Project 9 - Sales Analysis

January 18, 2023

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
df = pd.read_excel('1673872777_ausapparalsales4thqrt2020.xlsx')
```

[4]:
```
df.head()
```

[4]:

	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

[5]:
```
df.tail()
```

[5]:

	Date	Time	State	Group	Unit	Sales
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

[6]:
```
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.shape
```

[7]:
```
(7560, 6)
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Date     0
Time     0
State    0
Group    0
Unit     0
Sales    0
dtype: int64
```

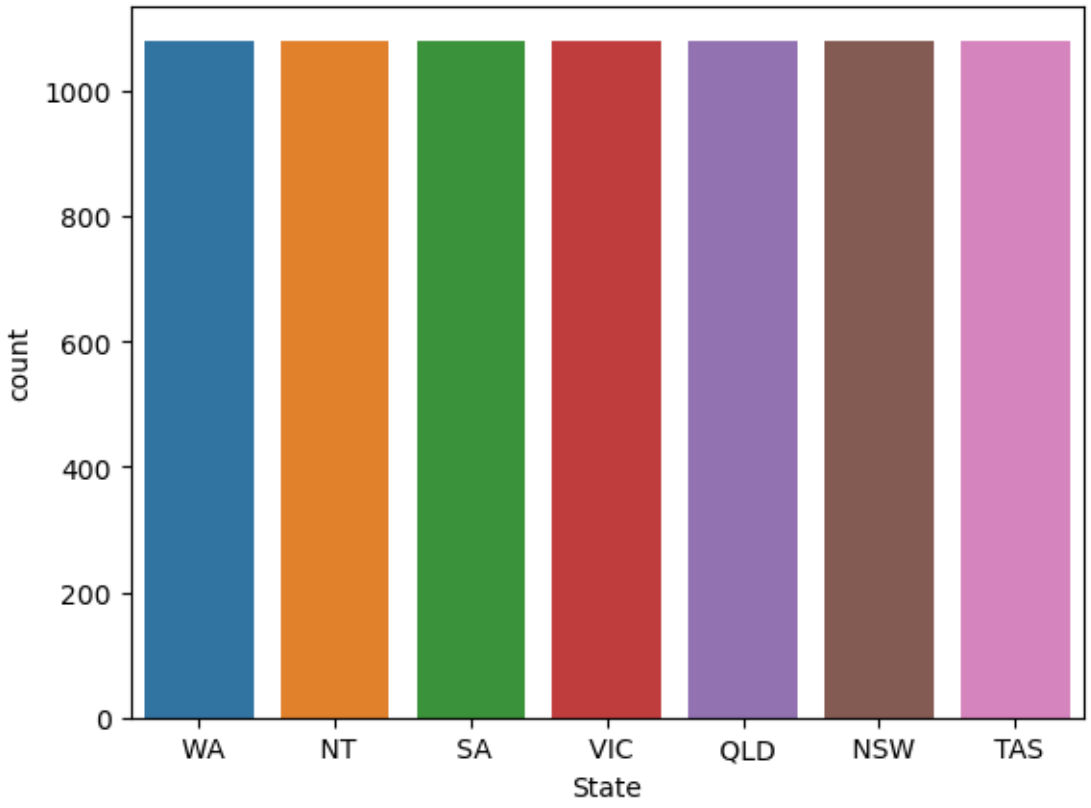
[9]:
```
df.dtypes
```

[9]:
```
Date     datetime64[ns]
Time             object
State            object
Group            object
Unit              int64
Sales             int64
dtype: object
```

[10]:
```
sns.countplot(df['State'])
```

```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

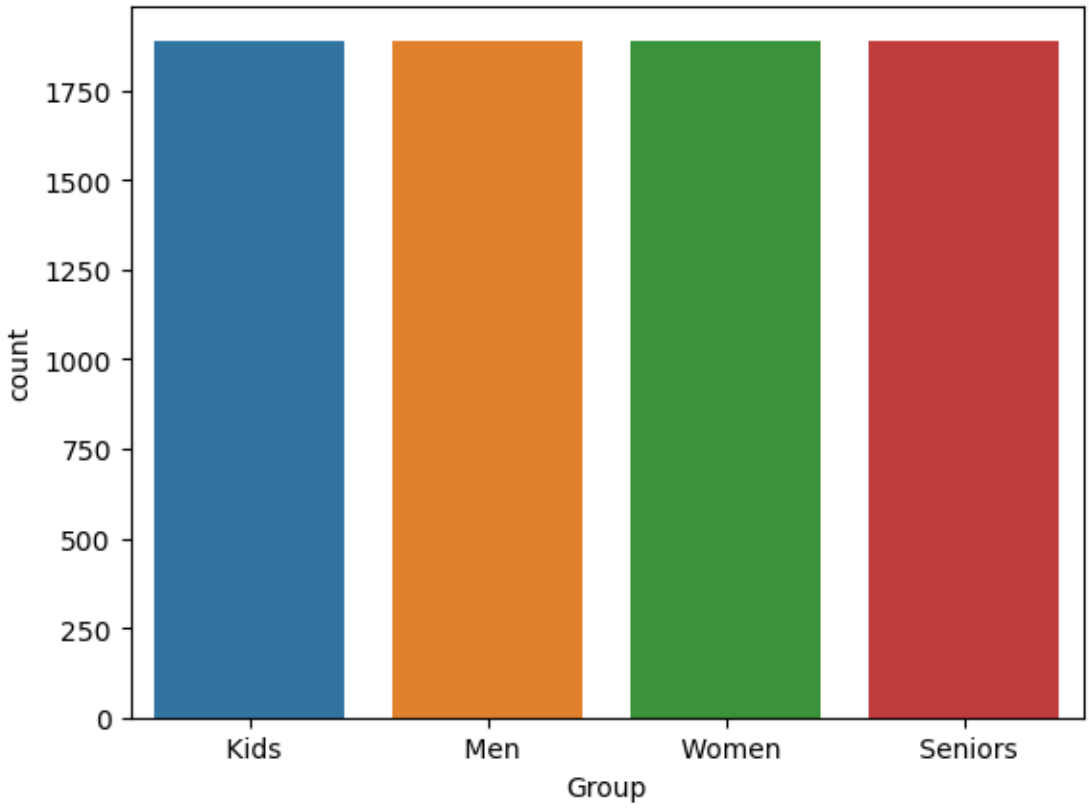
[10]:
```
<AxesSubplot:xlabel='State', ylabel='count'>
```



[16]:
```
sns.countplot(df['Group'])
```

```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

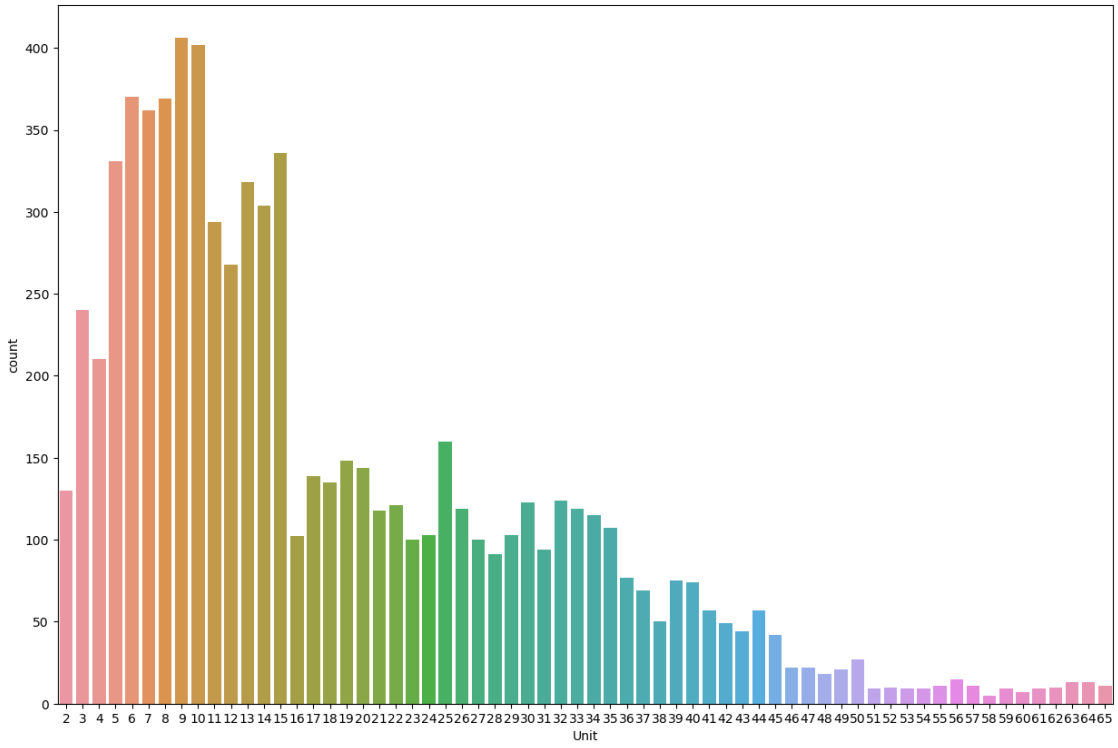
[16]:
```
<AxesSubplot:xlabel='Group', ylabel='count'>
```



[19]:
```
plt.figure(figsize=(15,10))
sns.countplot(df['Unit'])
```

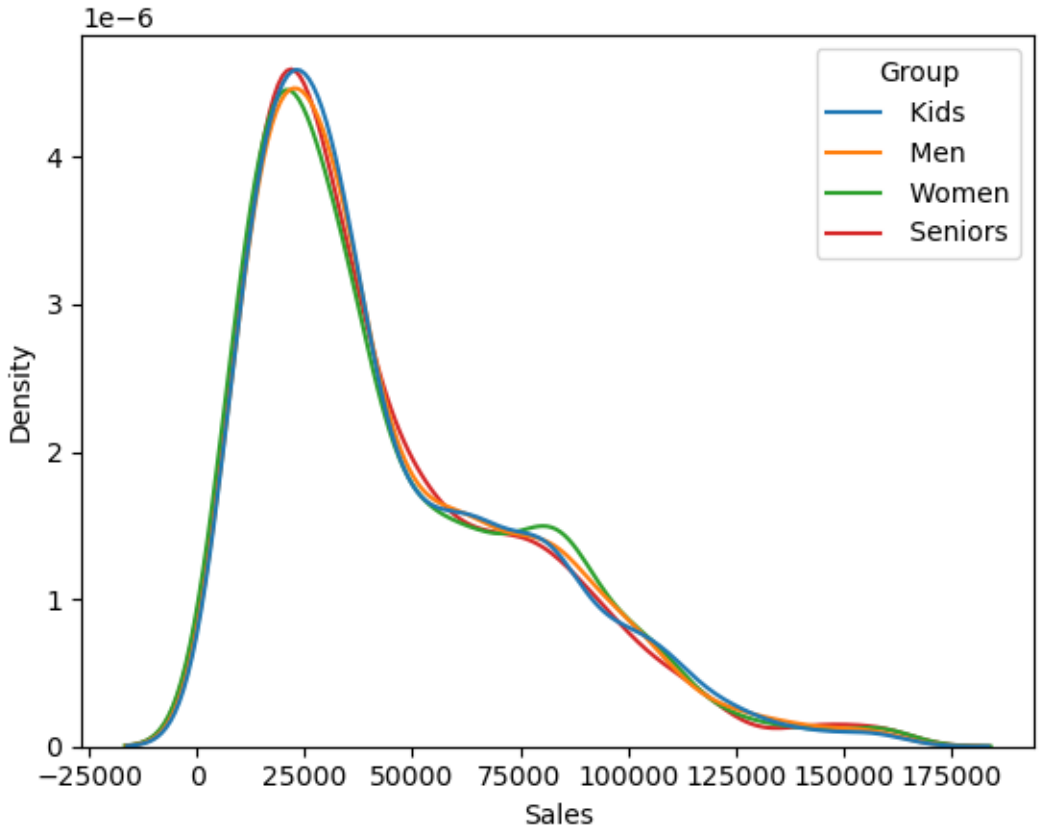
```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[19]:
```
<AxesSubplot:xlabel='Unit', ylabel='count'>
```



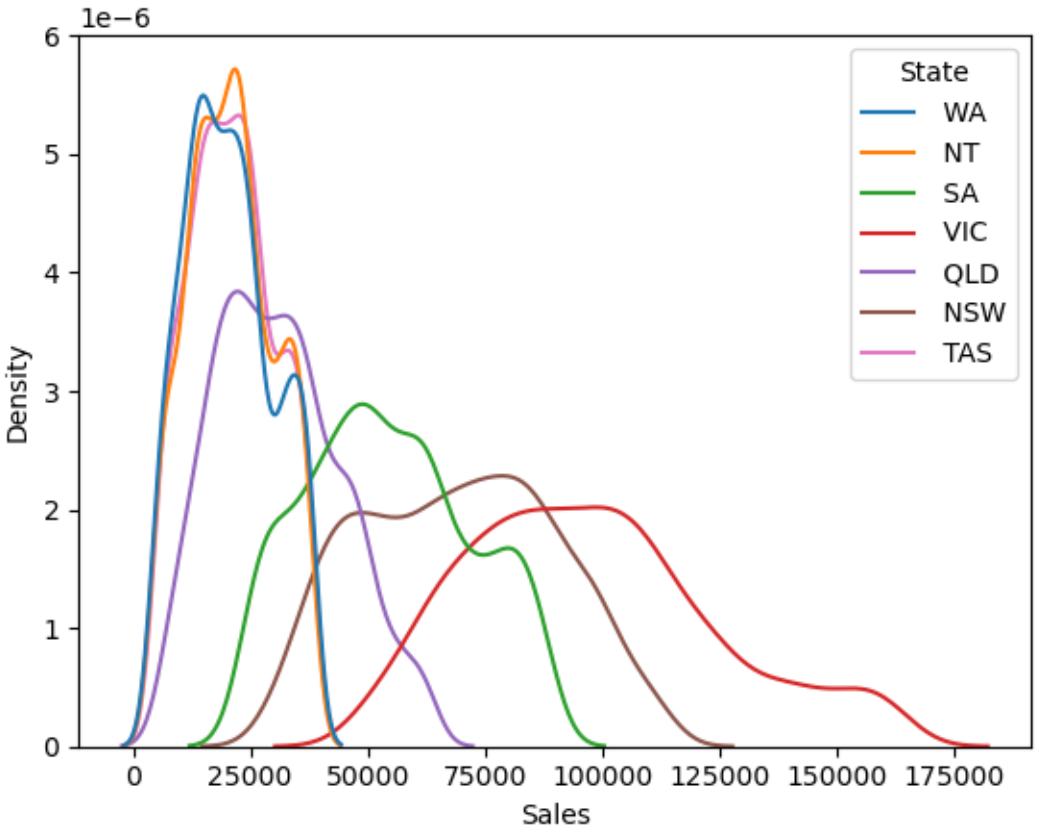
[22]:
```
sns.kdeplot(x=df['Sales'],hue=df['Group'])
```

[22]: <AxesSubplot:xlabel='Sales', ylabel='Density'>



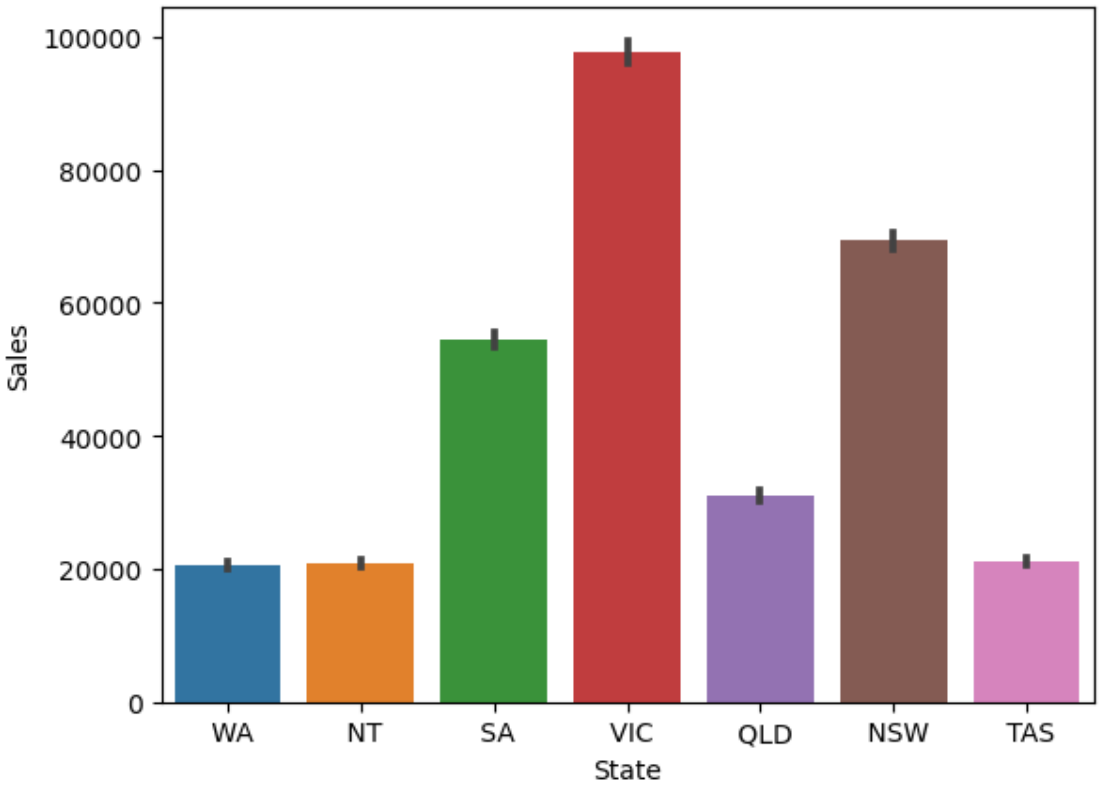
[25]:
```
sns.kdeplot(x=df['Sales'],hue=df['State'])
```

[25]:
```
<AxesSubplot:xlabel='Sales', ylabel='Density'>
```



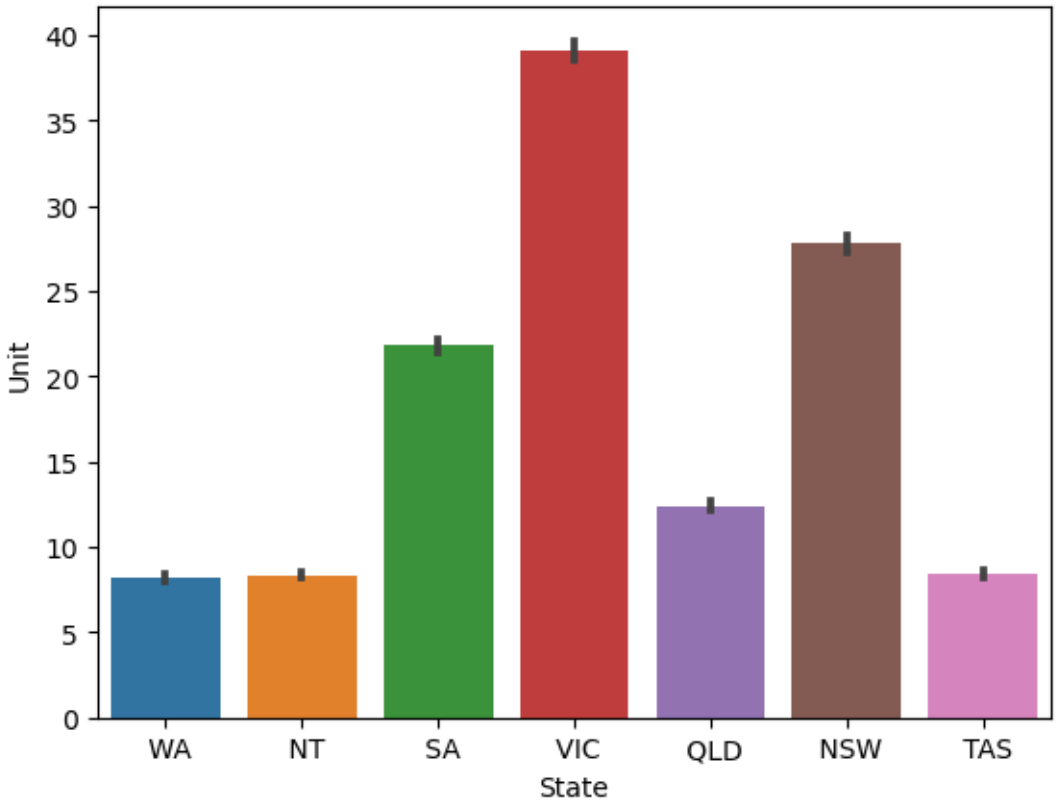
[33]:
```
sns.barplot(y=df['Sales'],x=df['State'])
```

[33]:
```
<AxesSubplot:xlabel='State', ylabel='Sales'>
```



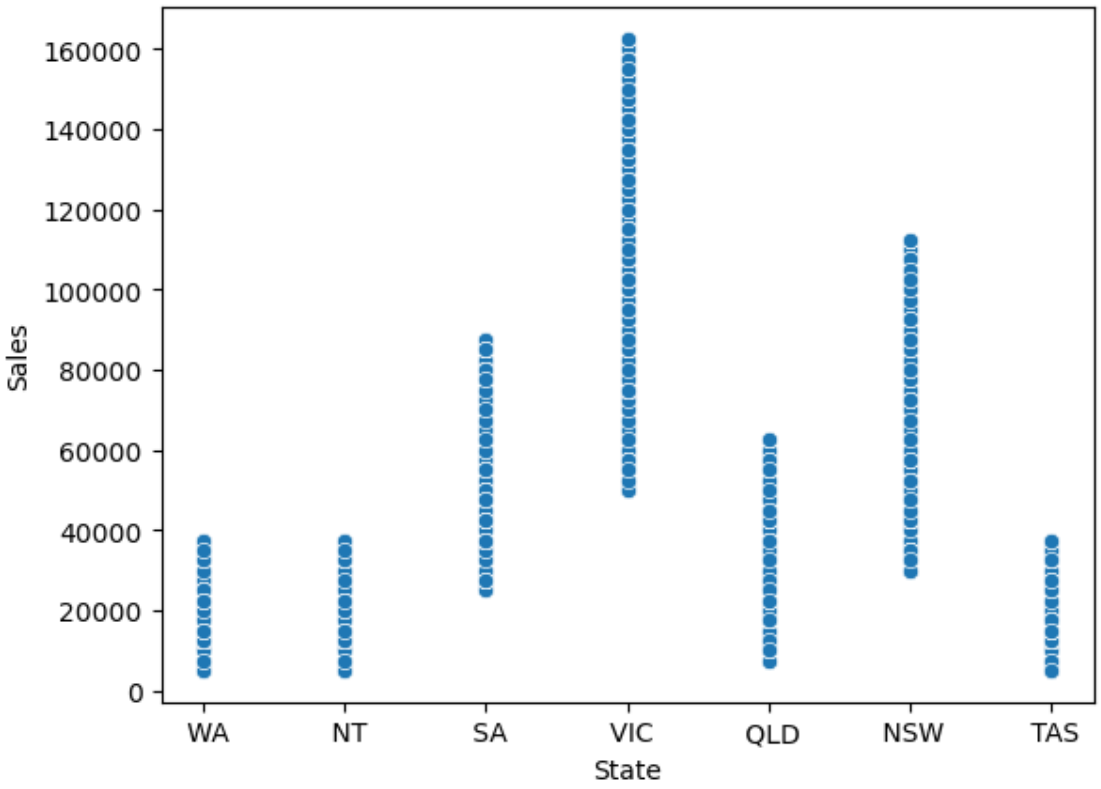
[131]:
```
sns.barplot(y=df['Unit'],x=df['State'])
```

[131]: <AxesSubplot:xlabel='State', ylabel='Unit'>



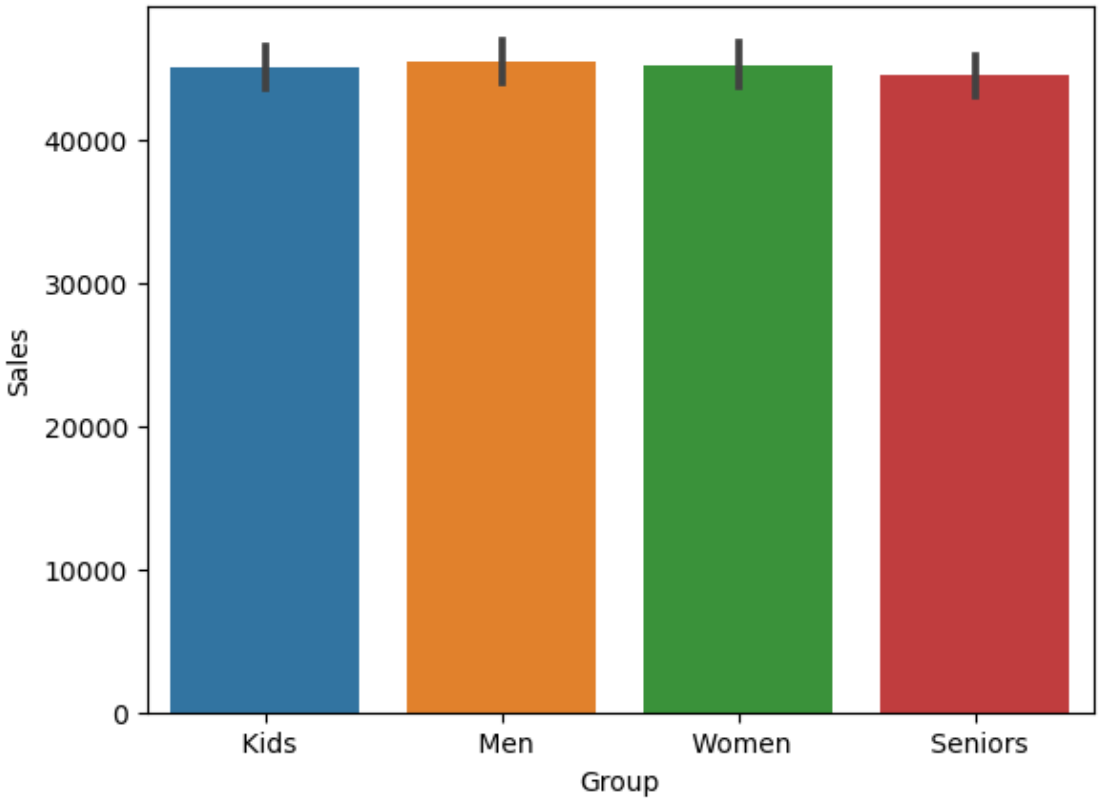
[23]:
```
sns.scatterplot(x=df['State'],y=df['Sales'])
```

[23]:
```
<AxesSubplot:xlabel='State', ylabel='Sales'>
```



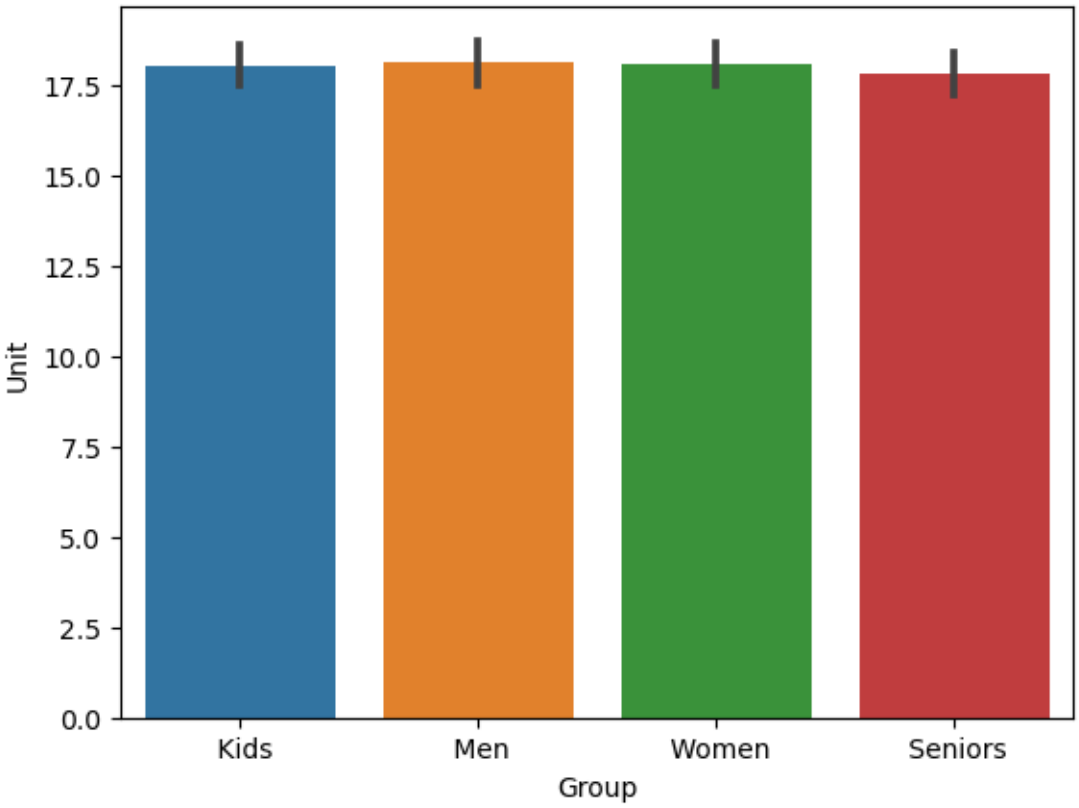
[34]:
```
sns.barplot(x=df['Group'],y=df['Sales'])
```

[34]:
```
<AxesSubplot:xlabel='Group', ylabel='Sales'>
```



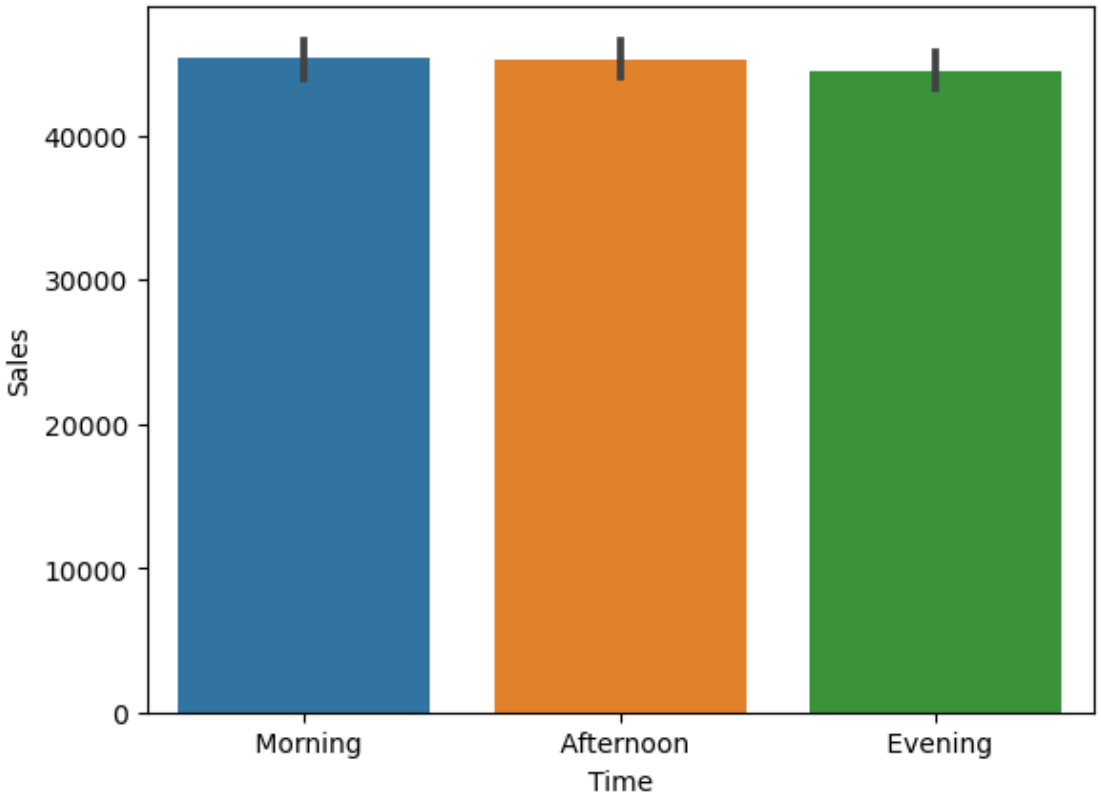
[132]:
```
sns.barplot(x=df['Group'],y=df['Unit'])
```

[132]:
```
<AxesSubplot:xlabel='Group', ylabel='Unit'>
```



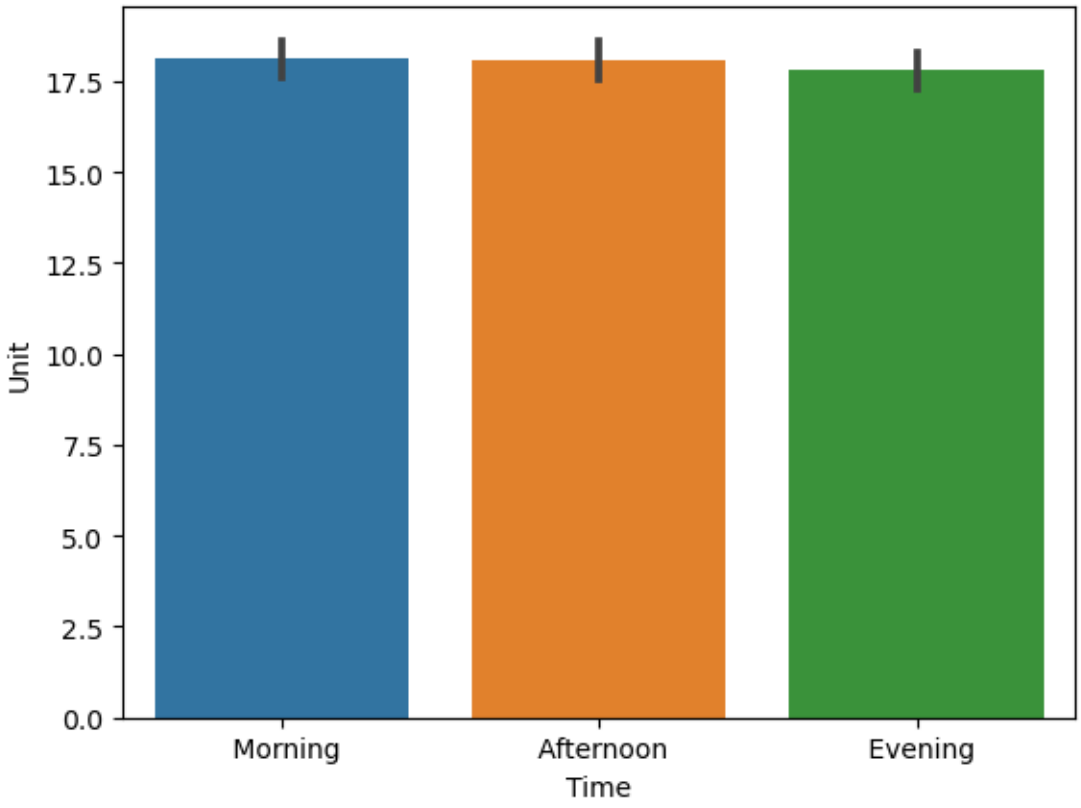
[39]:
```
sns.barplot(x=df['Time'],y=df['Sales'])
```

[39]: <AxesSubplot:xlabel='Time', ylabel='Sales'>



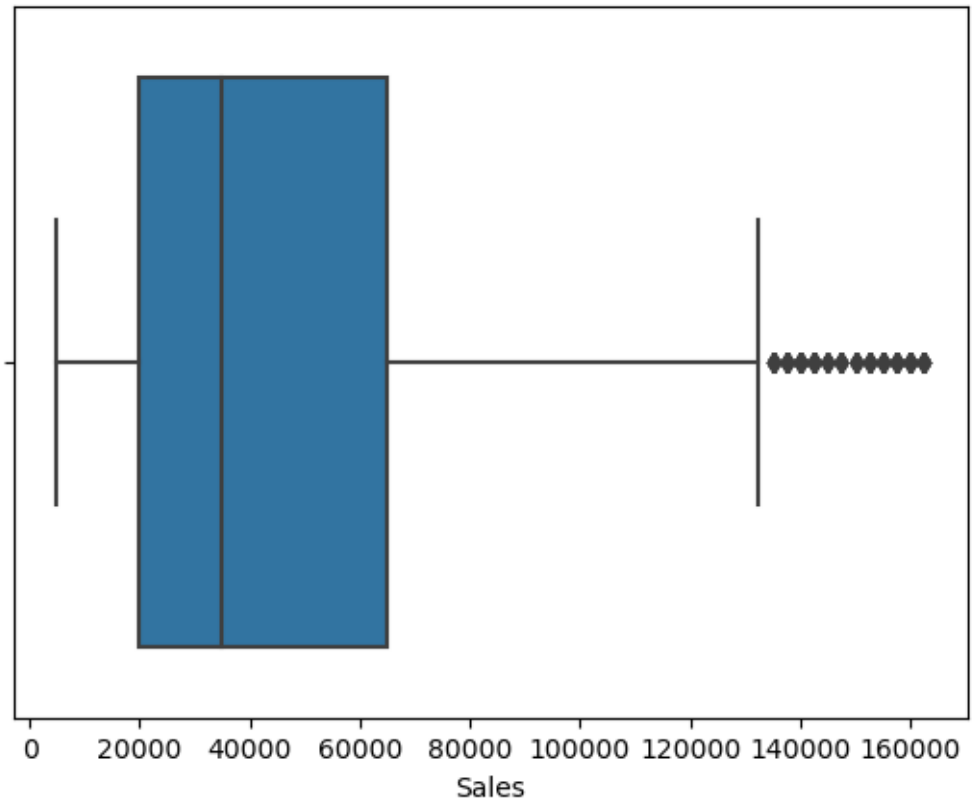
[45]:
```
sns.barplot(y=df['Unit'],x=df['Time'])
```

[45]:
```
<AxesSubplot:xlabel='Time', ylabel='Unit'>
```



[29]:
```
sns.boxplot(x=df['Sales'])
```

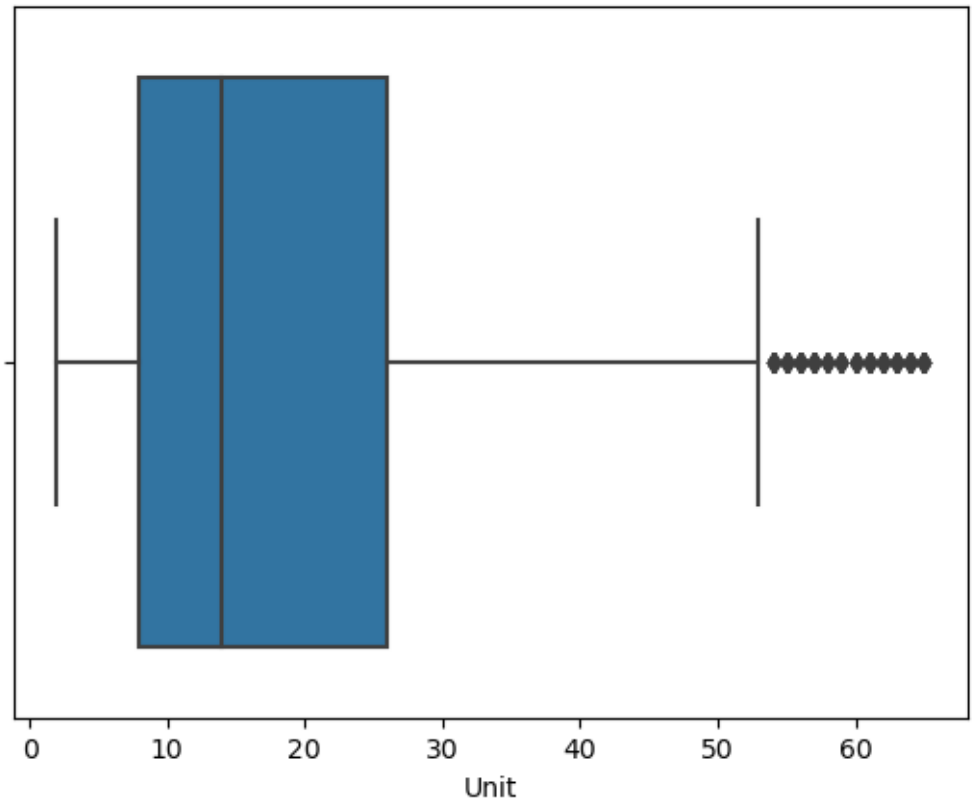
[29]: <AxesSubplot:xlabel='Sales'>



[30]:
```
sns.boxplot(df['Unit'])
```

```
C:\Users\Vinosh\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[30]:
```
<AxesSubplot:xlabel='Unit'>
```



[35]:
```
df_master = df.copy()
```

[36]:
```
df_master
```

[36]:

	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]

[59]:
```
df_master['Time'].value_counts()
```

[59]:
```
 Morning      2520
 Afternoon    2520
 Evening      2520
Name: Time, dtype: int64
```

[73]:
```
df_master['Time'] = df_master['Time'].replace(' Morning','0')
df_master['Time'] = df_master['Time'].replace(' Afternoon','1')
df_master['Time'] = df_master['Time'].replace(' Evening','2')
```

[74]:
```
df_master.dtypes
```

[74]:
```
Date     datetime64[ns]
Time             object
State            object
Group            object
Unit              int64
Sales             int64
dtype: object
```

[75]:
```
df_master['Time'] = df_master['Time'].astype(int)
```

[76]:
```
df_master['Time'].value_counts()
```

[76]:
```
0    2520
1    2520
2    2520
Name: Time, dtype: int64
```

[77]:
```
df_master['State'].value_counts()
```

[77]:
```
 WA     1080
 NT     1080
 SA     1080
 VIC    1080
 QLD    1080
 NSW    1080
 TAS    1080
Name: State, dtype: int64
```

[78]:
```
df_master['State'] = df_master['State'].replace(' WA','0')
df_master['State'] = df_master['State'].replace(' NT','1')
df_master['State'] = df_master['State'].replace(' SA','2')
df_master['State'] = df_master['State'].replace(' VIC','3')
df_master['State'] = df_master['State'].replace(' QLD','4')
df_master['State'] = df_master['State'].replace(' NSW','5')
df_master['State'] = df_master['State'].replace(' TAS','6')
```

[79]:
```
df_master.dtypes
```

[79]:
```
Date     datetime64[ns]
Time              int32
State            object
Group            object
Unit              int64
Sales             int64
dtype: object
```

[80]:
```
df_master['State'] = df_master['State'].astype(int)
```

[81]:
```
df_master['State'].value_counts()
```

[81]:
```
0    1080
1    1080
2    1080
3    1080
4    1080
5    1080
6    1080
Name: State, dtype: int64
```

[82]:
```
df_master.dtypes
```

[82]:
```
Date     datetime64[ns]
Time              int32
State             int32
Group            object
Unit              int64
Sales             int64
dtype: object
```

[83]:
```
df_master['Group'].value_counts()
```

[83]:
```
 Kids       1890
 Men        1890
 Women      1890
 Seniors    1890
Name: Group, dtype: int64
```

[84]:
```
df_master['Group'] = df_master['Group'].replace(' Kids','0')
df_master['Group'] = df_master['Group'].replace(' Men','1')
df_master['Group'] = df_master['Group'].replace(' Women','2')
df_master['Group'] = df_master['Group'].replace(' Seniors','3')
```

[85]:
```
df_master.dtypes
```

[85]:
```
Date     datetime64[ns]
Time              int32
```

```
State             int32
Group            object
Unit              int64
Sales             int64
dtype: object
```

[86]:
```
df_master['Group'] = df_master['Group'].astype(int)
```

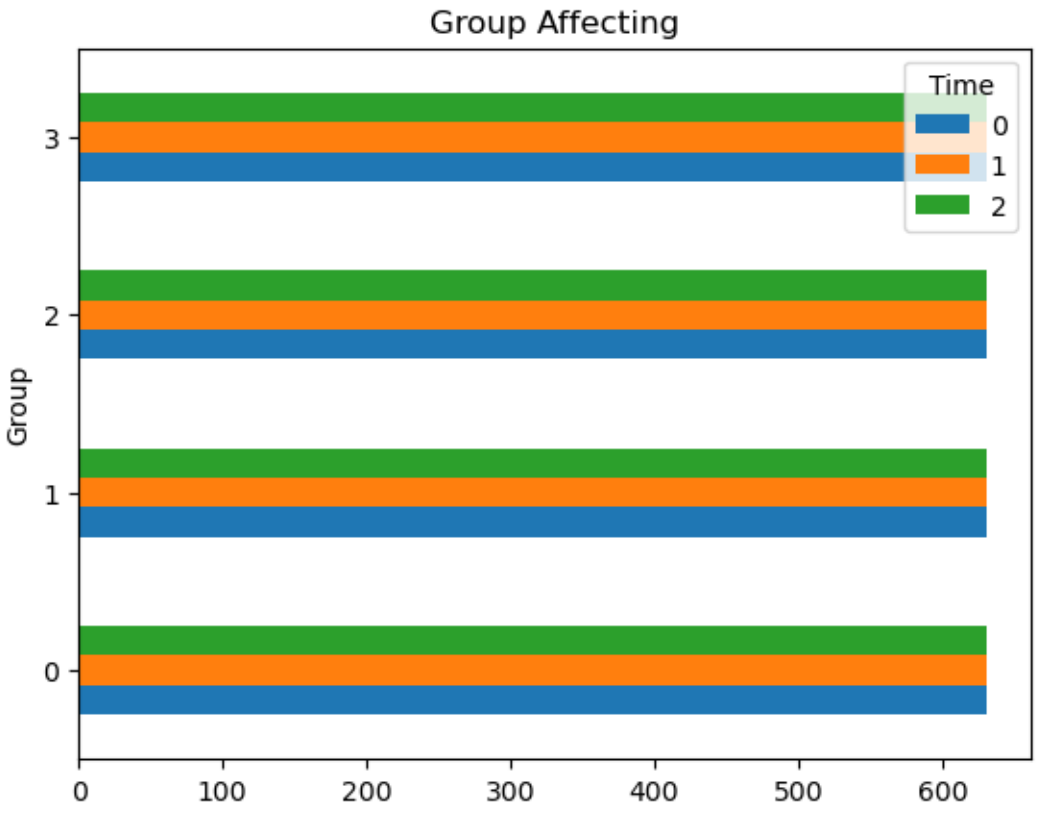
[87]:
```
df_master.dtypes
```

[87]:
```
Date     datetime64[ns]
Time              int32
State             int32
Group             int32
Unit              int64
Sales             int64
dtype: object
```

[88]:
```
df_master['Group'].value_counts()
```

[88]:
```
0    1890
1    1890
2    1890
3    1890
Name: Group, dtype: int64
```

[90]:
```
df_master.groupby(['Group','Time']).size().unstack().
  ↪plot(kind='barh',legend=True)
plt.title('Group Affecting')
plt.show()
```



[91]:
```
df_master.shape
```

[91]: (7560, 6)

[92]:
```
df_master.columns
```

[92]: Index(['Date', 'Time', 'State', 'Group', 'Unit', 'Sales'], dtype='object')

[93]:
```
X = df_master[['Time','State','Group','Unit']]
```

[94]:
```
y = df_master['Sales']
```

[95]:
```
X = df_master[['Time','State','Group','Unit']].values
```

[96]:
```
y = df_master['Sales'].values
```

[97]:
```
y
```

[97]: array([20000, 20000, 10000, …, 37500, 27500, 32500], dtype=int64)

[99]:
```
X
```

[99]:
```
array([[ 0,  0,  0,  8],
       [ 0,  0,  1,  8],
       [ 0,  0,  2,  4],
       …,
       [ 2,  6,  1, 15],
       [ 2,  6,  2, 11],
       [ 2,  6,  3, 13]], dtype=int64)
```

[100]:
```
from sklearn.model_selection import train_test_split
```

[101]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪20,random_state=42)
```

[102]:
```
X_train.shape
```

[102]:
```
(6048, 4)
```

[104]:
```
y_train.shape
```

[104]:
```
(6048,)
```

[105]:
```
X_test.shape
```

[105]:
```
(1512, 4)
```

[106]:
```
from sklearn.linear_model import LinearRegression
```

[113]:
```
lr = LinearRegression()
```

[114]:
```
lr.fit(X_train,y_train)
```

[114]:
```
LinearRegression()
```

[115]:
```
y_predict = lr.predict(X_test)
```

[116]:
```
y_predict
```

[116]:
```
array([30000., 25000., 37500., …, 62500., 50000., 15000.])
```

[117]:
```
y_test
```

[117]:
```
array([30000, 25000, 37500, …, 62500, 50000, 15000], dtype=int64)
```

[121]:
```
from sklearn.metrics import mean_squared_error
```

[122]:
```
mean_squared_error(y_test,y_predict)
```

[122]:
```
9.482848873403382e-22
```

[124]:
```
from sklearn.feature_selection import f_regression
```

[127]:
```
f_regression(X_train,y_train)
```

[127]:
```
(array([8.44909102e-01, 1.71685342e+02, 2.04179214e-01, 1.60169196e+18]),
 array([3.58033198e-01, 1.06330485e-38, 6.51384652e-01, 0.00000000e+00]))
```

[129]:
```
from sklearn import metrics
```

[130]:
```
metrics.r2_score(y_test,y_predict)
```

[130]:
```
1.0
```

1 Summary

1. The state VIC generates the highest Revenue
2. The state WA, NT and TAS generates the less revenue
3. VIC sold the highest number of Units
4. WA, NT and TAS sold the less number of Units
5. The newly opened branches are doing good and by the prediction its the right decisions to expand the company.

[ ]:
```

```