2018-03-01    362356.402581
2018-04-01    320684.037667
2018-05-01    353465.385161
2018-06-01    357523.963667
2018-07-01    314661.001290
2018-08-01    307189.123548
2018-09-01    314689.020000
2018-10-01    363490.013871
2018-11-01    343057.328667
2018-12-01    299826.192903
2019-01-01    361706.682581
2019-02-01    369212.184286
2019-03-01    345401.219032
2019-04-01    368343.055333
2019-05-01    372057.816452
2019-06-01    341808.482000
2019-07-01    338592.025806
2019-08-01    298709.680323
2019-09-01    327156.608000
2019-10-01    368285.736129
2019-11-01    344085.476333
2019-12-01    298426.802903
Freq: MS, Name: Price, dtype: float64
```

In [88]:
```
y.plot(figsize=(15, 6))
plt.show()
```



In [89]:
```
from pylab import rcParams
import statsmodels.api as sm

rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```



In [90]:
```
import itertools

p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d,

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [91]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
            results = mod.fit()
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, resu
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:990.8086587822096
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:1307.6843228951734
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:551.4019939595531
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:254.65549729166906
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:568.4168883967332
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:544.4000604404538
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:272.8496870039418
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:256.0638290721414
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:939.5965070144679
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:607.767554638711
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:522.1055932848725
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:234.7643073548988
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:659.8687402774735
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:607.3266312956558
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:278.98904186946146
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:235.21345765405493
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:811.3566711750359
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:515.117769312444
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:520.059900426114
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:237.77688207499557
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:534.8551771754007
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:516.415507097246
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:261.3902421705817
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:237.98364579100183
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:788.7796683338861
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:491.2474109987533
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:488.7425731957118
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:210.6048579263475
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:534.6134398210381
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:490.91693297308
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:256.06302444373347
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:212.60342352773057
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:835.8913653296239
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:539.9413986975621
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:539.705877470702
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:255.912489574594
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:541.1288626673722
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:540.9674473078671
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:254.9799411682497
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:256.8245461419579
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:813.3679403129245
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:515.8404036663094
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:513.6506106381179
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:234.0766789254978
```

```
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:537.5851670613832
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:514.0381821035269
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:253.9775371718073
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:233.78576975772725
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:812.8154016373703
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:517.4889954969456
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:520.3042144070739
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:236.91658312584525
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:517.6953384876224
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:518.9213025907186
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:237.24902498236287
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:238.1895473631557
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:790.7886982250402
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:493.17540642583737
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:490.67646519068546
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:211.44795705478626
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:513.7010712331055
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:492.81499372945586
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:234.44416649225022
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:213.40315592932183
```

In [92]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(0, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

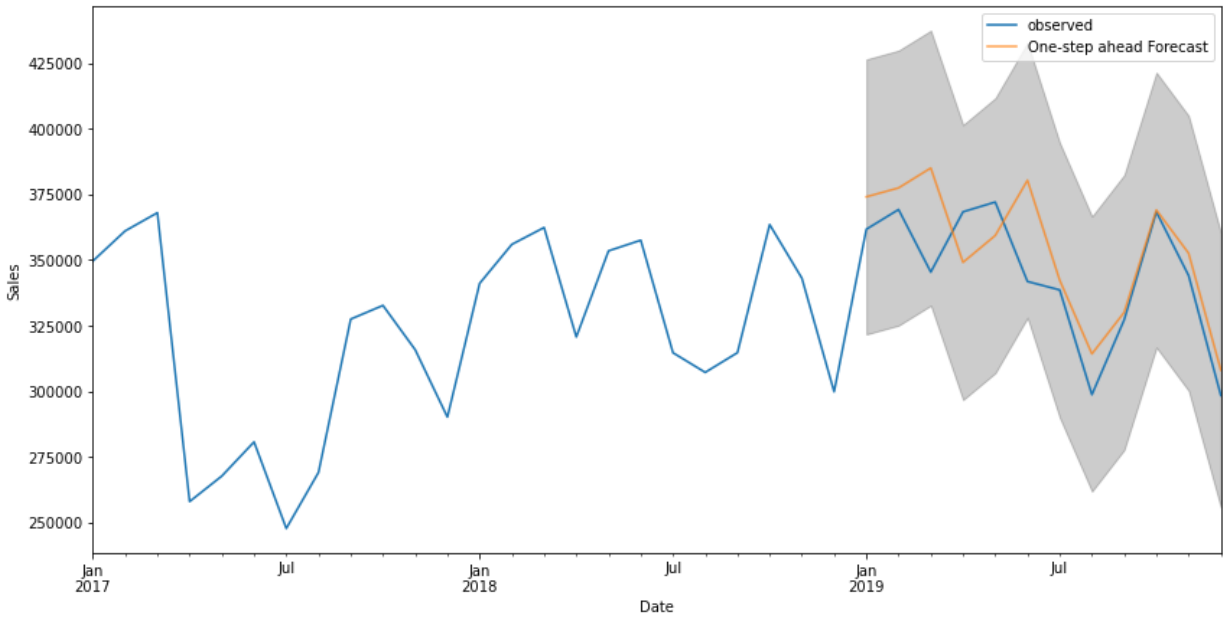
```
==============================================================================
=====
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----
ar.L1         -0.4628      1.008     -0.459      0.646      -2.438
1.512
ma.L1         -0.7254      0.332     -2.184      0.029      -1.376      -
0.074
ma.S.L12       0.0366      0.612      0.060      0.952      -1.163
1.236
sigma2       7.14e+08      5e-10   1.43e+18      0.000    7.14e+08    7.1
4e+08
==============================================================================
=====
```

In [95]:
```
pred = results.get_prediction(start=pd.to_datetime('2020-01-01'), dynamic=Fa
pred_ci = pred.conf_int()
pred_ci
```

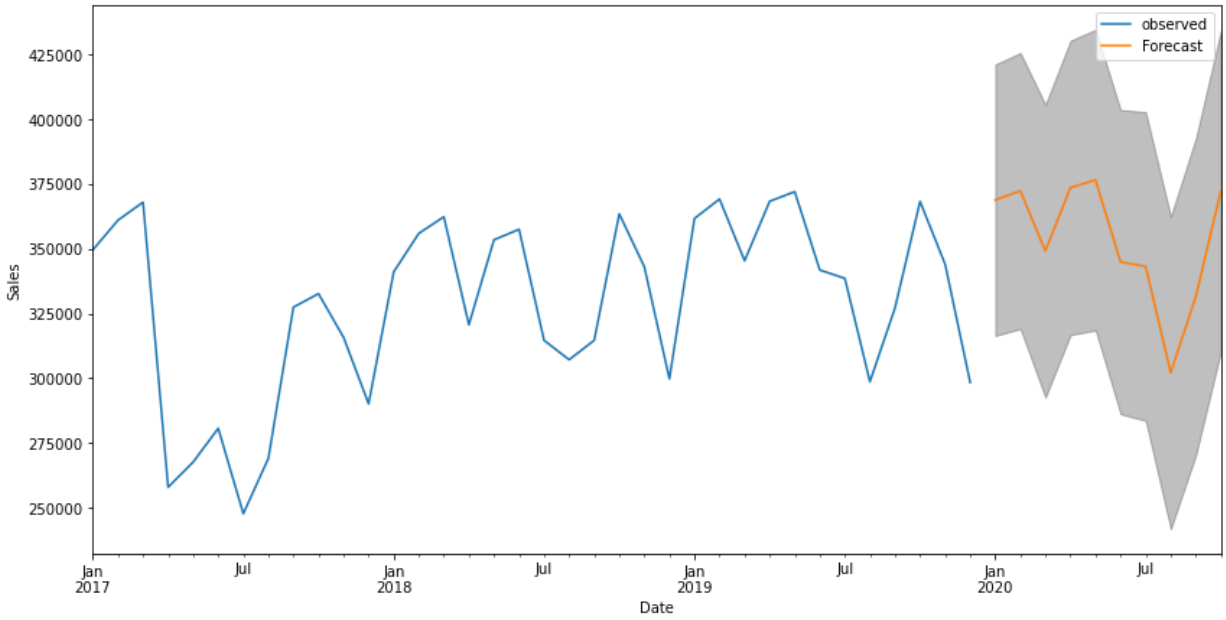
Out[95]:

	lower Price	upper Price
2020-01-01	316436.73451	421178.157188

In [102]:
```
pred = results.get_prediction(start=pd.to_datetime('2019-01-01'), dynamic=Fa
pred_ci = pred.conf_int()

ax = y['2017':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, f

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Date')
ax.set_ylabel('Sales')
plt.legend()
plt.show()
```



In [104]:
```
pred_uc = results.get_forecast(steps=10)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='observed', figsize=(14, 7))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('Sales')
plt.legend()
plt.show()
```



In [ ]: