```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from datetime import datetime

df = pd.read_csv('Sales.csv')
df.head()
```

	Date	Time	State	Group	Unit	Sales
0	01-Oct-20	Morning	WA	Kids	8	20000
1	01-Oct-20	Morning	WA	Men	8	20000
2	01-Oct-20	Morning	WA	Women	4	10000
3	01-Oct-20	Morning	WA	Seniors	15	37500
4	01-Oct-20	Afternoon	WA	Kids	3	7500

```
# The 'object' data type is the base class for all other classes in
Python
# 'object' is the generic datatype class (object data type is the most
general dtype.)
# Some common data types in Python include:
# Numeric: int, float, complex
# Sequence: list, tuple, str
# Mapping: dict
# Set: set
# Boolean: bool

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7560 entries, 0 to 7559
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    7560 non-null   object
 1   Time    7560 non-null   object
 2   State   7560 non-null   object
 3   Group   7560 non-null   object
 4   Unit    7560 non-null   int64
 5   Sales   7560 non-null   int64
dtypes: int64(2), object(4)
memory usage: 354.5+ KB
```

```
df.shape
```

```
(7560, 6)
```

```
df.describe().T
```

```
        count          mean           std     min      25%      50%
75%  \
Unit   7560.0     18.005423     12.901403     2.0      8.0     14.0
26.0
Sales  7560.0  45013.558201  32253.506944  5000.0  20000.0  35000.0
65000.0

            max
Unit       65.0
Sales  162500.0
```

```
# Chk for duplicates
df[ df.duplicated() ]
```

```
Empty DataFrame
Columns: [Date, Time, State, Group, Unit, Sales]
Index: []
```

```
# The dataset does not contain duplicates

# Check for any invalid entries

df['Time'].unique()
```

```
array([' Morning', ' Afternoon', ' Evening'], dtype=object)
```

```
df['State'].unique()
```

```
array([' WA', 'AZ', 'FL', 'KY', 'CA', 'NY', 'TX'], dtype=object)
```

```
df['Group'].unique()
```

```
array([' Kids', ' Men', ' Women', ' Seniors'], dtype=object)
```

```
# The columns Time, State and Group have valid entries

# Convert the object type column to string
df['Time'] = df['Time'].astype('string')
df['State'] = df['State'].astype('string')
df['Group'] = df['Group'].astype('string')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7560 entries, 0 to 7559
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    7560 non-null   object
 1   Time    7560 non-null   string
 2   State   7560 non-null   string
 3   Group   7560 non-null   string
 4   Unit    7560 non-null   int64
 5   Sales   7560 non-null   int64
```

```
dtypes: int64(2), object(1), string(3)
memory usage: 354.5+ KB
```

```
# Data correction
```

```
# correcting WA in State column
df['State'] = np.where(df['State'] == ' WA', 'WA', df['State'])
df['State'].unique()
```

```
array(['WA', 'AZ', 'FL', 'KY', 'CA', 'NY', 'TX'], dtype=object)
```

```
# Check for null values
df.isnull().sum()
```

```
Date     0
Time     0
State    0
Group    0
Unit     0
Sales    0
dtype: int64
```

```
nan_df = df[df.isna().any(axis='columns')]
nan_df
```

```
Empty DataFrame
Columns: [Date, Time, State, Group, Unit, Sales]
Index: []
```

```
# Find the min and max values of the Unit and Sales column
```

```
unit_min = df.Unit.min()
unit_max = df.Unit.max()
print('Unit : min = ', unit_min, ' Unit : max = ', unit_max)
```

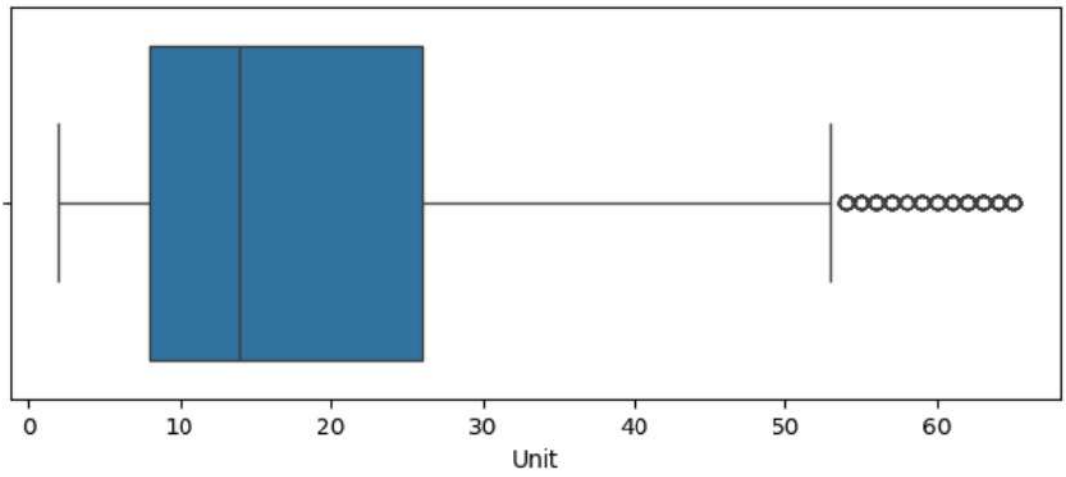
```
Unit : min =  2  Unit : max =  65
```

```
sales_min = df.Sales.min()
sales_max = df.Sales.max()
print('Sales : min = ', sales_min, ' Unit : max = ', sales_max)
```

```
Sales : min =  5000  Unit : max =  162500
```

```
# Outliers : View and Detect for column Unit
```

```
plt.figure(figsize=(8,3))
sns.boxplot(x = 'Unit', data = df);
```




```
Q1 = df.Unit.quantile(0.25)
Q3 = df.Unit.quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
18.0
```

```
Unit_lwrRange = Q1 - (1.5 * IQR)
Unit_uprRange = Q3 + (1.5 * IQR)
```

```
print(Unit_lwrRange, Unit_uprRange)
```

```
-19.0 53.0
```

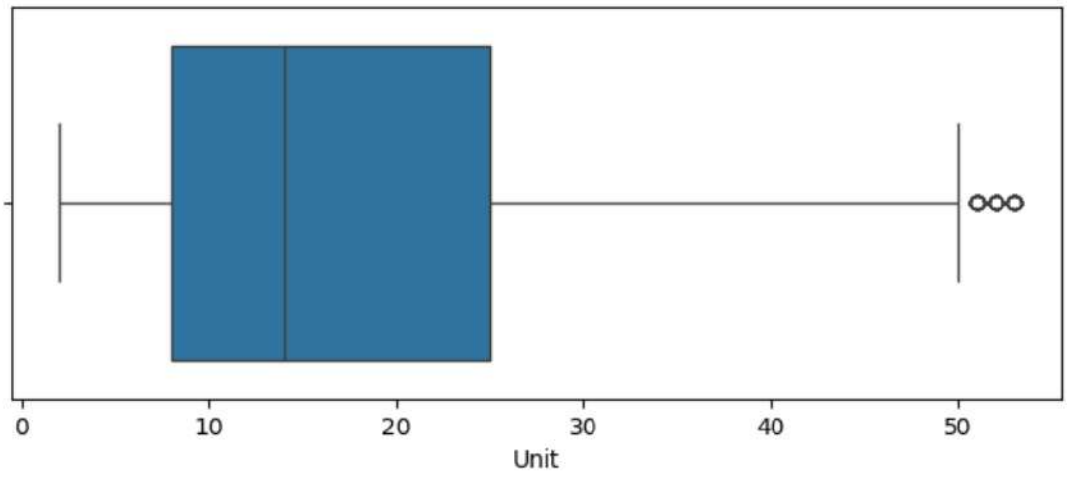
```
df[ (df.Unit < Unit_lwrRange) | (df.Unit > Unit_uprRange)]
```

	Date	Time	State	Group	Unit	Sales
5082	01-Dec-20	Afternoon	KY	Women	63	157500
5083	01-Dec-20	Afternoon	KY	Seniors	62	155000
5161	02-Dec-20	Morning	KY	Men	56	140000
5162	02-Dec-20	Morning	KY	Women	59	147500
5169	02-Dec-20	Evening	KY	Men	64	160000
...	...	...	...	...	...	...
7432	29-Dec-20	Afternoon	KY	Kids	65	162500
7433	29-Dec-20	Afternoon	KY	Men	54	135000
7437	29-Dec-20	Evening	KY	Men	54	135000
7515	30-Dec-20	Morning	KY	Seniors	65	162500
7519	30-Dec-20	Afternoon	KY	Seniors	62	155000

```
[123 rows x 6 columns]
```

```
df.drop(df[ (df.Unit < Unit_lwrRange) | (df.Unit >
Unit_uprRange)].index, inplace=True)
```

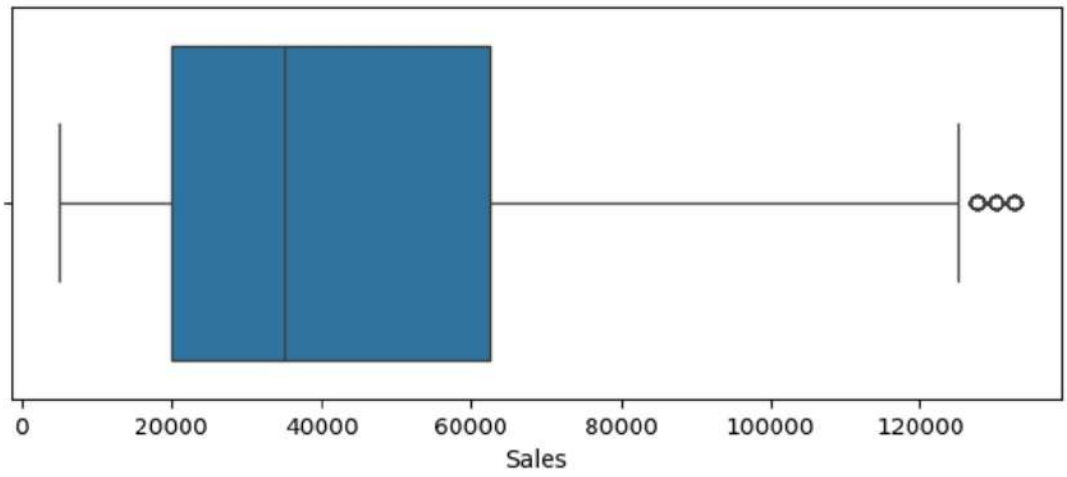
```
plt.figure(figsize=(8,3))
sns.boxplot(x = 'Unit', data = df);
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7437 entries, 0 to 7559
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    7437 non-null   object
 1   Time    7437 non-null   string
 2   State   7437 non-null   object
 3   Group   7437 non-null   string
 4   Unit    7437 non-null   int64
 5   Sales   7437 non-null   int64
dtypes: int64(2), object(2), string(2)
memory usage: 406.7+ KB
```

```
# Outliers : View and Detect for column Sales

plt.figure(figsize=(8,3))
sns.boxplot(x = 'Sales', data = df);
```




```
Q1 = df.Sales.quantile(0.25)
Q3 = df.Sales.quantile(0.75)
IQR = Q3 - Q1
Sales_lwrRange = Q1 - (1.5 * IQR)
Sales_uprRange = Q3 + (1.5 * IQR)
print(Sales_lwrRange, Sales_uprRange)
```

```
-43750.0 126250.0
```

```
sales_outliers = df[ (df.Sales < Sales_lwrRange) | (df.Sales >
Sales_uprRange)]
sales_outliers
```

	Date	Time	State	Group	Unit	Sales
5076	01-Dec-20	Morning	KY	Kids	51	127500
5085	01-Dec-20	Evening	KY	Men	52	130000
5248	03-Dec-20	Afternoon	KY	Kids	53	132500
5334	04-Dec-20	Afternoon	KY	Women	52	130000
5337	04-Dec-20	Evening	KY	Men	52	130000
5338	04-Dec-20	Evening	KY	Women	52	130000
5586	07-Dec-20	Afternoon	KY	Women	51	127500
5749	09-Dec-20	Morning	KY	Men	53	132500
5833	10-Dec-20	Morning	KY	Men	53	132500
5919	11-Dec-20	Morning	KY	Seniors	51	127500
5924	11-Dec-20	Evening	KY	Kids	51	127500
6084	13-Dec-20	Morning	KY	Kids	51	127500
6088	13-Dec-20	Afternoon	KY	Kids	53	132500
6422	17-Dec-20	Morning	KY	Women	52	130000
6423	17-Dec-20	Morning	KY	Seniors	51	127500
6426	17-Dec-20	Afternoon	KY	Women	52	130000
6429	17-Dec-20	Evening	KY	Men	53	132500
6597	19-Dec-20	Evening	KY	Men	51	127500
6676	20-Dec-20	Afternoon	KY	Kids	52	130000
6679	20-Dec-20	Afternoon	KY	Seniors	52	130000
6840	22-Dec-20	Morning	KY	Kids	52	130000

```
6849  22-Dec-20     Evening    KY       Men    53  132500
7013  24-Dec-20   Afternoon    KY       Men    52  130000
7097  25-Dec-20   Afternoon    KY       Men    53  132500
7182  26-Dec-20   Afternoon    KY     Women    53  132500
7261  27-Dec-20     Morning    KY       Men    51  127500
7428  29-Dec-20     Morning    KY      Kids    53  132500
7436  29-Dec-20     Evening    KY      Kids    51  127500
```

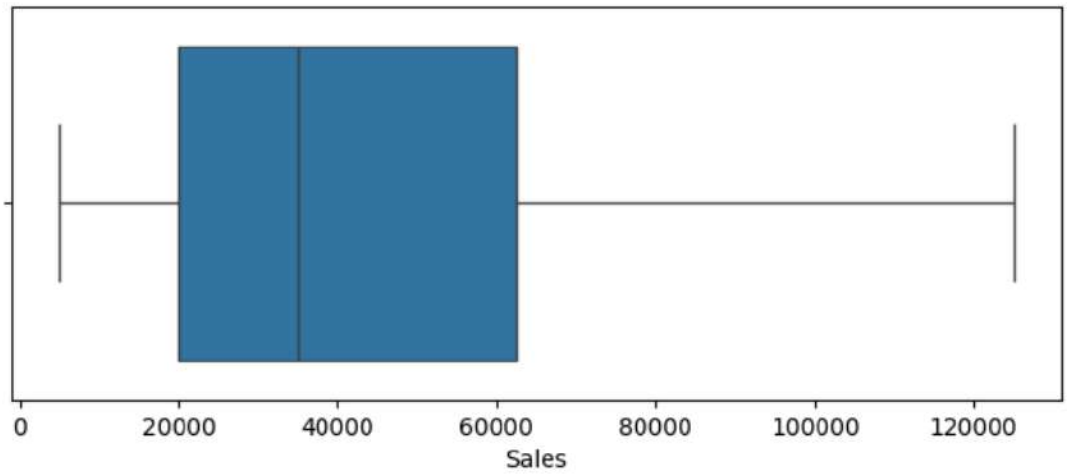
```
len(sales_outliers)
```

```
28
```

```
df.drop(df[ (df.Sales < Sales_lwrRange) | (df.Sales >
Sales_uprRange)].index, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7409 entries, 0 to 7559
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    7409 non-null   object
 1   Time    7409 non-null   string
 2   State   7409 non-null   object
 3   Group   7409 non-null   string
 4   Unit    7409 non-null   int64
 5   Sales   7409 non-null   int64
dtypes: int64(2), object(2), string(2)
memory usage: 405.2+ KB
```

```
plt.figure(figsize=(8,3))
sns.boxplot(x = 'Sales', data = df);
```




```
# Find the Group that is making min and max sales
```

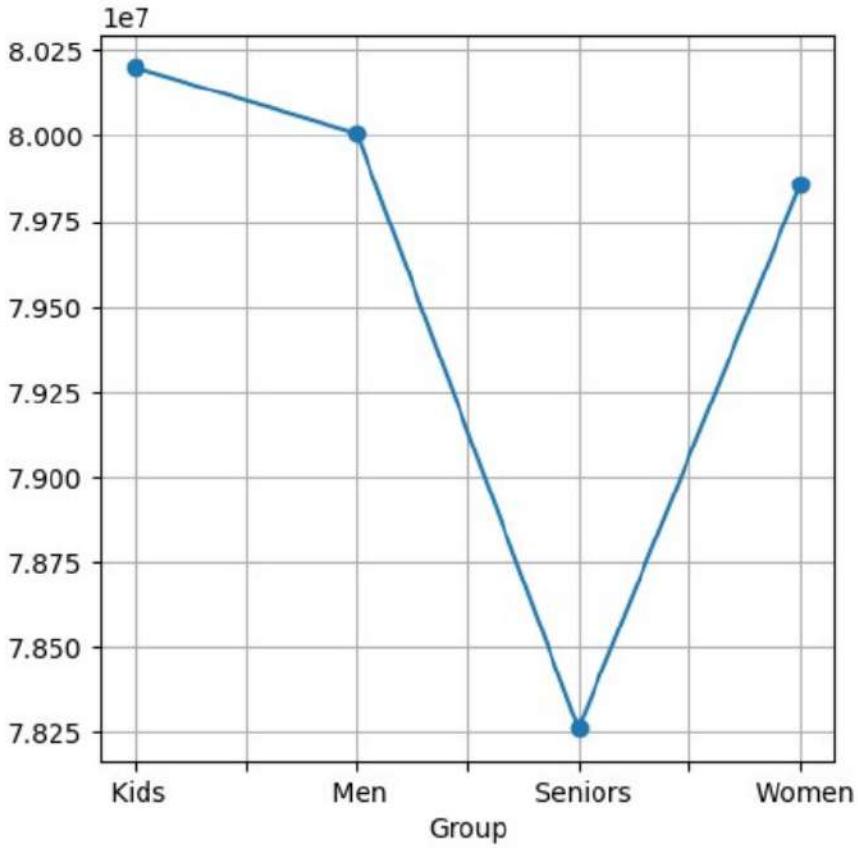
```
group_sales = df.groupby('Group').Sales.sum()
group_sales
```

```
Group
Kids       80200000
Men        80007500
Seniors    78262500
Women      79857500
Name: Sales, dtype: int64
```

```
# idxmin : Return index of first occurrence of minimum over requested
axis.
print('Group with minimum sales : ', group_sales.idxmin())
print('Group with maximum sales : ', group_sales.idxmax())
```

```
Group with minimum sales :   Seniors
Group with maximum sales :   Kids
```

```
group_sales.plot(kind = 'line', marker='o', figsize=(5,5))
plt.grid()
plt.show()
```




```
# We observed that Max sales are for group Kids and Min sales are for
group Seniors
```

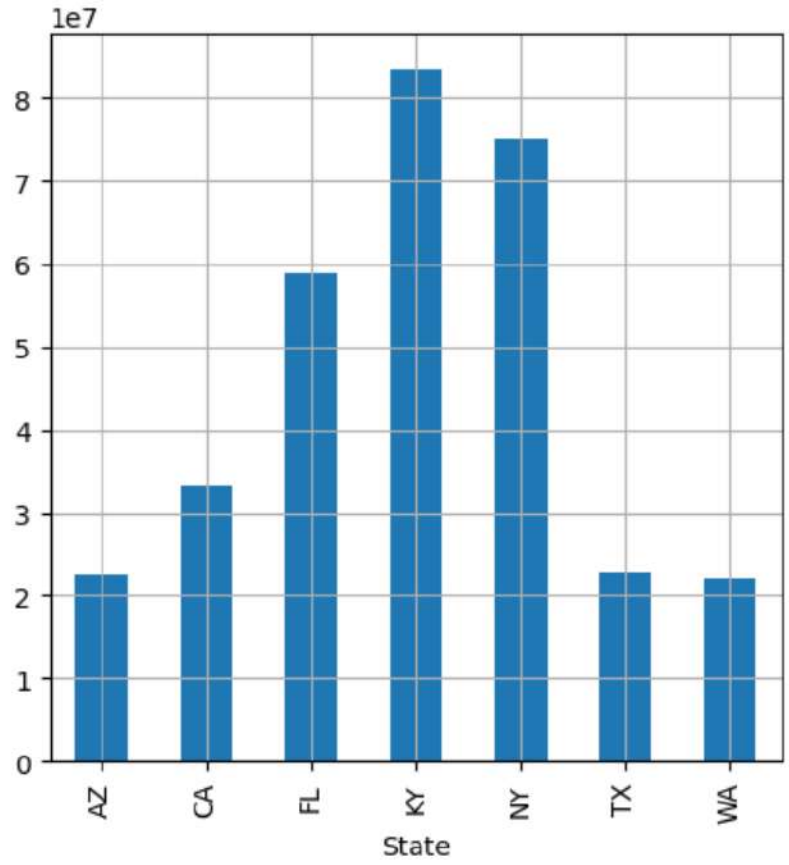
```
# Find the State that is making min and max sales

state_sales = df.groupby('State').Sales.sum()
state_sales
```

```
State
AZ    22580000
CA    33417500
FL    58857500
KY    83590000
NY    74970000
TX    22760000
WA    22152500
Name: Sales, dtype: int64
```

```
print('State with minimum sales : ', state_sales.idxmin())
print('State with maximum sales : ', state_sales.idxmax())
```

```
State with minimum sales :  WA
State with maximum sales :  KY
```

```
state_sales.plot(kind = 'bar', figsize=(5,5))
plt.grid()
plt.show()
```



```
# Get year and month from the Date and prepare various plots
# Date column type is object and format is '04-Dec-20'

df['Date'] = pd.to_datetime(df['Date'], format='%d-%b-%y')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7409 entries, 0 to 7559
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    7409 non-null   datetime64[ns]
 1   Time    7409 non-null   string        
 2   State   7409 non-null   object        
 3   Group   7409 non-null   string        
 4   Unit    7409 non-null   int64         
 5   Sales   7409 non-null   int64         
dtypes: datetime64[ns](1), int64(2), object(1), string(2)
memory usage: 405.2+ KB
```

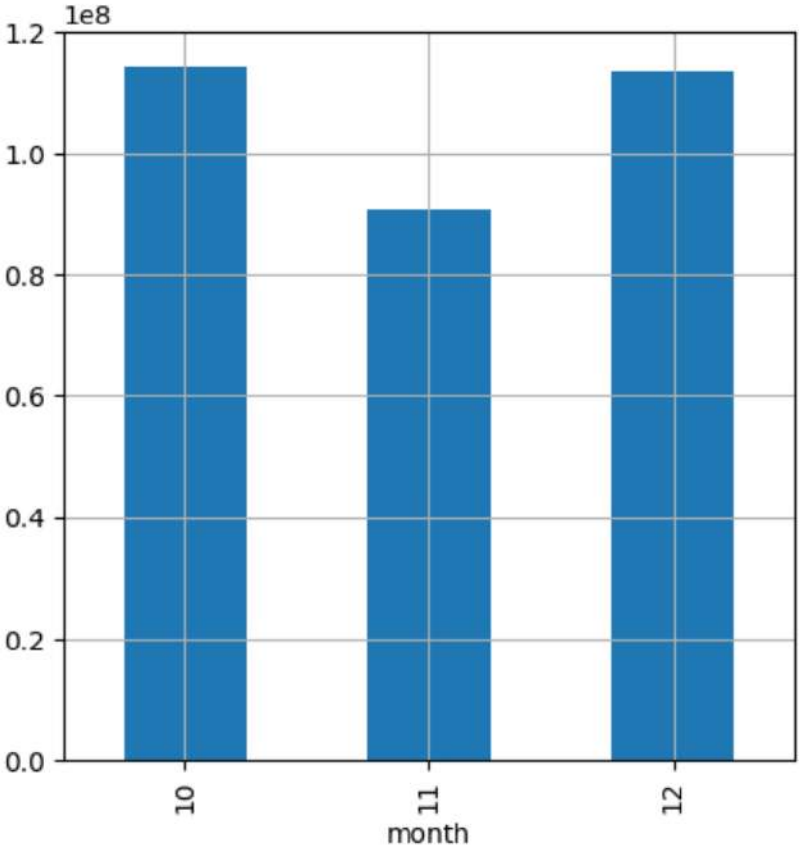
```
df['year'] = df.Date.dt.year
df['month'] = df.Date.dt.month
df.head()
```

	Date	Time	State	Group	Unit	Sales	year	month
0	2020-10-01	Morning	WA	Kids	8	20000	2020	10
1	2020-10-01	Morning	WA	Men	8	20000	2020	10
2	2020-10-01	Morning	WA	Women	4	10000	2020	10
3	2020-10-01	Morning	WA	Seniors	15	37500	2020	10
4	2020-10-01	Afternoon	WA	Kids	3	7500	2020	10

```
month_sales = df.groupby('month').Sales.sum()
month_sales
```

```
month
10    114290000
11     90682500
12    113355000
Name: Sales, dtype: int64
```

```
month_sales.plot(kind = 'bar', figsize=(5,5))
plt.grid()
plt.show()
```

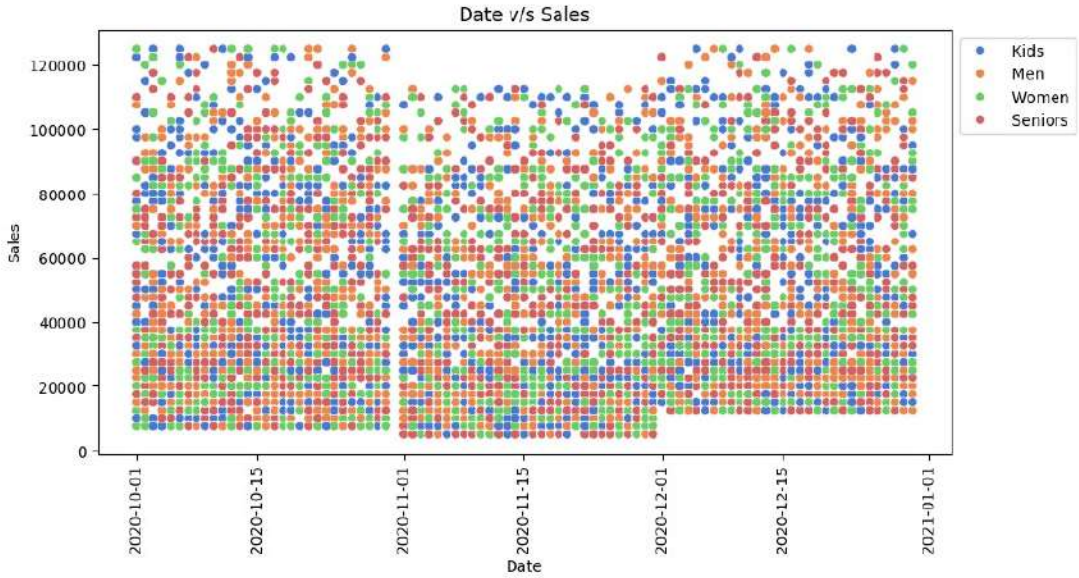



```
# Month of october has maximum sales

# Date v/s Sales lineplot

df['day'] = df.Date.dt.day
df.head()
```

```
        Date        Time State     Group  Unit  Sales  year  month
day
0 2020-10-01     Morning    WA      Kids     8  20000  2020     10
1
1 2020-10-01     Morning    WA       Men     8  20000  2020     10
1
2 2020-10-01     Morning    WA     Women     4  10000  2020     10
1
3 2020-10-01     Morning    WA   Seniors    15  37500  2020     10
1
4 2020-10-01   Afternoon    WA      Kids     3   7500  2020     10
1
```

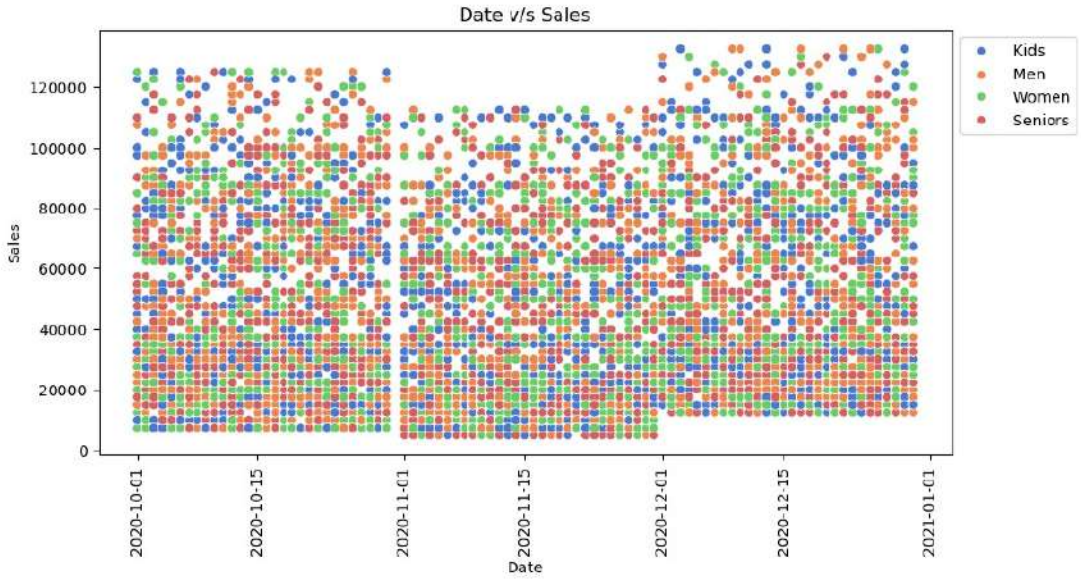
```
plt.figure(figsize=(10, 5))
sns.scatterplot(x = 'Date', y= 'Sales', data = df, palette='muted',
hue = 'Group')
plt.xlabel('Date')
```

```
plt.ylabel('Sales')
plt.title('Date v/s Sales')
plt.xticks(rotation = 90)
plt.legend(bbox_to_anchor=(1, 0, 0.1, 1))
plt.show()
```




```
# Pie Chart
```

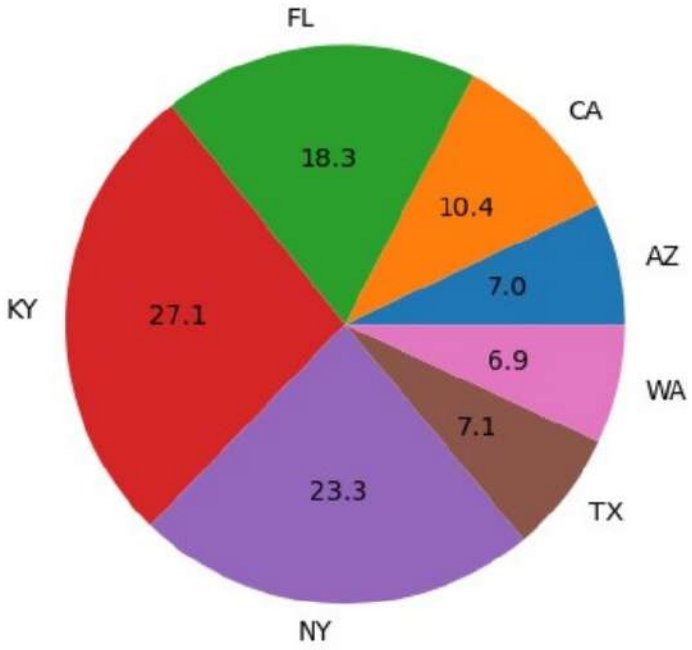
```
# Bar Chart
```



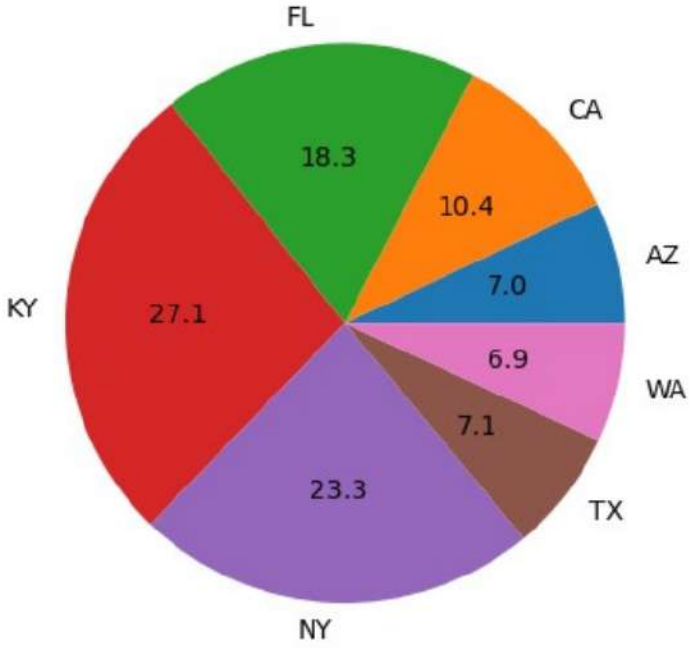

```
group_sales.index
```

```
Index(['AZ', 'CA', 'FL', 'KY', 'NY', 'TX', 'WA'], dtype='object',
name='State')
```

```
plt.pie(group_sales, labels=group_sales.index, autopct='%0.01f');
```




```
plt.pie(state_sales, labels=state_sales.index, autopct='%0.01f');
```




```
plt.figure(figsize=(15, 7))
sns.barplot(data = df, x = 'State', y = 'Sales', hue = 'Group', ci=0)
plt.show()
```