Download the necessary libraries

In [500…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
```

Load the Dataset

In [501…

```
df = pd.read_csv("C:\\Users\\captr\\OneDrive\\Desktop\\AusApparalSales4thQrt2020.cs
df.head(5)
```

Out[501]:

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

In [502…

```
type(df['Date'])
```

Out[502]:

```
pandas.core.series.Series
```

Checking for the missing values

In [503…

```
df.isnull().sum()
```

Out[503]:

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

Checking for the duplicate values (if any)

In [504…

```
df.duplicated().sum()
```

Out[504]:

```
0
```

Running Basic Summary Statistics of the data

In [505…

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

Out[505]:

	count	mean	std	min	25%	50%	75%	max
Unit	7560.0	18.005423	12.901403	2.0	8.0	14.0	26.0	65.0
Sales	7560.0	45013.558201	32253.506944	5000.0	20000.0	35000.0	65000.0	162500.0

Checking how many times different states of Australia appeared in our dataset

In [506…

```
state_counts=df['State'].value_counts()
state_counts
```

Out[506]:

```
State
 WA     1080
 NT     1080
 SA     1080
 VIC    1080
 QLD    1080
 NSW    1080
 TAS    1080
Name: count, dtype: int64
```

In [507…

```
state_counts.sort_values()
```

Out[507]:

```
State
 WA     1080
 NT     1080
 SA     1080
 VIC    1080
 QLD    1080
 NSW    1080
 TAS    1080
Name: count, dtype: int64
```

Two of our features i.e Unit and Sales were quite wide apart in terms of their range hence performed min-max-scaling to transform it to a fixed range between 0 to 1 or [0,1]

In [508…

```
from sklearn.preprocessing import MinMaxScaler
```

In [509…

```
scaler= MinMaxScaler()
scaler.fit(df[['Sales']])
df['Sales']= scaler.transform(df[['Sales']])
scaler.fit(df[['Unit']])
df['Unit']= scaler.transform(df[['Unit']])
df.head()
```

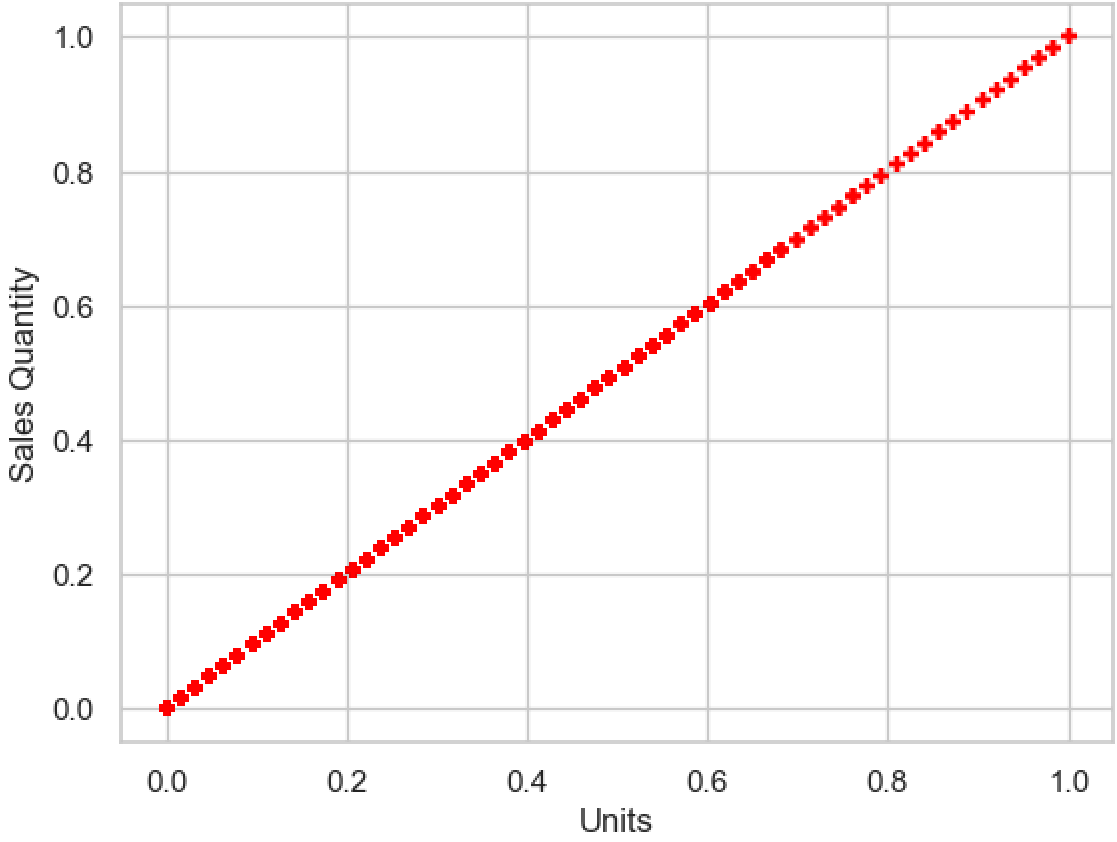
Out[509]:

	Date	Time	State	Group	Unit	Sales
0	1-Oct-2020	Morning	WA	Kids	0.095238	0.095238
1	1-Oct-2020	Morning	WA	Men	0.095238	0.095238
2	1-Oct-2020	Morning	WA	Women	0.031746	0.031746
3	1-Oct-2020	Morning	WA	Seniors	0.206349	0.206349
4	1-Oct-2020	Afternoon	WA	Kids	0.015873	0.015873

Plotted Unit against time

In [510…

```
%matplotlib inline
plt.scatter(df['Unit'],df['Sales'], color='red', marker='+')
plt.xlabel('Units')
plt.ylabel('Sales Quantity')
plt.show()
```



Grouped the data according to the State/region

In [511…

```
g=df.groupby('State')
g.head(5)
```

Out[511]:

	Date	Time	State	Group	Unit	Sales
0	1-Oct-2020	Morning	WA	Kids	0.095238	0.095238
1	1-Oct-2020	Morning	WA	Men	0.095238	0.095238
2	1-Oct-2020	Morning	WA	Women	0.031746	0.031746
3	1-Oct-2020	Morning	WA	Seniors	0.206349	0.206349
4	1-Oct-2020	Afternoon	WA	Kids	0.015873	0.015873
12	1-Oct-2020	Morning	NT	Kids	0.174603	0.174603
13	1-Oct-2020	Morning	NT	Men	0.047619	0.047619
14	1-Oct-2020	Morning	NT	Women	0.031746	0.031746
15	1-Oct-2020	Morning	NT	Seniors	0.126984	0.126984
16	1-Oct-2020	Afternoon	NT	Kids	0.174603	0.174603
24	1-Oct-2020	Morning	SA	Kids	0.158730	0.158730
25	1-Oct-2020	Morning	SA	Men	0.158730	0.158730
26	1-Oct-2020	Morning	SA	Women	0.222222	0.222222
27	1-Oct-2020	Morning	SA	Seniors	0.412698	0.412698
28	1-Oct-2020	Afternoon	SA	Kids	0.222222	0.222222
36	1-Oct-2020	Morning	VIC	Kids	0.746032	0.746032
37	1-Oct-2020	Morning	VIC	Men	0.539683	0.539683
38	1-Oct-2020	Morning	VIC	Women	0.507937	0.507937
39	1-Oct-2020	Morning	VIC	Seniors	0.380952	0.380952
40	1-Oct-2020	Afternoon	VIC	Kids	0.460317	0.460317
48	1-Oct-2020	Morning	QLD	Kids	0.285714	0.285714
49	1-Oct-2020	Morning	QLD	Men	0.253968	0.253968
50	1-Oct-2020	Morning	QLD	Women	0.206349	0.206349
51	1-Oct-2020	Morning	QLD	Seniors	0.190476	0.190476
52	1-Oct-2020	Afternoon	QLD	Kids	0.253968	0.253968
60	1-Oct-2020	Morning	NSW	Kids	0.587302	0.587302
61	1-Oct-2020	Morning	NSW	Men	0.238095	0.238095
62	1-Oct-2020	Morning	NSW	Women	0.507937	0.507937
63	1-Oct-2020	Morning	NSW	Seniors	0.333333	0.333333
64	1-Oct-2020	Afternoon	NSW	Kids	0.603175	0.603175
72	1-Oct-2020	Morning	TAS	Kids	0.174603	0.174603
73	1-Oct-2020	Morning	TAS	Men	0.063492	0.063492
74	1-Oct-2020	Morning	TAS	Women	0.015873	0.015873
75	1-Oct-2020	Morning	TAS	Seniors	0.095238	0.095238
76	1-Oct-2020	Afternoon	TAS	Kids	0.079365	0.079365

In [512…

```
for i, i_df in g:
    print(i)
    print(i_df)
```

```
 NSW
            Date       Time State     Group      Unit     Sales
60     1-Oct-2020    Morning   NSW      Kids  0.587302  0.587302
61     1-Oct-2020    Morning   NSW       Men  0.238095  0.238095
62     1-Oct-2020    Morning   NSW     Women  0.507937  0.507937
63     1-Oct-2020    Morning   NSW   Seniors  0.333333  0.333333
64     1-Oct-2020  Afternoon   NSW      Kids  0.603175  0.603175
...           ...        ...   ...       ...       ...       ...
7543  30-Dec-2020  Afternoon   NSW   Seniors  0.269841  0.269841
7544  30-Dec-2020    Evening   NSW      Kids  0.555556  0.555556
7545  30-Dec-2020    Evening   NSW       Men  0.619048  0.619048
7546  30-Dec-2020    Evening   NSW     Women  0.555556  0.555556
7547  30-Dec-2020    Evening   NSW   Seniors  0.333333  0.333333

[1080 rows x 6 columns]
 NT
            Date       Time State     Group      Unit     Sales
12     1-Oct-2020    Morning    NT      Kids  0.174603  0.174603
13     1-Oct-2020    Morning    NT       Men  0.047619  0.047619
14     1-Oct-2020    Morning    NT     Women  0.031746  0.031746
15     1-Oct-2020    Morning    NT   Seniors  0.126984  0.126984
16     1-Oct-2020  Afternoon    NT      Kids  0.174603  0.174603
...           ...        ...   ...       ...       ...       ...
7495  30-Dec-2020  Afternoon    NT   Seniors  0.174603  0.174603
7496  30-Dec-2020    Evening    NT      Kids  0.206349  0.206349
7497  30-Dec-2020    Evening    NT       Men  0.063492  0.063492
7498  30-Dec-2020    Evening    NT     Women  0.142857  0.142857
7499  30-Dec-2020    Evening    NT   Seniors  0.190476  0.190476

[1080 rows x 6 columns]
 QLD
            Date       Time State     Group      Unit     Sales
48     1-Oct-2020    Morning   QLD      Kids  0.285714  0.285714
49     1-Oct-2020    Morning   QLD       Men  0.253968  0.253968
50     1-Oct-2020    Morning   QLD     Women  0.206349  0.206349
51     1-Oct-2020    Morning   QLD   Seniors  0.190476  0.190476
52     1-Oct-2020  Afternoon   QLD      Kids  0.253968  0.253968
...           ...        ...   ...       ...       ...       ...
7531  30-Dec-2020  Afternoon   QLD   Seniors  0.111111  0.111111
7532  30-Dec-2020    Evening   QLD      Kids  0.174603  0.174603
7533  30-Dec-2020    Evening   QLD       Men  0.365079  0.365079
7534  30-Dec-2020    Evening   QLD     Women  0.285714  0.285714
7535  30-Dec-2020    Evening   QLD   Seniors  0.253968  0.253968

[1080 rows x 6 columns]
 SA
            Date       Time State     Group      Unit     Sales
24     1-Oct-2020    Morning    SA      Kids  0.158730  0.158730
25     1-Oct-2020    Morning    SA       Men  0.158730  0.158730
26     1-Oct-2020    Morning    SA     Women  0.222222  0.222222
27     1-Oct-2020    Morning    SA   Seniors  0.412698  0.412698
28     1-Oct-2020  Afternoon    SA      Kids  0.222222  0.222222
...           ...        ...   ...       ...       ...       ...
7507  30-Dec-2020  Afternoon    SA   Seniors  0.269841  0.269841
7508  30-Dec-2020    Evening    SA      Kids  0.460317  0.460317
7509  30-Dec-2020    Evening    SA       Men  0.206349  0.206349
7510  30-Dec-2020    Evening    SA     Women  0.507937  0.507937
7511  30-Dec-2020    Evening    SA   Seniors  0.507937  0.507937

[1080 rows x 6 columns]
 TAS
            Date       Time State     Group      Unit     Sales
72     1-Oct-2020    Morning   TAS      Kids  0.174603  0.174603
73     1-Oct-2020    Morning   TAS       Men  0.063492  0.063492
```

```
74     1-Oct-2020     Morning   TAS    Women  0.015873  0.015873
75     1-Oct-2020     Morning   TAS  Seniors  0.095238  0.095238
76     1-Oct-2020   Afternoon   TAS     Kids  0.079365  0.079365
...           ...         ...   ...      ...       ...       ...
7555  30-Dec-2020   Afternoon   TAS  Seniors  0.190476  0.190476
7556  30-Dec-2020     Evening   TAS     Kids  0.206349  0.206349
7557  30-Dec-2020     Evening   TAS      Men  0.206349  0.206349
7558  30-Dec-2020     Evening   TAS    Women  0.142857  0.142857
7559  30-Dec-2020     Evening   TAS  Seniors  0.174603  0.174603

[1080 rows x 6 columns]
 VIC
             Date        Time State    Group      Unit     Sales
36     1-Oct-2020     Morning   VIC     Kids  0.746032  0.746032
37     1-Oct-2020     Morning   VIC      Men  0.539683  0.539683
38     1-Oct-2020     Morning   VIC    Women  0.507937  0.507937
39     1-Oct-2020     Morning   VIC  Seniors  0.380952  0.380952
40     1-Oct-2020   Afternoon   VIC     Kids  0.460317  0.460317
...           ...         ...   ...      ...       ...       ...
7519  30-Dec-2020   Afternoon   VIC  Seniors  0.952381  0.952381
7520  30-Dec-2020     Evening   VIC     Kids  0.444444  0.444444
7521  30-Dec-2020     Evening   VIC      Men  0.523810  0.523810
7522  30-Dec-2020     Evening   VIC    Women  0.444444  0.444444
7523  30-Dec-2020     Evening   VIC  Seniors  0.476190  0.476190

[1080 rows x 6 columns]
 WA
             Date        Time State    Group      Unit     Sales
0      1-Oct-2020     Morning    WA     Kids  0.095238  0.095238
1      1-Oct-2020     Morning    WA      Men  0.095238  0.095238
2      1-Oct-2020     Morning    WA    Women  0.031746  0.031746
3      1-Oct-2020     Morning    WA  Seniors  0.206349  0.206349
4      1-Oct-2020   Afternoon    WA     Kids  0.015873  0.015873
...           ...         ...   ...      ...       ...       ...
7483  30-Dec-2020   Afternoon    WA  Seniors  0.206349  0.206349
7484  30-Dec-2020     Evening    WA     Kids  0.063492  0.063492
7485  30-Dec-2020     Evening    WA      Men  0.063492  0.063492
7486  30-Dec-2020     Evening    WA    Women  0.190476  0.190476
7487  30-Dec-2020     Evening    WA  Seniors  0.111111  0.111111

[1080 rows x 6 columns]
```

Performed one hot encoding to convert the categorical variables in Unit and Sales column to machine readable numeric form

In [513…

```python
from sklearn.preprocessing import LabelEncoder
le_Time=LabelEncoder()
le_State= LabelEncoder()
le_Group = LabelEncoder()
```

In [514…

```python
df['Time_n']= le_Time.fit_transform(df['Time'])
df['State_n']= le_State.fit_transform(df['State'])
df['Group_n']= le_Group.fit_transform(df['Group'])
df.head()
```

Out[514]:

	Date	Time	State	Group	Unit	Sales	Time_n	State_n	Group_n
0	1-Oct-2020	Morning	WA	Kids	0.095238	0.095238	2	6	0
1	1-Oct-2020	Morning	WA	Men	0.095238	0.095238	2	6	1
2	1-Oct-2020	Morning	WA	Women	0.031746	0.031746	2	6	3
3	1-Oct-2020	Morning	WA	Seniors	0.206349	0.206349	2	6	2
4	1-Oct-2020	Afternoon	WA	Kids	0.015873	0.015873	0	6	0

In [515…

```python
df.head()
df = df.reset_index()  # Resets the index, turning 'Date' back into a column
print(df.head())  # Check the first few rows
```

```
   index        Date       Time State    Group      Unit     Sales  Time_n  \
0      0  1-Oct-2020    Morning    WA     Kids  0.095238  0.095238       2   
1      1  1-Oct-2020    Morning    WA      Men  0.095238  0.095238       2   
2      2  1-Oct-2020    Morning    WA    Women  0.031746  0.031746       2   
3      3  1-Oct-2020    Morning    WA  Seniors  0.206349  0.206349       2   
4      4  1-Oct-2020  Afternoon    WA     Kids  0.015873  0.015873       0   

   State_n  Group_n  
0        6        0  
1        6        1  
2        6        3  
3        6        2  
4        6        0  
```

Calculated the mean and Standard Deviation of the Unit Column

In [516…

```python
mean_value = df['Unit'].mean()
median_value = df['Unit'].median()
std_value = df['Unit'].std()

print("Mean of Unit column:", mean_value)
print("Median of Unit column:", median_value)
print("Standard Deviation of Unit column:", std_value)
```

```
Mean of Unit column: 0.2540543377844965
Median of Unit column: 0.19047619047619047
Standard Deviation of Unit column: 0.20478417107280086
```

Calculated the mean and standard deviation of the Sales Column

In [517…

```python
mean_value = df['Sales'].mean()
median_value = df['Sales'].median()
std_value = df['Sales'].std()

print("Mean of Sales column:", mean_value)
print("Median of Sales column:", median_value)
print("Standard Deviation of Sales column:", std_value)
```

```
Mean of Sales column: 0.25405433778449654
Median of Sales column: 0.1904761904761905
Standard Deviation of Sales column: 0.2047841710728009
```

Identifying the group with the highest sales and lowest sales respectively

In [518…

```
# Group by 'Group' and calculate total sales
group_sales = df.groupby('Group_n')['Sales'].sum()

# Identify the group with the highest and lowest sales
highest_sales_group = group_sales.idxmax()
lowest_sales_group = group_sales.idxmin()

# Get the sales values
highest_sales_value = group_sales.max()
lowest_sales_value = group_sales.min()

print(f"Group with the highest sales: {highest_sales_group} ({highest_sales_value})
print(f"Group with the lowest sales: {lowest_sales_group} ({lowest_sales_value})")
```

```
Group with the highest sales: 1 (484.44444444444446)
Group with the lowest sales: 2 (473.57142857142856)
```

In [519…

```
df.columns
```

Out[519]:

```
Index(['index', 'Date', 'Time', 'State', 'Group', 'Unit', 'Sales', 'Time_n',
       'State_n', 'Group_n'],
      dtype='object')
```

Gathering the weekly, Monthly and quarterly report

In [520…

```
import pandas as pd

# Assuming `df` contains 'Date' and 'Sales' columns
df['Date'] = pd.to_datetime(df['Date'])  # Ensure the Date column is in datetime fo

# Set the Date column as the index for resampling
df.set_index('Date', inplace=True)

# Weekly Report
weekly_report = df.resample('W').sum()

# Monthly Report
monthly_report = df.resample('M').sum()

# Quarterly Report
quarterly_report = df.resample('Q').sum()

# Reset index for reports (optional, for better formatting)
weekly_report.reset_index(inplace=True)
monthly_report.reset_index(inplace=True)
quarterly_report.reset_index(inplace=True)

# Save reports to files
weekly_report.to_csv('weekly_report.csv', index=False)
monthly_report.to_csv('monthly_report.csv', index=False)
quarterly_report.to_csv('quarterly_report.csv', index=False)

# Print summaries
print("Weekly Report:\n", weekly_report.head())
print("Monthly Report:\n", monthly_report.head())
print("Quarterly Report:\n", quarterly_report.head())
```

```
Weekly Report:
         Date    index                                                Time  \
0 2020-10-04    56280   Morning Morning Morning Morning Afternoon Aft...
1 2020-10-11   370146   Morning Morning Morning Morning Afternoon Aft...
2 2020-10-18   715890   Morning Morning Morning Morning Afternoon Aft...
3 2020-10-25  1061634   Morning Morning Morning Morning Afternoon Aft...
4 2020-11-01  1185156   Morning Morning Morning Morning Afternoon Aft...

                                               State  \
0   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...
1   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...
2   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...
3   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...
4   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...

                                               Group        Unit       Sales  \
0   Kids Men Women Seniors Kids Men Women Seniors...   84.857143   84.857143
1   Kids Men Women Seniors Kids Men Women Seniors...  152.777778  152.777778
2   Kids Men Women Seniors Kids Men Women Seniors...  150.476190  150.476190
3   Kids Men Women Seniors Kids Men Women Seniors...  151.587302  151.587302
4   Kids Men Women Seniors Kids Men Women Seniors...  122.460317  122.460317

   Time_n  State_n  Group_n
0     336     1008      504
1     588     1764      882
2     588     1764      882
3     588     1764      882
4     504     1512      756
Monthly Report:
         Date     index                                                Time  \
0 2020-10-31   3173940   Morning Morning Morning Morning Afternoon Aft...
1 2020-11-30   9524340   Morning Morning Morning Morning Afternoon Aft...
2 2020-12-31  15874740   Morning Morning Morning Morning Afternoon Aft...

                                               State  \
0   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...
1   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...
2   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...

                                               Group        Unit       Sales  \
0   Kids Men Women Seniors Kids Men Women Seniors...  645.650794  645.650794
1   Kids Men Women Seniors Kids Men Women Seniors...  495.761905  495.761905
2   Kids Men Women Seniors Kids Men Women Seniors...  779.238095  779.238095

   Time_n  State_n  Group_n
0    2520     7560     3780
1    2520     7560     3780
2    2520     7560     3780
Quarterly Report:
         Date     index                                                Time  \
0 2020-12-31  28573020   Morning Morning Morning Morning Afternoon Aft...

                                               State  \
0   WA WA WA WA WA WA WA WA WA WA WA WA NT NT NT ...

                                               Group         Unit  \
0   Kids Men Women Seniors Kids Men Women Seniors...  1920.650794

         Sales  Time_n  State_n  Group_n
0  1920.650794    7560    22680    11340
```

In [521…

```
df_new= pd.read_csv("C:\\Users\\captr\\OneDrive\\Desktop\\AusApparalSales4thQrt2020
df_new.head(3)
```

Out[521]:

	Date	Time	State	Group	Unit	Sales
0	1-Oct-2020	Morning	WA	Kids	8	20000
1	1-Oct-2020	Morning	WA	Men	8	20000
2	1-Oct-2020	Morning	WA	Women	4	10000

Plotted Group Vs States

In [522…

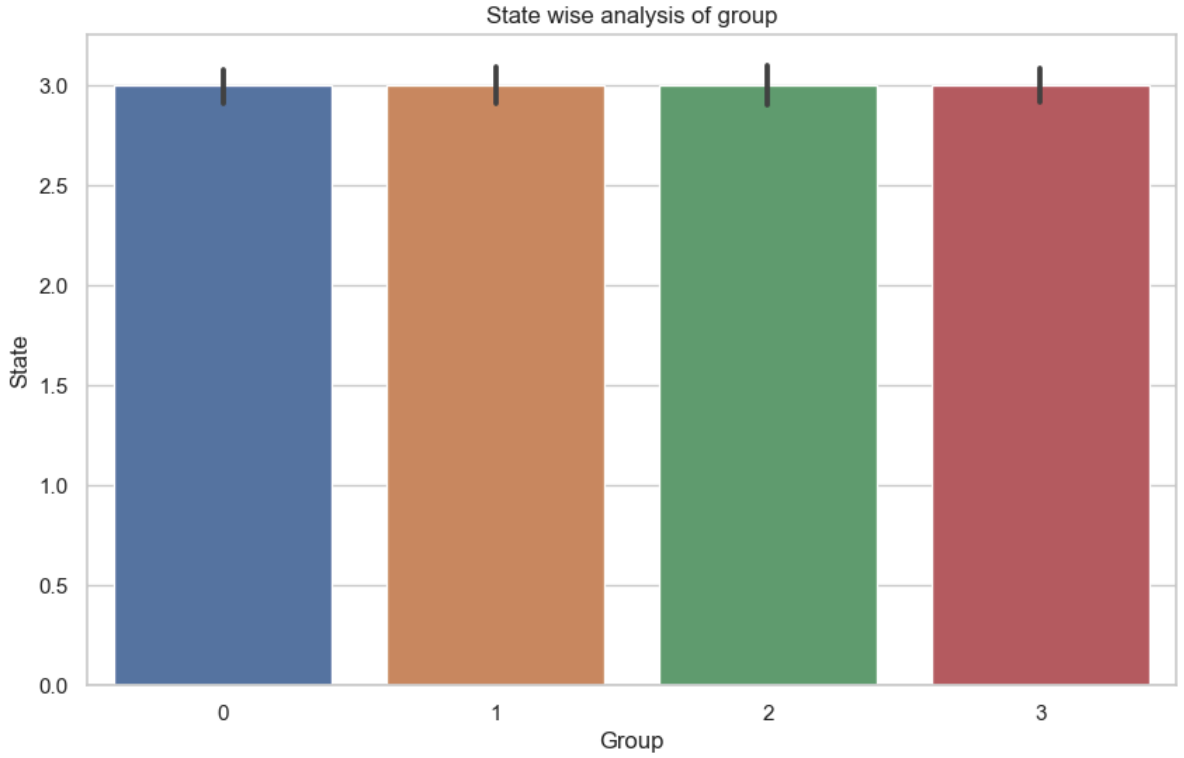
```
Group_counts= df_new['Group'].value_counts()
Group_counts
top_groups=Group_counts.head(10)
top_groups
```

Out[522]:

```
Group
 Kids       1890
 Men        1890
 Women      1890
 Seniors    1890
Name: count, dtype: int64
```

In [523…

```
import seaborn as sns
plt.figure(figsize=(10,6))
sns.barplot(x=df['Group_n'], y=df['State_n'])
plt.xlabel('Group')
plt.ylabel('State')
plt.title('State wise analysis of group')
plt.show()
```



In [524…

```
demography= df.groupby(['Group_n','State_n'])
df=df.reset_index()
df.head()
```

Out[524]:

	Date	index	Time	State	Group	Unit	Sales	Time_n	State_n	Group_n
0	2020-10-01	0	Morning	WA	Kids	0.095238	0.095238	2	6	0
1	2020-10-01	1	Morning	WA	Men	0.095238	0.095238	2	6	1
2	2020-10-01	2	Morning	WA	Women	0.031746	0.031746	2	6	3
3	2020-10-01	3	Morning	WA	Seniors	0.206349	0.206349	2	6	2
4	2020-10-01	4	Afternoon	WA	Kids	0.015873	0.015873	0	6	0

State-wise Sales Analysis for Different Demographic Groups

In [525…

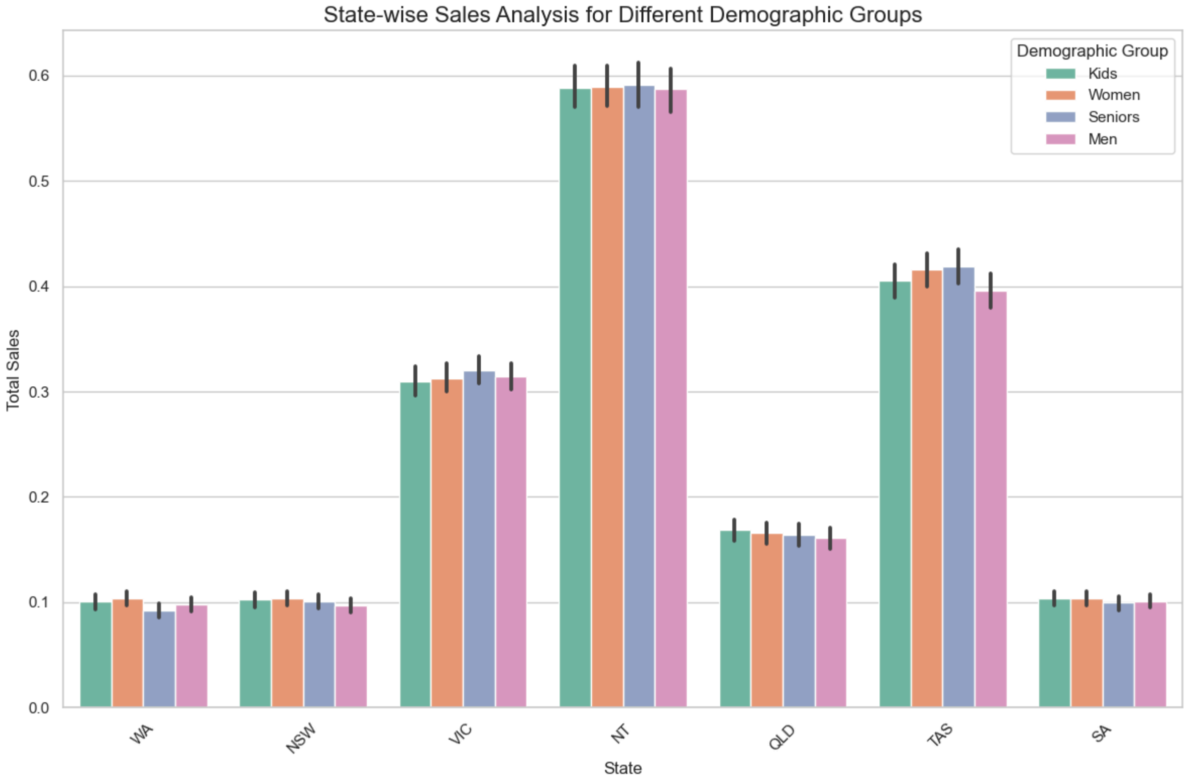
```python
# Mapping Group Codes to Demographics
group_mapping = {0: 'Kids', 1: 'Women', 2: 'Men', 3: 'Seniors'}
df['Demographic'] = df['Group_n'].map(group_mapping)
group_mapping_2 = {0: 'TAS', 1: 'NSW', 2: 'QLD', 3: 'VIC', 4:'SA', 5:'NT', 6:'WA'}
df['Region']= df['State_n'].map(group_mapping_2)

sales_data = df.groupby(['State_n', 'Demographic'])['Sales'].sum().reset_index()
sales_data
```

Out[525]:

	State_n	Demographic	Sales
0	0	Kids	109.444444
1	0	Men	106.904762
2	0	Seniors	113.158730
3	0	Women	112.206349
4	1	Kids	27.619048
5	1	Men	26.126984
6	1	Seniors	27.317460
7	1	Women	28.015873
8	2	Kids	45.460317
9	2	Men	43.428571
10	2	Seniors	44.285714
11	2	Women	44.714286
12	3	Kids	83.587302
13	3	Men	84.873016
14	3	Seniors	86.476190
15	3	Women	84.476190
16	4	Kids	28.095238
17	4	Men	27.301587
18	4	Seniors	26.841270
19	4	Women	27.984127
20	5	Kids	158.793651
21	5	Men	158.507937
22	5	Seniors	159.571429
23	5	Women	159.095238
24	6	Kids	27.142857
25	6	Men	26.428571
26	6	Seniors	24.841270
27	6	Women	27.952381

In [526…

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Set plot style
sns.set(style="whitegrid")

# Create the bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Region', y='Sales', hue='Demographic', data=df, palette="Set2")

# Customize plot
plt.title("State-wise Sales Analysis for Different Demographic Groups", fontsize=1
plt.xlabel("State", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.legend(title="Demographic Group")
plt.xticks(rotation=45)
plt.tight_layout()

# Show plot
plt.show()
```



In [527…

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

Out[527]:

```
array([6, 1, 3, 5, 2, 0, 4])
```

In [528…

```
df.tail()
```

Out[528]:

	Date	index	Time	State	Group	Unit	Sales	Time_n	State_n	Group_n	Den
7555	2020-12-30	7555	Afternoon	TAS	Seniors	0.190476	0.190476	0	4	2	
7556	2020-12-30	7556	Evening	TAS	Kids	0.206349	0.206349	1	4	0	
7557	2020-12-30	7557	Evening	TAS	Men	0.206349	0.206349	1	4	1	
7558	2020-12-30	7558	Evening	TAS	Women	0.142857	0.142857	1	4	3	
7559	2020-12-30	7559	Evening	TAS	Seniors	0.174603	0.174603	1	4	2	

State-wise Sales Analysis for Different Regions

In [529…

```
group_mapping = {0: 'Kids', 1: 'Women', 2: 'Men', 3: 'Seniors'}
df['Demographic'] = df['Group_n'].map(group_mapping)
```

```
group_mapping_2 = {0: 'TAS', 1: 'NSW', 2: 'QLD', 3: 'VIC', 4:'SA', 5:'NT', 6:'WA'}
df['Region'] = df['State_n'].map(group_mapping_2)

sales_data_2 = df.groupby(['Group_n', 'Region'])['Sales'].sum().reset_index()
df.head()
```

Out[529]:

	Date	index	Time	State	Group	Unit	Sales	Time_n	State_n	Group_n	Demog
0	2020-10-01	0	Morning	WA	Kids	0.095238	0.095238	2	6	0	
1	2020-10-01	1	Morning	WA	Men	0.095238	0.095238	2	6	1	W
2	2020-10-01	2	Morning	WA	Women	0.031746	0.031746	2	6	3	S
3	2020-10-01	3	Morning	WA	Seniors	0.206349	0.206349	2	6	2	
4	2020-10-01	4	Afternoon	WA	Kids	0.015873	0.015873	0	6	0	

In [530…

```
sales_data_2
```

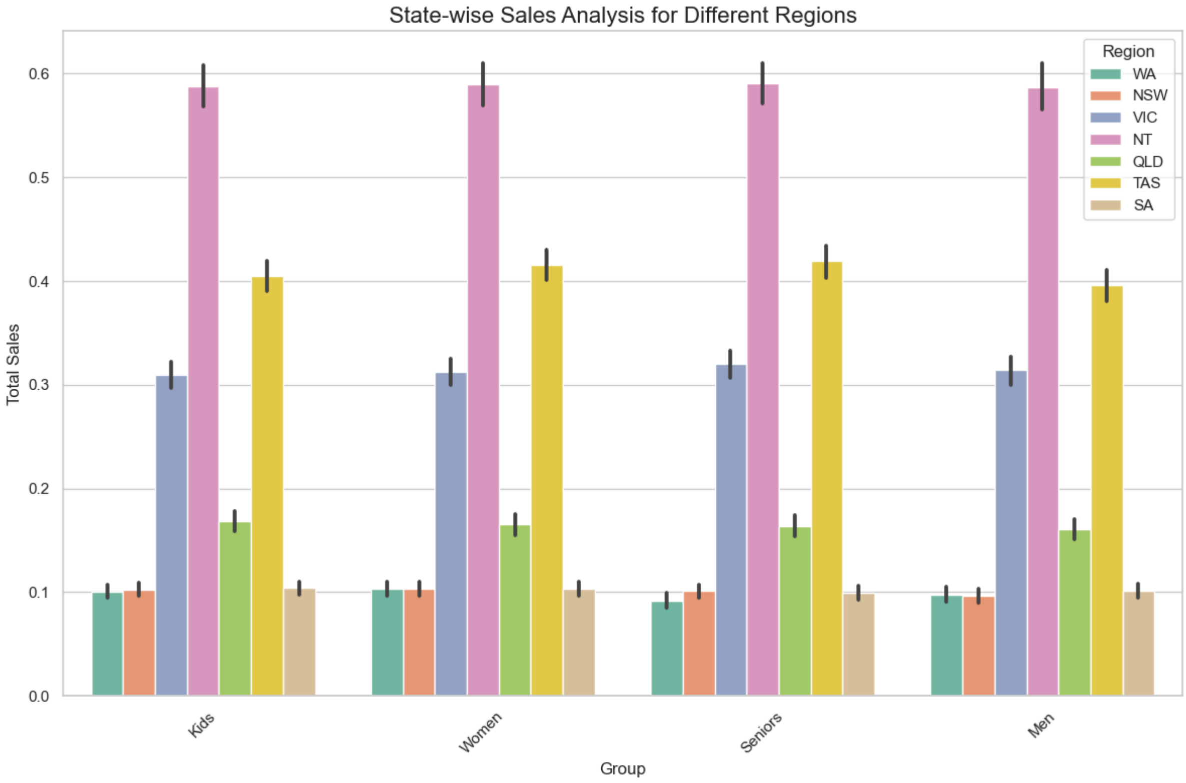
Out[530]:

	Group_n	Region	Sales
0	0	NSW	27.619048
1	0	NT	158.793651
2	0	QLD	45.460317
3	0	SA	28.095238
4	0	TAS	109.444444
5	0	VIC	83.587302
6	0	WA	27.142857
7	1	NSW	28.015873
8	1	NT	159.095238
9	1	QLD	44.714286
10	1	SA	27.984127
11	1	TAS	112.206349
12	1	VIC	84.476190
13	1	WA	27.952381
14	2	NSW	26.126984
15	2	NT	158.507937
16	2	QLD	43.428571
17	2	SA	27.301587
18	2	TAS	106.904762
19	2	VIC	84.873016
20	2	WA	26.428571
21	3	NSW	27.317460
22	3	NT	159.571429
23	3	QLD	44.285714
24	3	SA	26.841270
25	3	TAS	113.158730
26	3	VIC	86.476190
27	3	WA	24.841270

In [531…

```python
# Ensure 'sales_data_2' has the correct data
print(sales_data_2.head())  # Check the first few rows

# Create the bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Demographic', y='Sales', hue='Region', data=df, palette="Set2")

# Customize the plot
plt.title("State-wise Sales Analysis for Different Regions", fontsize=16)
plt.xlabel("Group", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.legend(title="Region")
plt.xticks(rotation=45)
plt.tight_layout()

# Show plot
plt.show()
```

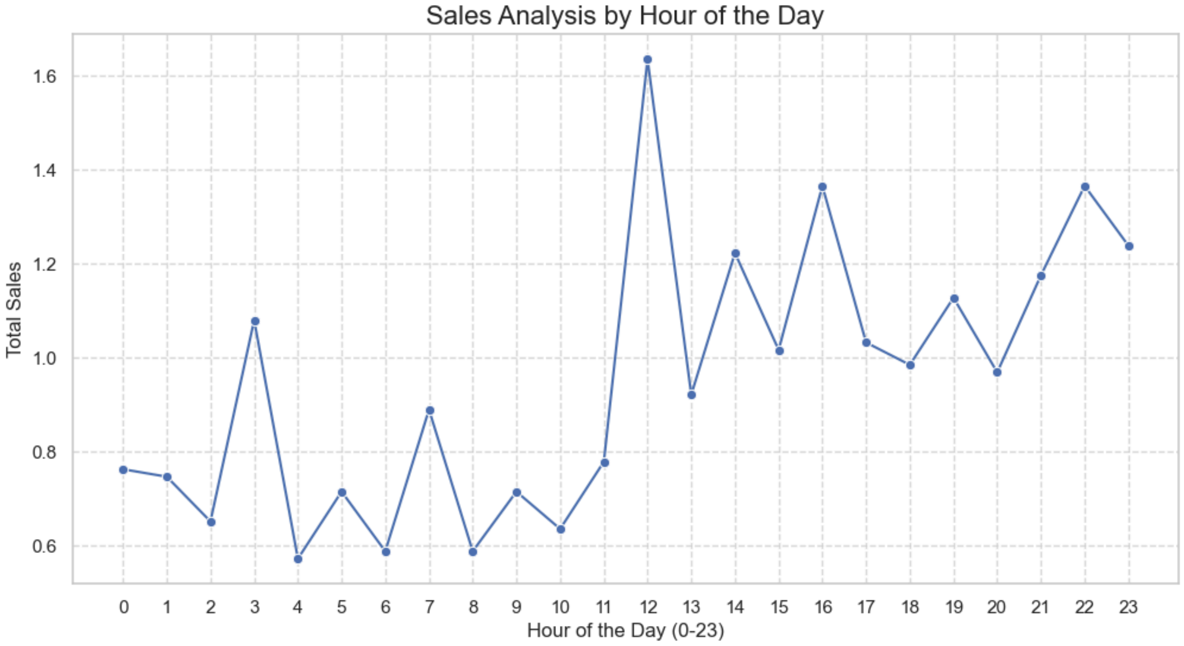
```
   Group_n Region       Sales
0        0    NSW   27.619048
1        0     NT  158.793651
2        0    QLD   45.460317
3        0     SA   28.095238
4        0    TAS  109.444444
```



Sales Analysis by Hour of the Day

In [532…

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Sample DataFrame (Use your actual dataset)
# Ensure 'Date' and 'Time' are in proper datetime format
data = pd.DataFrame({
    'Date': pd.date_range(start='2020-01-01', periods=100, freq='H'),
    'Sales': [i % 24 + 1 for i in range(100)]
})

# Ensure 'Date' is in datetime format
data['Date'] = pd.to_datetime(data['Date'])

# Extract the hour from the Date column
df['Hour_of_day'] = data['Date'].dt.hour

# Group sales data by hour of the day
sales_by_time = df.groupby('Hour_of_day')['Sales'].sum().reset_index()

# Plotting
plt.figure(figsize=(12, 6))
sns.lineplot(x='Hour_of_day', y='Sales', data=sales_by_time, marker="o", color="b"
# Formatting the plot
plt.title("Sales Analysis by Hour of the Day", fontsize=16)
plt.xlabel("Hour of the Day (0-23)", fontsize=12)
plt.ylabel("Total Sales", fontsize=12)
plt.xticks(range(0, 24))  # Ensure all hours from 0 to 23 are displayed
plt.grid(True, linestyle="--", alpha=0.7)
plt.show()
```



In [533…

```
sales_by_time = df.groupby('Hour_of_day')['Sales'].sum().reset_index()
sales_by_time
```

Out[533]:

	Hour_of_day	Sales
0	0.0	0.761905
1	1.0	0.746032
2	2.0	0.650794
3	3.0	1.079365
4	4.0	0.571429
5	5.0	0.714286
6	6.0	0.587302
7	7.0	0.888889
8	8.0	0.587302
9	9.0	0.714286
10	10.0	0.634921
11	11.0	0.777778
12	12.0	1.634921
13	13.0	0.920635
14	14.0	1.222222
15	15.0	1.015873
16	16.0	1.365079
17	17.0	1.031746
18	18.0	0.984127
19	19.0	1.126984
20	20.0	0.968254
21	21.0	1.174603
22	22.0	1.365079
23	23.0	1.238095

In [534…

```
df.head()
```

Out[534]:

	Date	index	Time	State	Group	Unit	Sales	Time_n	State_n	Group_n	Demog
0	2020-10-01	0	Morning	WA	Kids	0.095238	0.095238	2	6	0	
1	2020-10-01	1	Morning	WA	Men	0.095238	0.095238	2	6	1	W
2	2020-10-01	2	Morning	WA	Women	0.031746	0.031746	2	6	3	S
3	2020-10-01	3	Morning	WA	Seniors	0.206349	0.206349	2	6	2	
4	2020-10-01	4	Afternoon	WA	Kids	0.015873	0.015873	0	6	0	

Plotting Correlation matrix to identify the important features that affects the sales

In [543…

```
df_numeric= df.drop(['Time','State','Group','Demographic','Region'],axis='columns'
df_numeric
```

Out[543]:

	Date	index	Unit	Sales	Time_n	State_n	Group_n	Hour_of_day
0	2020-10-01	0	0.095238	0.095238	2	6	0	0.0
1	2020-10-01	1	0.095238	0.095238	2	6	1	1.0
2	2020-10-01	2	0.031746	0.031746	2	6	3	2.0
3	2020-10-01	3	0.206349	0.206349	2	6	2	3.0
4	2020-10-01	4	0.015873	0.015873	0	6	0	4.0
...	...	...	...	...	...	...	...	...
7555	2020-12-30	7555	0.190476	0.190476	0	4	2	NaN
7556	2020-12-30	7556	0.206349	0.206349	1	4	0	NaN
7557	2020-12-30	7557	0.206349	0.206349	1	4	1	NaN
7558	2020-12-30	7558	0.142857	0.142857	1	4	3	NaN
7559	2020-12-30	7559	0.174603	0.174603	1	4	2	NaN

7560 rows × 8 columns

In [544…

```
corr_matrix= df_numeric.corr()
corr_matrix
```

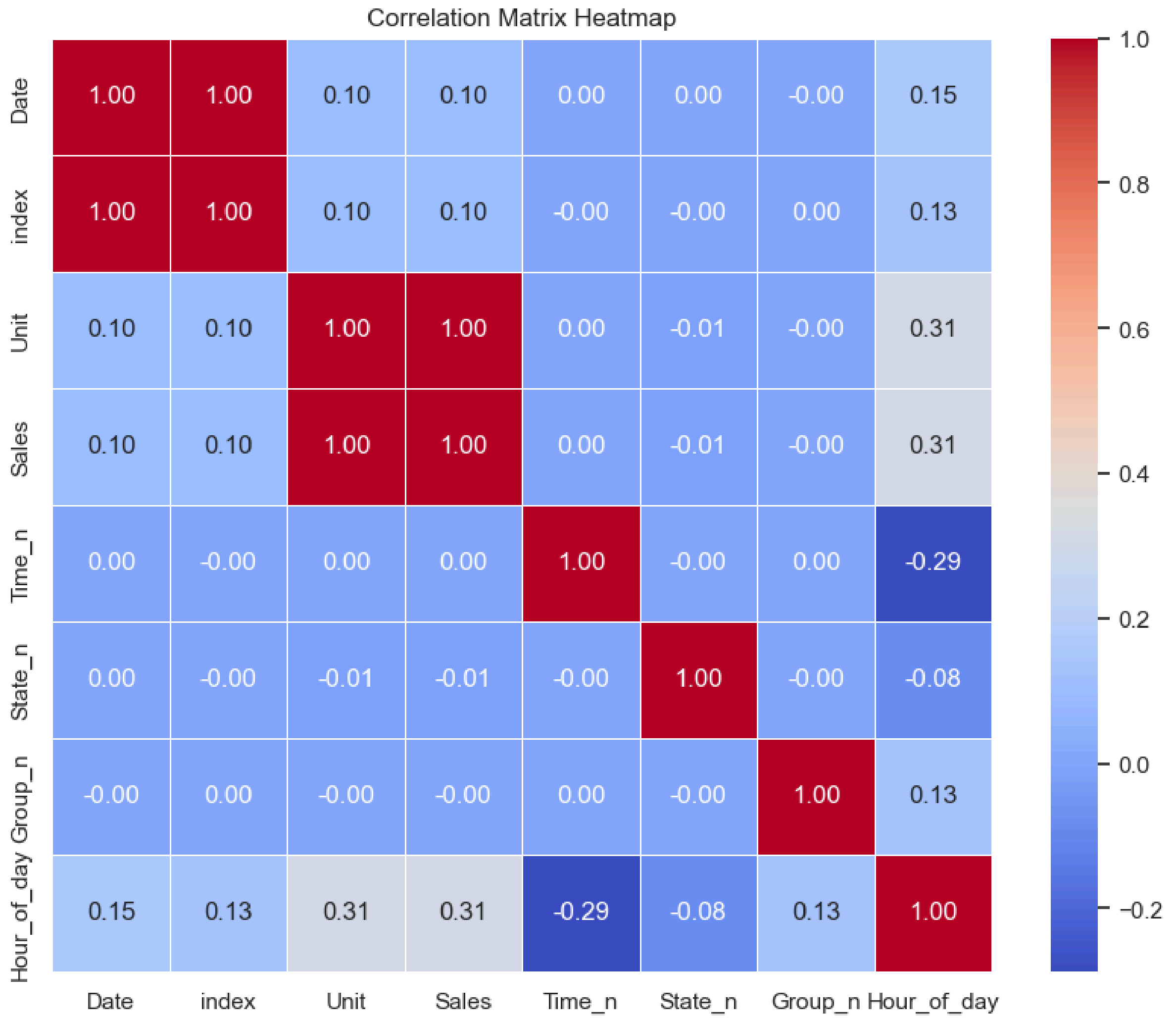
Out[544]:

	Date	index	Unit	Sales	Time_n	State_n	Grou
Date	1.000000e+00	0.999885	0.100658	0.100658	2.593151e-15	2.481039e-15	-1.3179
index	9.998850e-01	1.000000	0.104095	0.104095	-7.482612e-04	-3.534798e-03	4.098395
Unit	1.006582e-01	0.104095	1.000000	1.000000	1.004629e-03	-6.439166e-03	-1.1051
Sales	1.006582e-01	0.104095	1.000000	1.000000	1.004629e-03	-6.439166e-03	-1.1051
Time_n	2.593151e-15	-0.000748	0.001005	0.001005	1.000000e+00	-3.003558e-18	5.743231
State_n	2.481039e-15	-0.003535	-0.006439	-0.006439	-3.003558e-18	1.000000e+00	-3.5990
Group_n	-1.317944e-15	0.000410	-0.001105	-0.001105	5.743231e-18	-3.599018e-18	1.000000e
Hour_of_day	1.482944e-01	0.131689	0.309362	0.309362	-2.860248e-01	-8.309249e-02	1.266305

In [545…

```
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
```

```
plt.title("Correlation Matrix Heatmap")
plt.show()
```



from the correlation matrix we can clearly see that only Unit and Hour of the day column/feature has a strong correlation with the Sales. But still we would confirm this with Hypothesis testing

In [546…

```
df_numeric.head()
```

Out[546]:

	Date	index	Unit	Sales	Time_n	State_n	Group_n	Hour_of_day
0	2020-10-01	0	0.095238	0.095238	2	6	0	0.0
1	2020-10-01	1	0.095238	0.095238	2	6	1	1.0
2	2020-10-01	2	0.031746	0.031746	2	6	3	2.0
3	2020-10-01	3	0.206349	0.206349	2	6	2	3.0
4	2020-10-01	4	0.015873	0.015873	0	6	0	4.0

In [558…

```
data = df_numeric.rename(columns=lambda x: x.strip())  # Removes leading/trailing s
data.head()
```

Out[558]:

	Date	index	Unit	Sales	Time_n	State_n	Group_n	Hour_of_day
0	2020-10-01	0	0.095238	0.095238	2	6	0	0.0
1	2020-10-01	1	0.095238	0.095238	2	6	1	1.0
2	2020-10-01	2	0.031746	0.031746	2	6	3	2.0
3	2020-10-01	3	0.206349	0.206349	2	6	2	3.0
4	2020-10-01	4	0.015873	0.015873	0	6	0	4.0

Null Hypothesis (H0): "Units sold has nothing to do with sales revenue"

In [560…

```
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf


Unit_Sales_corr, Unit_Sales_pval = stats.spearmanr(data['Unit'], data['Sales'])

print(f"Spearman correlation between Unit and Sales: {Unit_Sales_corr:.2f}, p-value

# Regression analysis for age and shopping preferences
Unit_Sales_model = smf.ols('Sales ~ Unit', data=data).fit()


print("\nIn-store Purchases Regression Summary:\n", Unit_Sales_model.summary())
```

```
Spearman correlation between Unit and Sales: 1.00, p-value: 0.0000

In-store Purchases Regression Summary:
                             OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.949e+33
Date:                Sat, 01 Feb 2025   Prob (F-statistic):               0.00
Time:                        17:43:01   Log-Likelihood:             2.5725e+05
No. Observations:                7560   AIC:                        -5.145e+05
Df Residuals:                    7558   BIC:                        -5.145e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -5.345e-16   7.39e-18    -72.324      0.000    -5.49e-16    -5.2e-16
Unit           1.0000   2.26e-17   4.42e+16      0.000       1.000       1.000
==============================================================================
Omnibus:                     1514.899   Durbin-Watson:                   0.054
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2734.456
Skew:                          -1.271   Prob(JB):                         0.00
Kurtosis:                       4.491   Cond. No.                         5.21
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

Since the p-value is < 0.05 The result is highly statistically significant, meaning there is strong evidence to reject the null

hypothesis that "Units sold has nothing to do with sales revenue".

Model Construction

Input Variable (X): Unit and Hour of the day

In [ ]:
```
X= df_numeric.drop(['Date','index','Sales','Time_n','State_n','Group_n'],axis= 'col
X.head()
```

In [ ]:
```
X['Hour_of_day'].isnull().sum()
```

In [ ]:
```
X['Hour_of_day'].fillna(X['Hour_of_day'].mode()[0], inplace=True)  # Fill with mode
```

Target Variable (Y) : Sales

In [ ]:
```
Y= df['Sales']
Y.head()
```

Splitting the dataset into train and test parts where we are using 80% our data to train and 20% to test

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test= train_test_split(X,Y,test_size=0.2, random_state=
```

Trying to plot a Linear regression model as we had previously seen that our data follows a linear pattern

In [ ]:
```
from sklearn.linear_model import LinearRegression
reg= LinearRegression()
reg.fit(X_train, Y_train)
reg.score(X_test,Y_test)
```

The R2 score of our model resulted in a perfect fit.

In [ ]:
```
Y_pred= reg.predict(X_test)
Y_pred
```

In [ ]:
```
len(Y_pred)
```

In [ ]:
```
len(Y)
```

In [ ]:
```
Y = Y[:len(Y_pred)]
r2 = r2_score(Y_test, Y_pred)
```

In [ ]:
```
from sklearn.metrics import mean_squared_error, r2_score
r2 = r2_score(Y_test, Y_pred)
print("R-squared Score:", r2)
```

Calculated the root mean square error as well to see how our model is performing

In [ ]:

```
rmse = np.sqrt(mean_squared_error(Y_test, Y_pred))
rmse
```

As it can be seen that our Root Mean Squared Error (RMSE) = 3.42×10^{-16} is extremely close to zero, which aligns with our $R^2 = 1.0$. This means our model is predicting Y_test almost perfectly. However, this is highly unusual in real-world scenarios and often indicates overfitting. Hence just to confirm we are using K-fold cross validation and Ridge Regression

In [ ]:

```
import numpy as np
from sklearn.model_selection import cross_val_score
reg= LinearRegression()
Score= cross_val_score(reg,X_test,Y_test,cv=3)
np.average(Score)
```

In [ ]:

```
from sklearn.linear_model import Ridge
ridge_reg= Ridge(alpha=50, max_iter=100, tol=0.1)
ridge_reg.fit(X_train,Y_train)
ridge_reg.score(X_test,Y_test)
```

As it can be seen and confirmed from above that the model performs and fits properly even after perform cross validation and ridge regression. This indicates that our model is reliable.

In [ ]:

```
from sklearn.linear_model import LinearRegression

# Instantiate the model
reg = LinearRegression()

# Fit the model on training data
reg.fit(X_train, Y_train)

# Now make predictions on the test data
Y_pred = reg.predict(X_test)

# You can now evaluate the performance using Y_pred
print("Predictions:", Y_pred)
```

In [540…

```
reg.predict([[0.015873,4.0]])
```

```
C:\Users\captr\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X doe
s not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

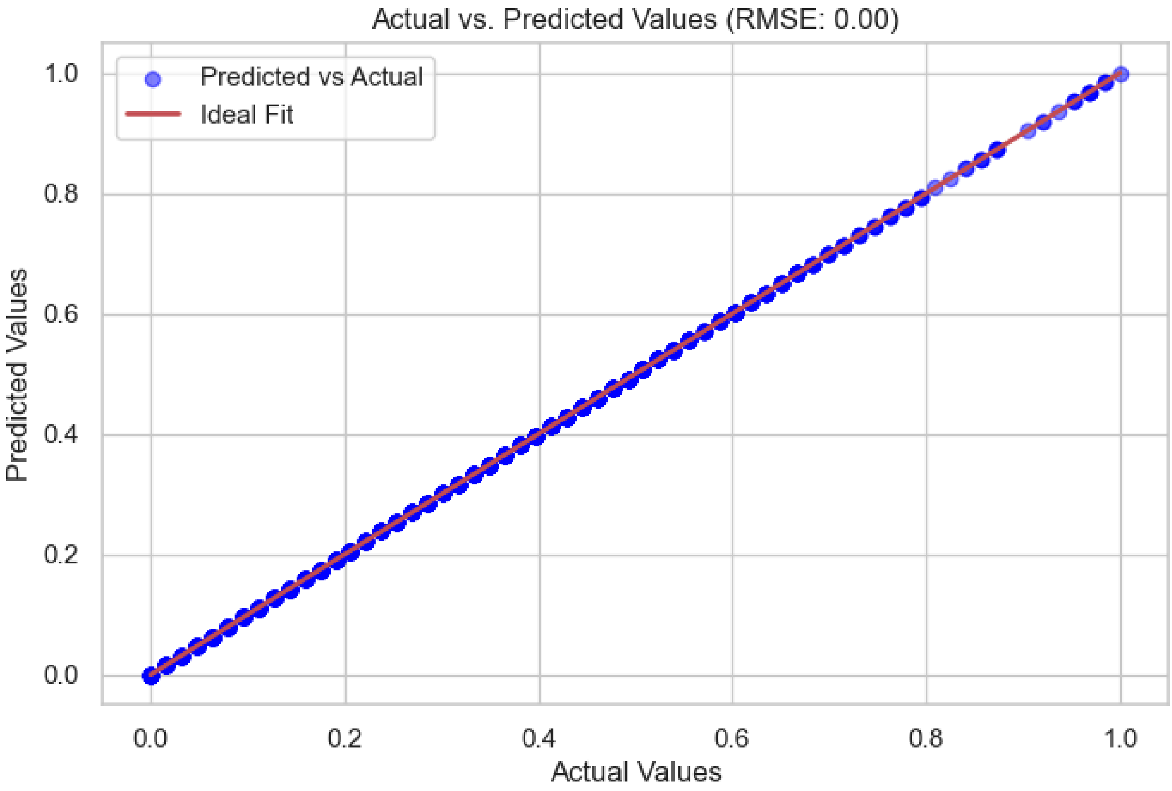
Out[540]:

```
array([0.015873])
```

RMSE Plot

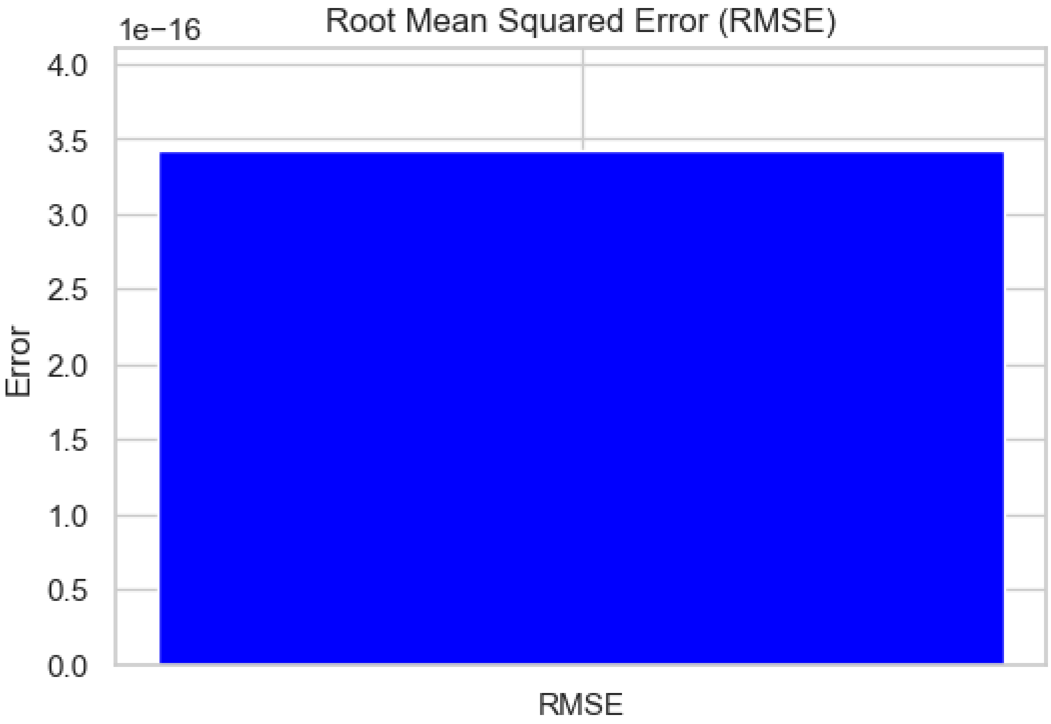
In [541…

```
plt.figure(figsize=(8, 5))
plt.scatter(Y_test, Y_pred, color="blue", alpha=0.5, label="Predicted vs Actual")
plt.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], 'r', lw=2, lat
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title(f"Actual vs. Predicted Values (RMSE: {rmse:.2f})")
plt.legend()
plt.show()
```



In [542…

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error

# Compute RMSE
rmse = np.sqrt(mean_squared_error(Y_test, Y_pred))

# Plot RMSE
plt.figure(figsize=(6, 4))
plt.bar(["RMSE"], [rmse], color='blue')
plt.ylabel("Error")
plt.title("Root Mean Squared Error (RMSE)")
plt.ylim(0, rmse * 1.2)  # Adding some padding for better visualization
plt.show()
```


Root Mean Squared Error (RMSE)
1e−16
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Error
RMSE

```
In [ ]:
```