Sales Analysis

September 9, 2023

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[2]: df = pd.read_excel("sales.xlsx")
```

```
[3]: df.head()
```

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

```
[4]: df.shape
```

```
[4]: (7560, 6)
```

1.Data Wrangling

```
[5]: # q.1 Ensure that the data is clean and that there is no missing or incorrect␣
       ↪data.so as
     #per output there is nonull values or incorrect values in data
     df.isnull().sum()
```

```
[5]: Date     0
     Time     0
     State    0
     Group    0
     Unit     0
     Sales    0
     dtype: int64
```

```
[6]: #q.2 Select an appropriate Data Wrangling approach -  data standardization or␣
       ↪data normalization. Perform the standardization or normalization and present␣
       ↪the data.
```

```
df.describe()
```

[6]:

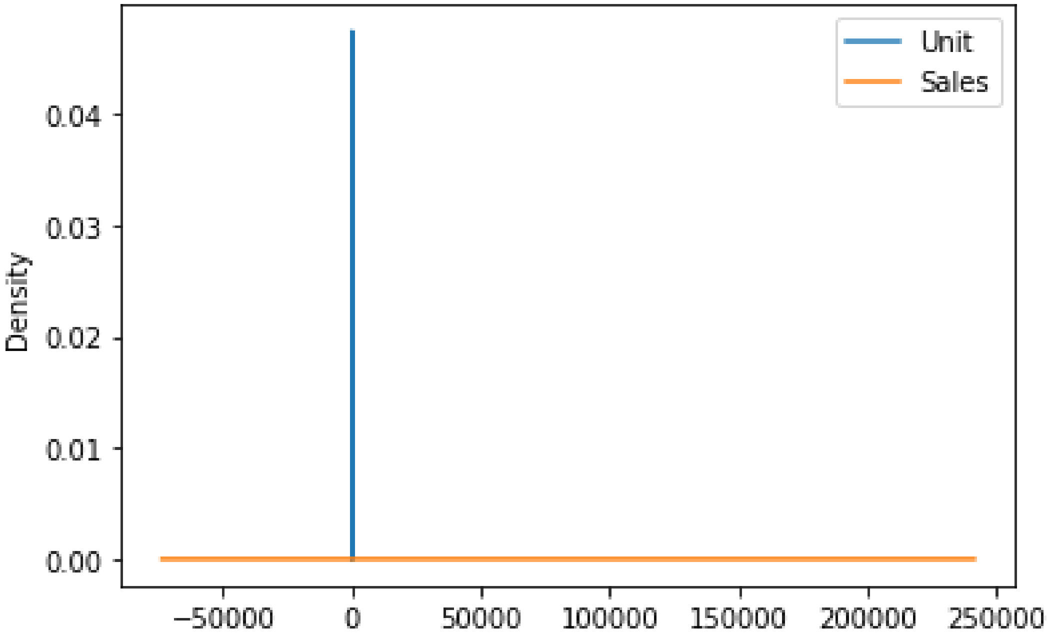
```
              Unit          Sales
count  7560.000000    7560.000000
mean     18.005423   45013.558201
std      12.901403   32253.506944
min       2.000000    5000.000000
25%       8.000000   20000.000000
50%      14.000000   35000.000000
75%      26.000000   65000.000000
max      65.000000  162500.000000
```

[7]:

```
df[['Unit','Sales']].plot.kde()
```

[7]:

```
<AxesSubplot: ylabel='Density'>
```



[8]:

```
from sklearn.preprocessing import StandardScaler
```

[9]:

```
ss = StandardScaler()
```

[10]:

```
newdf=df[['Unit','Sales']]
```

[11]:

```
data_transformed = ss.fit_transform(newdf)
```

[12]:

```
type(data_transformed)
```

[12]:
```
numpy.ndarray
```

[13]:
```
newdf = pd.DataFrame(data_transformed,columns = ['Unit','Sales'])
```

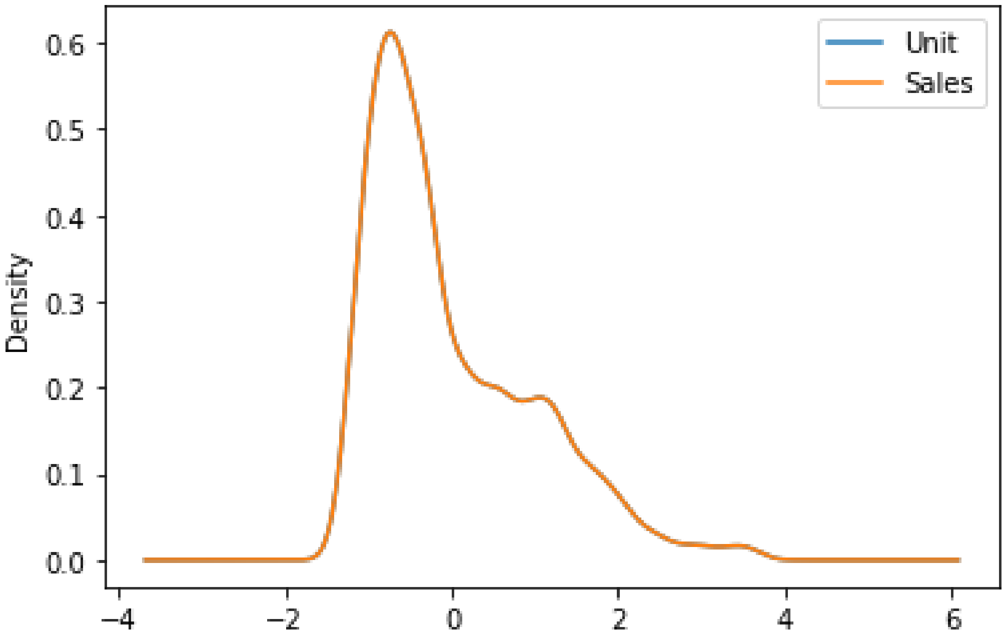
[14]:
```
newdf
```

[14]:
```
          Unit     Sales
0    -0.775581 -0.775581
1    -0.775581 -0.775581
2    -1.085645 -1.085645
3    -0.232969 -0.232969
4    -1.163162 -1.163162
…          …        …
7555 -0.310485 -0.310485
7556 -0.232969 -0.232969
7557 -0.232969 -0.232969
7558 -0.543033 -0.543033
7559 -0.388001 -0.388001

[7560 rows x 2 columns]
```

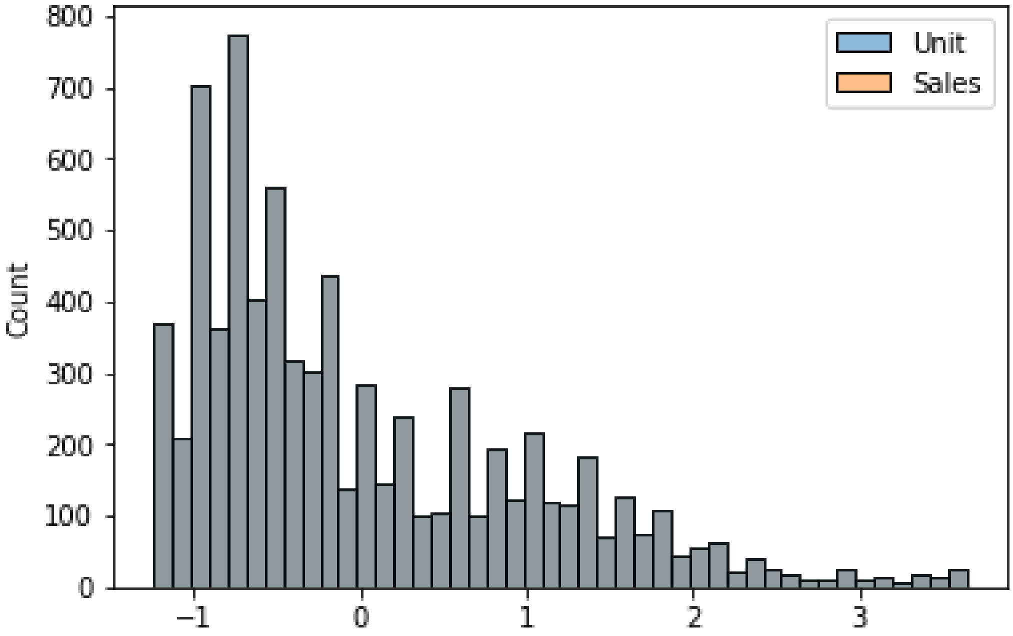
[15]:
```
newdf.plot.kde()
```

[15]:
```
<AxesSubplot: ylabel='Density'>
```



[16]:
```
sns.histplot(newdf)
```

[16]:
```
<AxesSubplot: ylabel='Count'>
```



[17]:
```
from sklearn.preprocessing import MinMaxScaler
```

[18]:
```
mm =MinMaxScaler()
```

[19]:
```
normdf = mm.fit_transform(newdf)
```

[20]:
```
testdf = pd.DataFrame(normdf,columns=newdf.columns)
```

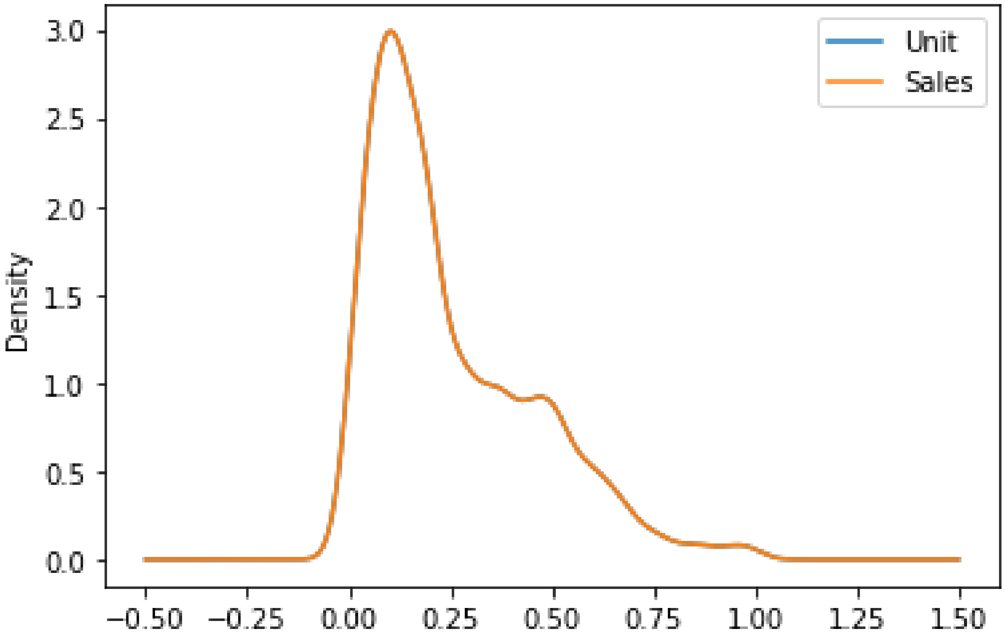
[21]:
```
testdf
```

[21]:
```
          Unit     Sales
0     0.095238  0.095238
1     0.095238  0.095238
2     0.031746  0.031746
3     0.206349  0.206349
4     0.015873  0.015873
…          …         …
7555  0.190476  0.190476
7556  0.206349  0.206349
7557  0.206349  0.206349
7558  0.142857  0.142857
7559  0.174603  0.174603

[7560 rows x 2 columns]
```

[22]:
```
testdf.plot.kde()
```

[22]:
```
<AxesSubplot: ylabel='Density'>
```



[23]:
```
#ans.standardization and normalization both were presented in the upper section
```

[24]:
```
#q.3 Share your recommendation on the usage of the groupby() function for data␣
 ↪chunking or merging.
```

[25]:
```
df
```

[25]:
```
            Date       Time State    Group  Unit  Sales
0     2020-10-01    Morning    WA     Kids     8  20000
1     2020-10-01    Morning    WA      Men     8  20000
2     2020-10-01    Morning    WA    Women     4  10000
3     2020-10-01    Morning    WA  Seniors    15  37500
4     2020-10-01  Afternoon    WA     Kids     3   7500
…            …          …     …        …     …      …
7555  2020-12-30  Afternoon   TAS  Seniors    14  35000
7556  2020-12-30    Evening   TAS     Kids    15  37500
7557  2020-12-30    Evening   TAS      Men    15  37500
7558  2020-12-30    Evening   TAS    Women    11  27500
7559  2020-12-30    Evening   TAS  Seniors    13  32500

[7560 rows x 6 columns]
```

[26]:
```
state_wise_totalsales=df.groupby('State')["Sales"].sum().sort_values()
```

[27]:
```
# ans.Merging data of State and Sales to visualize data.
state_wise_totalsales
```

[27]:
```
State
 WA      22152500
 NT      22580000
 TAS     22760000
 QLD     33417500
 SA      58857500
 NSW     74970000
 VIC    105565000
Name: Sales, dtype: int64
```

2.Data Analysis

[28]:
```
# 2.1 Perform descriptive statistical analysis on the data (Sales and Unit␣
 ↪columns)
#(Techniques such as mean, median, mode and standard deviation can be used.)
newdf=df[['Unit','Sales']]
```

[29]:
```
newdf
```

[29]:
```
      Unit  Sales
0        8  20000
1        8  20000
2        4  10000
3       15  37500
4        3   7500
…     …     …
7555    14  35000
7556    15  37500
7557    15  37500
7558    11  27500
7559    13  32500

[7560 rows x 2 columns]
```

[30]:
```
newdf.mean()
```

[30]:
```
Unit        18.005423
Sales    45013.558201
dtype: float64
```

[31]:
```
newdf.median()
```

[31]:
```
Unit         14.0
Sales     35000.0
dtype: float64
```

[32]:
```
newdf.mode()
```

[32]:
```
   Unit  Sales
0     9  22500
```

[33]:
```
newdf.std()
```

[33]:
```
Unit         12.901403
Sales     32253.506944
dtype: float64
```

[34]:
```
#2.2 Determine which group is generating the highest sales, and which group is␣
 ↪generating the lowest sales.
grouped_df1 = df.groupby('Group')
grouped_df1.apply(lambda x: x.sort_values(by = 'Sales', ascending=False))
```

[34]:
```
                  Date        Time State   Group  Unit   Sales
Group
 Kids  7432 2020-12-29   Afternoon   VIC    Kids    65  162500
       6340 2020-12-16   Afternoon   VIC    Kids    65  162500
       6928 2020-12-23   Afternoon   VIC    Kids    63  157500
       7008 2020-12-24     Morning   VIC    Kids    63  157500
       7180 2020-12-26   Afternoon   VIC    Kids    63  157500
…                …         …   …     …   …     …
 Women 3366 2020-11-11   Afternoon    WA   Women     2    5000
       3358 2020-11-10     Evening   TAS   Women     2    5000
       3286 2020-11-10     Evening    WA   Women     2    5000
       3686 2020-11-14     Morning   TAS   Women     2    5000
       3130 2020-11-08     Evening    NT   Women     2    5000

[7560 rows x 6 columns]
```

[35]:
```
grouped_df1.apply(lambda x: x.sort_values(by = 'Sales', ascending=False)).max()
```

[35]:
```
Date     2020-12-30 00:00:00
Time                 Morning
State                     WA
Group                  Women
Unit                      65
Sales                 162500
dtype: object
```

[36]:
```
grouped_df1.apply(lambda x: x.sort_values(by = 'Sales', ascending=False)).min()
```

[36]:

```
Date     2020-10-01 00:00:00
Time               Afternoon
State                    NSW
Group                   Kids
Unit                       2
Sales                   5000
dtype: object
```

[37]:

```
#ans 2.2 Determine 'women' group is generating the highest sales, and 'kids' group is generating the lowest sales.
from datetime import datetime
from matplotlib import dates
%matplotlib inline
```

[38]:

```
df['Date'] = pd.to_datetime(df['Date'])
```

[39]:

```
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   datetime64[ns]
 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: datetime64[ns](1), int64(2), object(3)
memory usage: 354.5+ KB
```

[40]:

```
#spliting Date into Day,Month,Year
df['Day'] = pd.DatetimeIndex(df['Date']).day
df['Month']=pd.DatetimeIndex(df['Date']).month
df['Year']=pd.DatetimeIndex(df['Date']).year
```

[41]:

```
df
```

[41]:

```
           Date        Time State     Group  Unit  Sales  Day  Month  Year
0    2020-10-01     Morning    WA      Kids     8  20000    1     10  2020
1    2020-10-01     Morning    WA       Men     8  20000    1     10  2020
2    2020-10-01     Morning    WA     Women     4  10000    1     10  2020
3    2020-10-01     Morning    WA   Seniors    15  37500    1     10  2020
4    2020-10-01   Afternoon    WA      Kids     3   7500    1     10  2020
…         …         …   …       …   …    …  …    …   …
7555 2020-12-30   Afternoon   TAS   Seniors    14  35000   30     12  2020
```

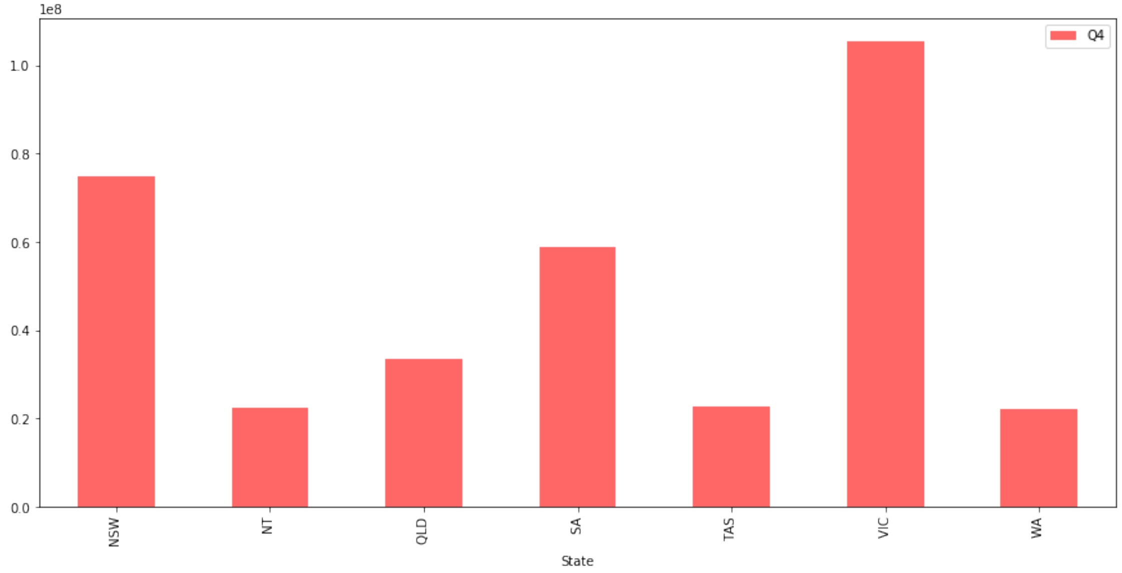
```
7556 2020-12-30     Evening   TAS      Kids     15  37500    30       12   2020
7557 2020-12-30     Evening   TAS       Men     15  37500    30       12   2020
7558 2020-12-30     Evening   TAS     Women     11  27500    30       12   2020
7559 2020-12-30     Evening   TAS   Seniors     13  32500    30       12   2020

[7560 rows x 9 columns]
```

```
[42]: Q4 = df[(df['Date']>='2020-10-01') & (df['Date']<='2020-12-31')].
       ↪groupby('State')['Sales'].sum()
```

```
[45]: plt.figure(figsize = (15,7))

      Q4.plot(kind='bar', alpha = 0.6, color='r', legend = True)
      plt.legend(["Q4"])
      plt.show()
```



```
[46]: df.Date = pd.to_datetime(df.Date)
      df.set_index('Date', inplace=True)
```

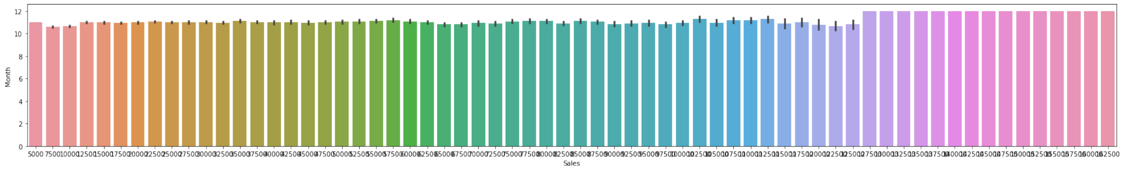
```
[44]: df.columns
```

```
[44]: Index(['Date', 'Time', 'State', 'Group', 'Unit', 'Sales', 'Day', 'Month',
             'Year'],
            dtype='object')
```

```
[48]: df_monthly_report = df[['Sales','Month']]
```

[56]:
```
plt.figure(figsize = (30,4))
sns.barplot(x='Sales', y= "Month", data = df_monthly_report)
```

[56]:
```
<AxesSubplot: xlabel='Sales', ylabel='Month'>
```

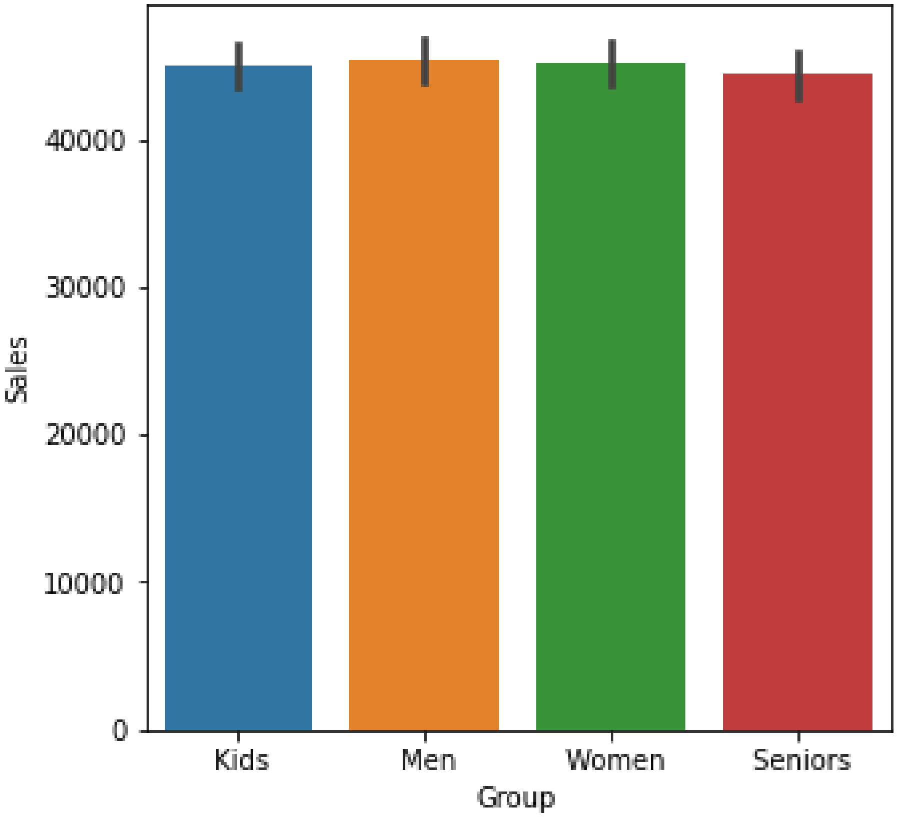


[ ]:
```
#ans. we can clearly see that in the month of december the sales is high.
```

3.Data Visualization

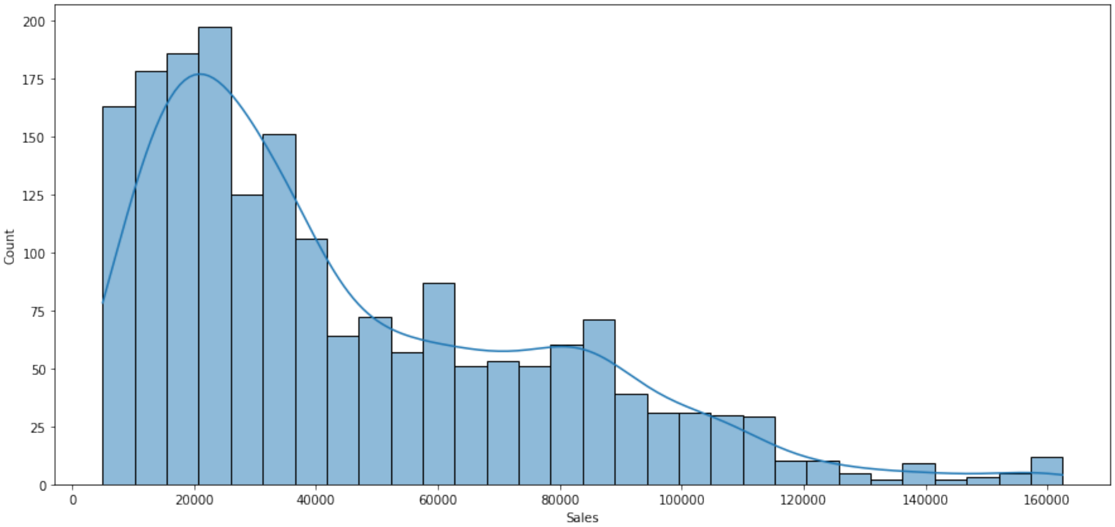
[70]:
```
#3.1 State-wise sales analysis for different groups (kids, women, men, and␣
 ↪seniors)
plt.figure(figsize = (5,5))
sns.barplot(x="Group",y="Sales",data = df)
```

[70]:
```
<AxesSubplot: xlabel='Group', ylabel='Sales'>
```



[80]:

```
std_data = pd.DataFrame(df.groupby('Group')["Sales"].std().
  ↪sort_values(ascending= False))
plt.figure(figsize = (15,7))
sns.histplot(df[df['Group'] == std_data.head(1).index[0]]['Sales'], kde =True,␣
  ↪bins = 30)
plt.show()
```



[82]:

```
#3.3 Time-of-the-day analysis: during which time of the day are sales the␣
  ↪highest, and during which time are sales the lowest?
grouped_df1.apply(lambda x: x.sort_values(by = 'Sales', ascending=False)).max()
```

[82]:

```
Date      2020-12-30 00:00:00
Time                  Morning
State                      WA
Group                   Women
Unit                       65
Sales                  162500
Day                        30
Month                      12
Year                     2020
dtype: object
```

[ ]:

```
grouped_df1.apply(lambda x: x.sort_values(by = 'Sales', ascending=False)).min()
```

[ ]:

```
Date      2020-10-01 00:00:00
Time                Afternoon
```

```
State                  NSW
Group                 Kids
Unit                     2
Sales                 5000
Day                      1
Month                   10
Year                  2020
dtype: object
```