project-customer-analysis

March 1, 2024

#Understand the dataset

[1]:
```
#1.Data Collection
```

[2]:
```
##Import the data
```

[3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

[4]:
```
df=pd.read_excel('1688639662_ausapparalsales4thqrt2020.xlsx')
```

[5]:
```
df
```

[5]:
```
           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]
```

[6]:
```
#2.Data Inspection
```

[7]:
```
df.columns
```

[7]:
```
Index(['Date', 'Time', 'State', 'Group', 'Unit', 'Sales'], dtype='object')
```

[8]:
```
len(df.columns)
```

[8]: 6

[9]:

```
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
```

[10]:

```
df.describe()
```

[10]:

```
              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
```

[11]:

```
df.head()
```

[11]:

```
        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
```

[12]:

```
df.tail()
```

[12]:

```
           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
```

[13]:

```
#3.Data Wrangling
```

[14]:

```
# 3.1Checking for missing values
missing_values = df.isnull().sum()
print("Missing Values per Column:")
print(missing_values)
```

```
Missing Values per Column:
Date     0
Time     0
State    0
Group    0
Unit     0
Sales    0
dtype: int64
```

[15]:

```
#3.2 Removing duplicate records
df_no_duplicates = df.drop_duplicates()
```

[16]:

```
df_no_duplicates
```

[16]:

```
           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]
```

1.No Null values in the data observed 2.No duplictes in the data.

[17]:

```
#4.Data Cleaning
```

[18]:

```
# Cleaning data by standardizing formats
df['Date'] = pd.to_datetime(df['Date'])
# Displaying the DataFrame after cleaning
print("DataFrame after cleaning data by standardizing formats:")
print(df)
```

```
DataFrame after cleaning data by standardizing formats:
           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]
```

[19]:

```
#6.Data Transformation
```

[20]:

```
# Normalize 'Sales' column and create a new feature 'Normalized_Sales'
df['Normalized_Sales'] = (df['Sales'] - df['Sales'].min()) / (df['Sales'].max()␣
 ↪- df['Sales'].min())
print(df)
```

```
           Date        Time State     Group  Unit  Sales  Normalized_Sales
0    2020-10-01     Morning    WA      Kids     8  20000          0.095238
1    2020-10-01     Morning    WA       Men     8  20000          0.095238
2    2020-10-01     Morning    WA     Women     4  10000          0.031746
3    2020-10-01     Morning    WA   Seniors    15  37500          0.206349
4    2020-10-01   Afternoon    WA      Kids     3   7500          0.015873
…         …         …   …       …   …    …               …
7555 2020-12-30   Afternoon   TAS   Seniors    14  35000          0.190476
7556 2020-12-30     Evening   TAS      Kids    15  37500          0.206349
7557 2020-12-30     Evening   TAS       Men    15  37500          0.206349
7558 2020-12-30     Evening   TAS     Women    11  27500          0.142857
7559 2020-12-30     Evening   TAS   Seniors    13  32500          0.174603

[7560 rows x 7 columns]
```

[21]:

```
numerical_feature_columns = list(df._get_numeric_data().columns)
numerical_feature_columns
```

[21]:

```
['Unit', 'Sales', 'Normalized_Sales']
```

[22]:

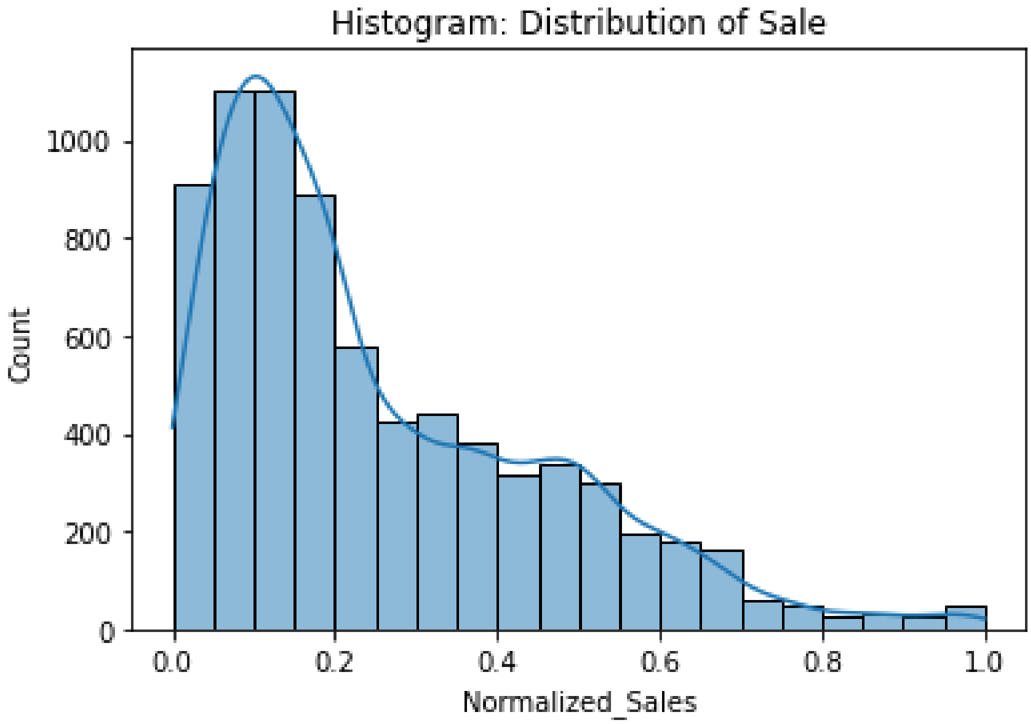
```
categorical_feature_columns = list(set(df.columns) - set(df._get_numeric_data().
 ↪columns))
categorical_feature_columns
```

[22]:

```
['Date', 'Time', 'State', 'Group']
```

[23]:

```
sns.histplot(df['Normalized_Sales'], bins=20, kde=True)
plt.title('Histogram: Distribution of Sale')
plt.show()
```

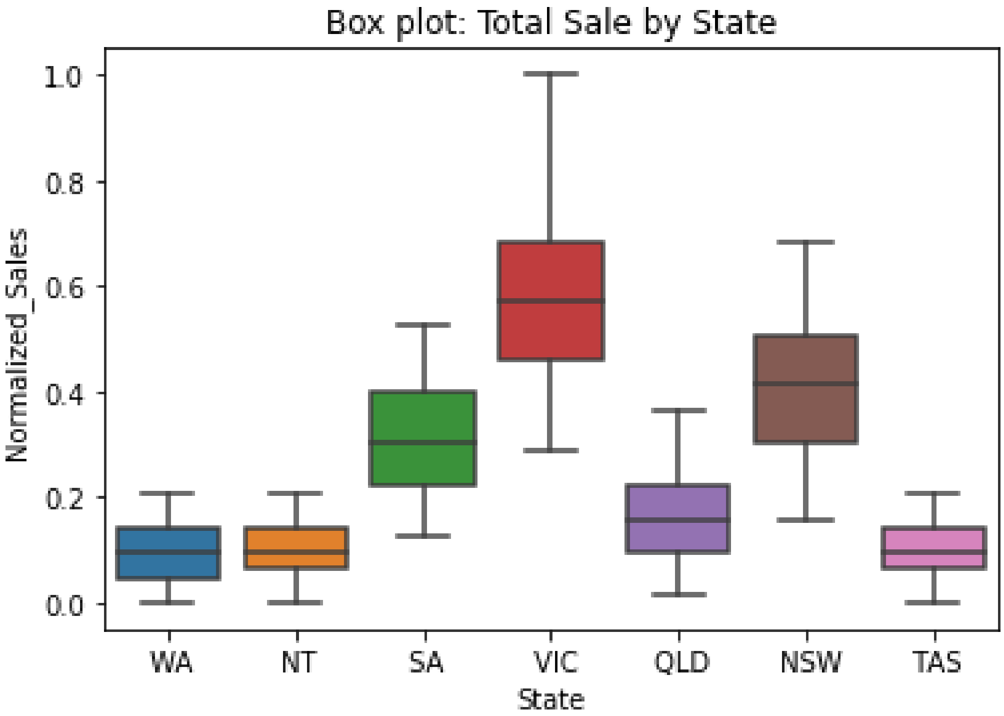


[24]:

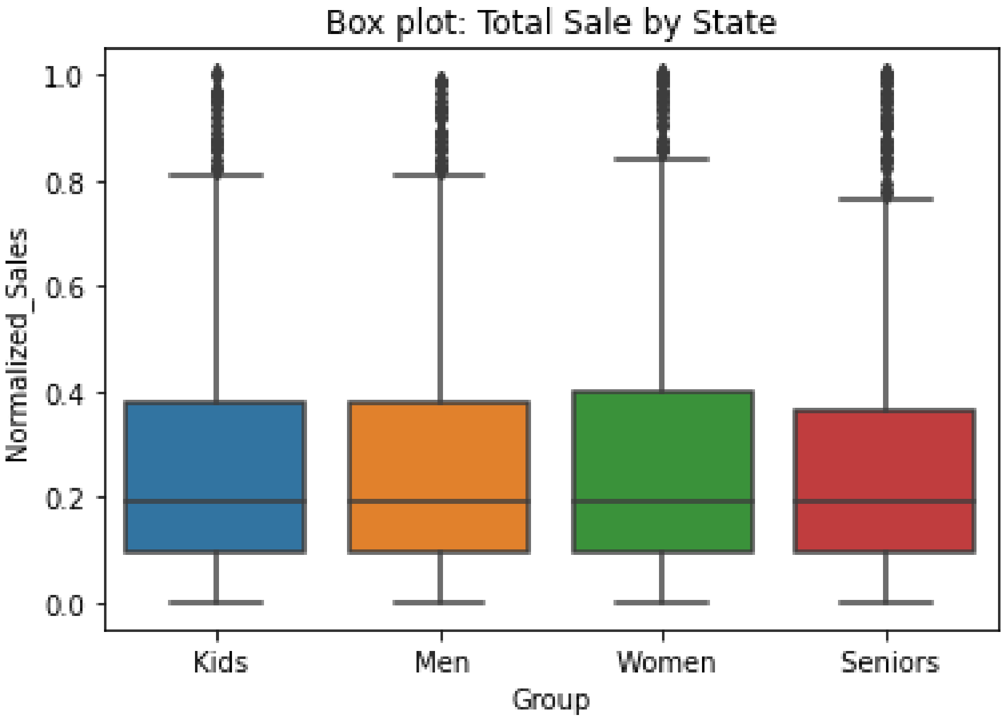
```
#from above KDE plot we can observed that graph is left skwed it means very␣
 ↪less time the sale is below average(mean<median)
```

[25]:

```
sns.boxplot(x='State', y='Normalized_Sales', data=df)
plt.title('Box plot: Total Sale by State')
plt.show()
```

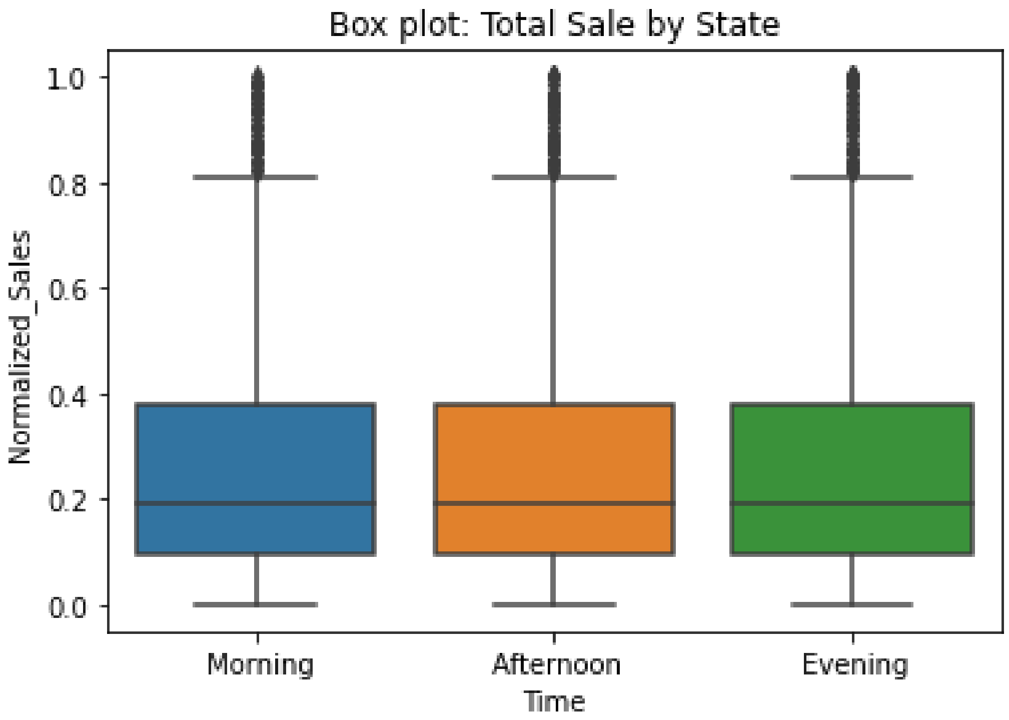


[26]:
```
sns.boxplot(x='Group', y='Normalized_Sales', data=df)
plt.title('Box plot: Total Sale by State')
plt.show()
```



[27]:

```
sns.boxplot(x='Time', y='Normalized_Sales', data=df)
plt.title('Box plot: Total Sale by State')
plt.show()
```



[28]:

```
#7.Data Binning
```

[29]:

```
if 'Sales' in df.columns:
    bin_edges = [0, 100, 200, 300, 400, 500, np.inf]
    bin_labels = ['0-100', '101-200', '201-300', '301-400', '401-500', '501+']
    df['Sales_Category'] = pd.cut(df['Sales'], bins=bin_edges,␣
  ↪labels=bin_labels, right=False)
    print("DataFrame with Sales_Category column:")
    print(df)
else:
    print("The 'Sales' column does not exist in the DataFrame.")
```

```
DataFrame with Sales_Category column:
           Date        Time State     Group  Unit  Sales  Normalized_Sales  \
0    2020-10-01     Morning    WA      Kids     8  20000          0.095238
1    2020-10-01     Morning    WA       Men     8  20000          0.095238
2    2020-10-01     Morning    WA     Women     4  10000          0.031746
3    2020-10-01     Morning    WA   Seniors    15  37500          0.206349
4    2020-10-01   Afternoon    WA      Kids     3   7500          0.015873
…         …         …   …       …   …    …               …
7555 2020-12-30   Afternoon   TAS   Seniors    14  35000          0.190476
7556 2020-12-30     Evening   TAS      Kids    15  37500          0.206349
7557 2020-12-30     Evening   TAS       Men    15  37500          0.206349
7558 2020-12-30     Evening   TAS     Women    11  27500          0.142857
```

```
7559 2020-12-30     Evening   TAS   Seniors    13  32500          0.174603

     Sales_Category
0              501+
1              501+
2              501+
3              501+
4              501+
…             …
7555           501+
7556           501+
7557           501+
7558           501+
7559           501+

[7560 rows x 8 columns]
```

[30]:

```
#8.Handling Outliers
```

[31]:

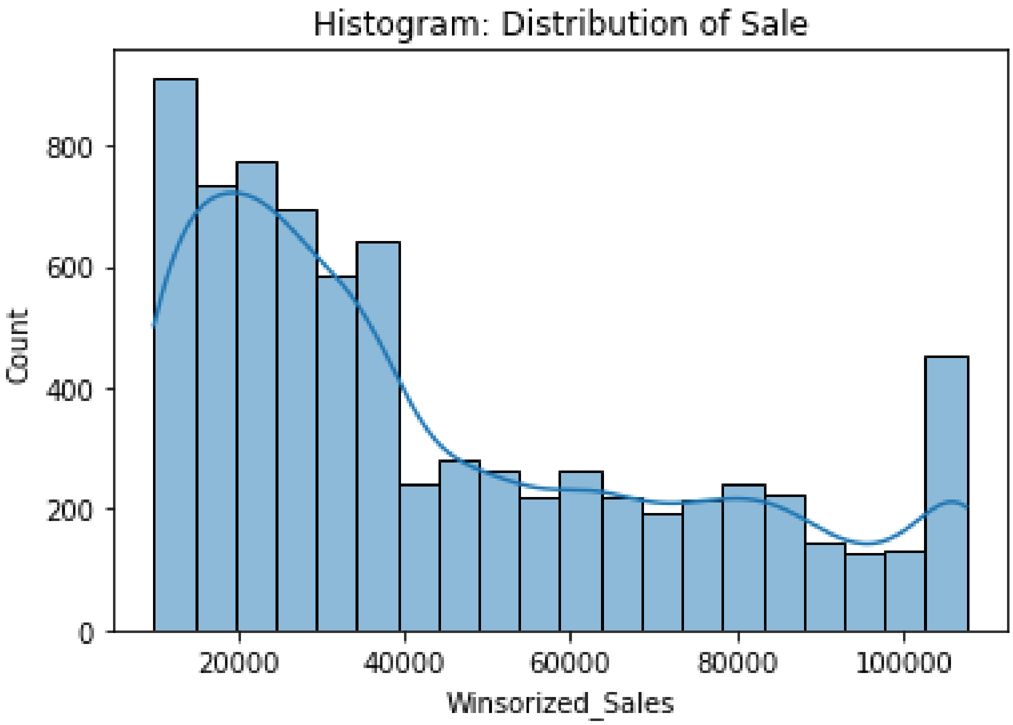
```
# Handling outliers by winsorizing
from scipy.stats.mstats import winsorize

# Check if 'Sale' column exists in the DataFrame
if 'Sales' in df.columns:
    # Winsorizing the 'Sales' column with limits [0.05, 0.05]
    df['Winsorized_Sales'] = winsorize(df['Sales'], limits=[0.05, 0.05])

    # Displaying the DataFrame with the winsorized column
    print("DataFrame with winsorized column:")
    print(df)
else:
    print("The 'Sales' column does not exist in the DataFrame.")
```

```
DataFrame with winsorized column:
           Date        Time State     Group  Unit  Sales  Normalized_Sales  \
0     2020-10-01    Morning    WA      Kids     8  20000          0.095238
1     2020-10-01    Morning    WA       Men     8  20000          0.095238
2     2020-10-01    Morning    WA     Women     4  10000          0.031746
3     2020-10-01    Morning    WA   Seniors    15  37500          0.206349
4     2020-10-01  Afternoon    WA      Kids     3   7500          0.015873
…         …         …   …       …   …    …               …
7555 2020-12-30  Afternoon   TAS   Seniors    14  35000          0.190476
7556 2020-12-30    Evening   TAS      Kids    15  37500          0.206349
7557 2020-12-30    Evening   TAS       Men    15  37500          0.206349
7558 2020-12-30    Evening   TAS     Women    11  27500          0.142857
7559 2020-12-30    Evening   TAS   Seniors    13  32500          0.174603

     Sales_Category  Winsorized_Sales
```

```
0               501+             20000
1               501+             20000
2               501+             10000
3               501+             37500
4               501+             10000
…              …                …
7555            501+             35000
7556            501+             37500
7557            501+             37500
7558            501+             27500
7559            501+             32500

[7560 rows x 9 columns]
```

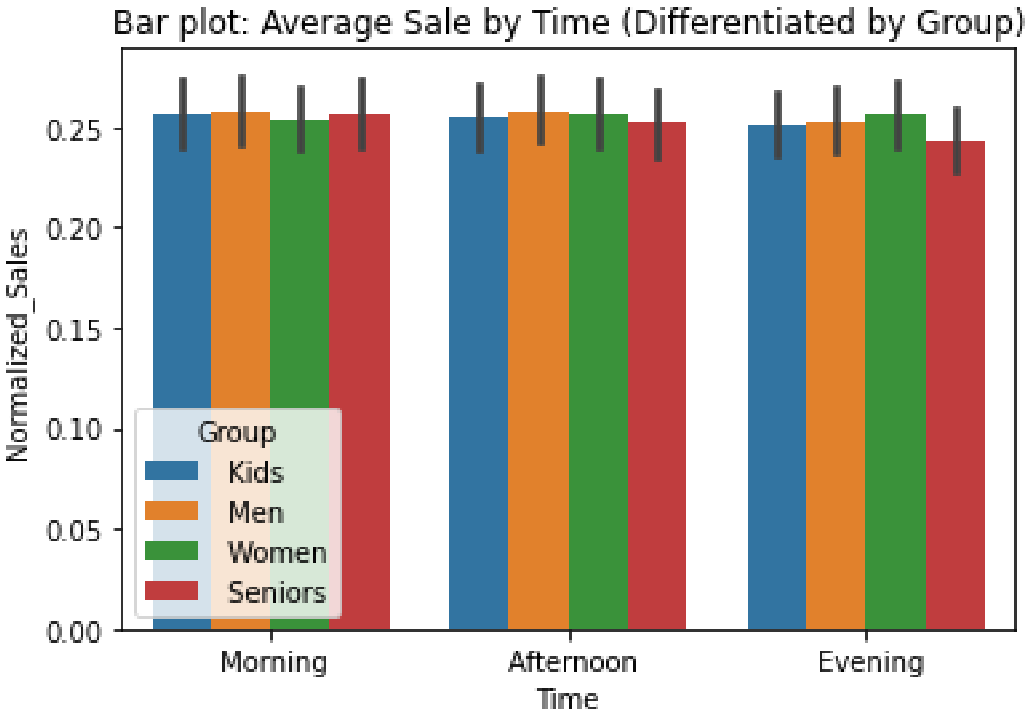
[32]:

```python
sns.histplot(df['Winsorized_Sales'], bins=20, kde=True)
plt.title('Histogram: Distribution of Sale')
plt.show()
```



[33]:

```python
sns.barplot(x='Time', y='Normalized_Sales', data=df, hue='Group')
plt.title('Bar plot: Average Sale by Time (Differentiated by Group)')
plt.show()
```



```
[50]: #1.Morning time - all section sale is almost same, women section is having␣
        ↪lowest sale as compared to others.
      #2.Afternoon time - senior section sale is lowest
      #3.Evening time - Senior section sale is lowest
```

```
[34]: df.State
```

```
[34]: 0          WA
      1          WA
      2          WA
      3          WA
      4          WA
               …
      7555      TAS
      7556      TAS
      7557      TAS
      7558      TAS
      7559      TAS
      Name: State, Length: 7560, dtype: object
```

```
[35]: State=df.State.unique()
      len(State)
```

```
[35]: 7
```

```
[36]: State_by_sales=df.State.value_counts()
      State_by_sales
```

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

```
[37]: # From above it is observed Statewise sale is equal for overall groups.
```

```
[38]: state_wise_Group_sales=df.Group.value_counts()
      state_wise_Group_sales
```

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

```
[39]: statewise_sales=df.groupby(['State','Group'])['Sales','Unit'].max()
      statewise_sales
```

```
/tmp/ipykernel_145/2493984019.py:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  statewise_sales=df.groupby(['State','Group'])['Sales','Unit'].max()
```

```
[39]:                  Sales  Unit
      State Group
       NSW   Kids     112500    45
             Men      112500    45
             Seniors  112500    45
             Women    112500    45
       NT    Kids      37500    15
             Men       37500    15
             Seniors   37500    15
             Women     37500    15
       QLD   Kids      62500    25
             Men       62500    25
             Seniors   62500    25
             Women     62500    25
```

```
SA    Kids      87500   35
      Men       87500   35
      Seniors   87500   35
      Women     87500   35
TAS   Kids      37500   15
      Men       37500   15
      Seniors   37500   15
      Women     37500   15
VIC   Kids     162500   65
      Men      160000   64
      Seniors  162500   65
      Women    162500   65
WA    Kids      37500   15
      Men       37500   15
      Seniors   37500   15
      Women     37500   15
```

[51]:

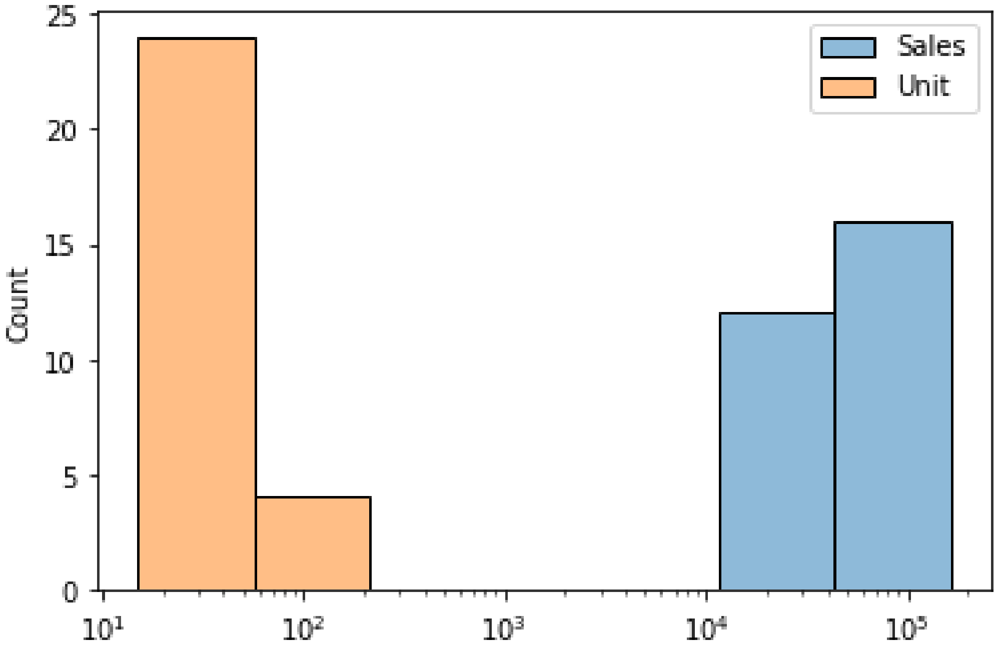
```
#largest selling unit is of Kids section and it is from NSW state
# Almost Every type of unit is same from each type except VIC state.
#Every state has largest selling unit is of Kids and smallest selling unit is␣
 ↪of women section
# overall smallest sale is of WA state
```

[40]:

```
sns.histplot(statewise_sales, log_scale=True)
```

[40]:

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



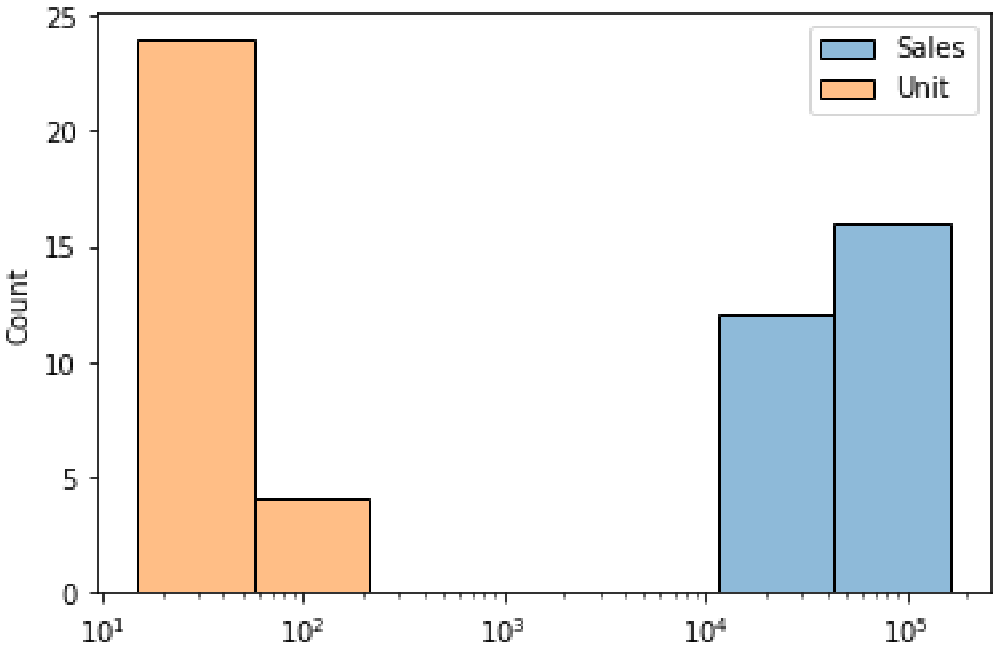
[52]:
```
groupwise_sales=df.groupby(['Group','State'])['Sales','Unit'].max()
groupwise_sales
```

```
/tmp/ipykernel_145/1761288186.py:1: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  groupwise_sales=df.groupby(['Group','State'])['Sales','Unit'].max()
```

[52]:
```
                 Sales  Unit
Group    State
 Kids     NSW   112500    45
          NT     37500    15
          QLD    62500    25
          SA     87500    35
          TAS    37500    15
          VIC   162500    65
          WA     37500    15
 Men      NSW   112500    45
          NT     37500    15
          QLD    62500    25
          SA     87500    35
          TAS    37500    15
          VIC   160000    64
          WA     37500    15
 Seniors  NSW   112500    45
          NT     37500    15
          QLD    62500    25
          SA     87500    35
          TAS    37500    15
          VIC   162500    65
          WA     37500    15
 Women    NSW   112500    45
          NT     37500    15
          QLD    62500    25
          SA     87500    35
          TAS    37500    15
          VIC   162500    65
          WA     37500    15
```

[53]:
```
sns.histplot(groupwise_sales, log_scale=True)
```

[53]:
```
<AxesSubplot: ylabel='Count'>
```



[54]:

```
df.columns
```

[54]:

```
Index(['Date', 'Time', 'State', 'Group', 'Unit', 'Sales', 'Normalized_Sales',
       'Sales_Category', 'Winsorized_Sales'],
      dtype='object')
```

[55]:

```
df.Date
```

[55]:

```
0       2020-10-01
1       2020-10-01
2       2020-10-01
3       2020-10-01
4       2020-10-01
           …
7555    2020-12-30
7556    2020-12-30
7557    2020-12-30
7558    2020-12-30
7559    2020-12-30
Name: Date, Length: 7560, dtype: datetime64[ns]
```

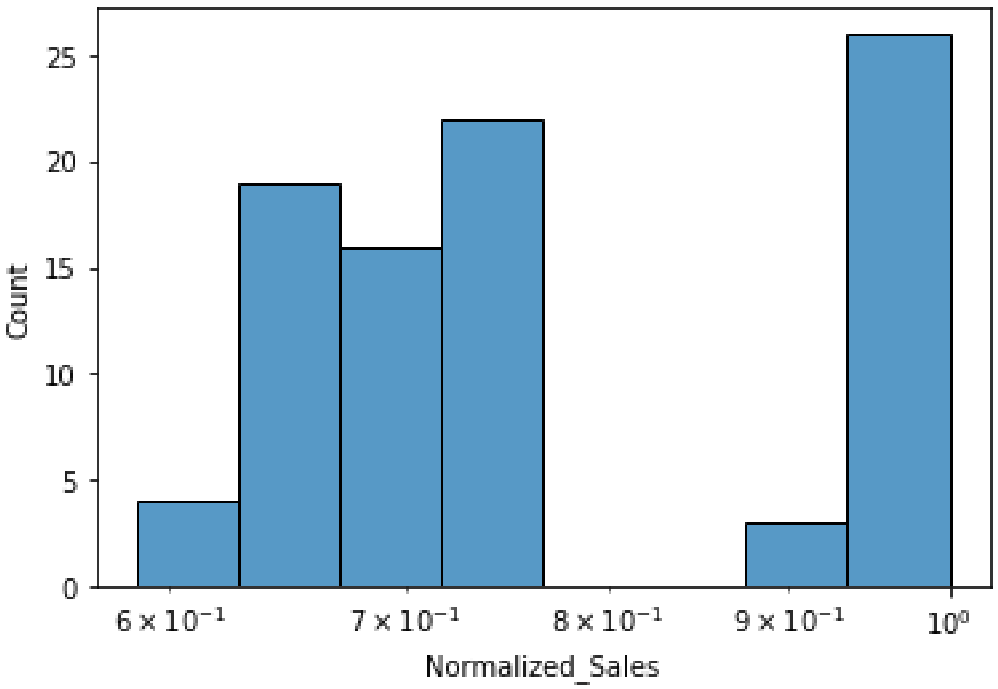
[46]:

```
datewise_sales=df.groupby(['Date'])['Normalized_Sales'].max()
datewise_sales
```

```
[46]: Date
      2020-10-01    0.761905
      2020-10-02    0.730159
      2020-10-03    0.761905
      2020-10-04    0.698413
      2020-10-05    0.666667
                      …
      2020-12-26    0.984127
      2020-12-27    0.952381
      2020-12-28    1.000000
      2020-12-29    1.000000
      2020-12-30    1.000000
      Name: Normalized_Sales, Length: 90, dtype: float64
```

```
[47]: sns.histplot(datewise_sales, log_scale=True)
```

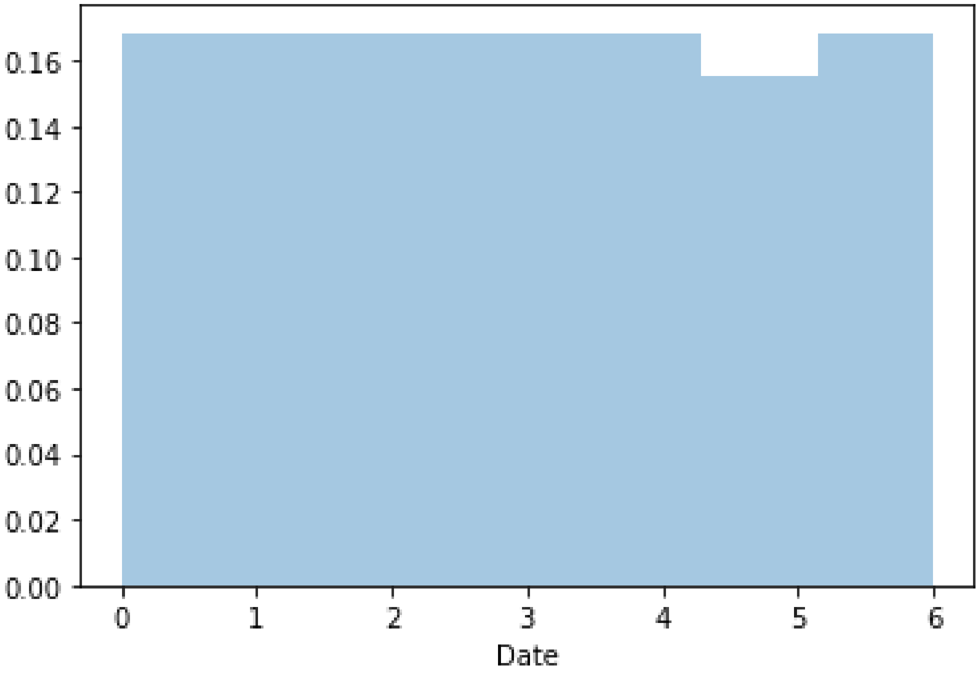
```
[47]: <AxesSubplot: xlabel='Normalized_Sales', ylabel='Count'>
```



```
[48]: sns.distplot(df.Date.dt.dayofweek, bins=7, kde=False, norm_hist=True )
```

```
/tmp/ipykernel_145/4160792428.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Date.dt.dayofweek, bins=7, kde=False, norm_hist=True )
```

[48]:
```
<AxesSubplot: xlabel='Date'>
```



[57]:
```
#All of days in week sale is equal except 5th day in week.
```

[ ]: