AS2SalesAnalysis1F

April 7, 2024

0.0.1 Course 3 - Applied Data Science with Python

Assignment 2 - Sales Analysis

Submission by - Prabhat Priyadarshi

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sbn
from matplotlib import pyplot as plt

sales_df = pd.read_csv("AusApparalSales4thQrt2020.csv")
```

[3]:
```
print(sales_df)
```

```
             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
…             …           …   …       …    …    …
7555  30-Dec-2020   Afternoon   TAS   Seniors    14  35000
7556  30-Dec-2020     Evening   TAS      Kids    15  37500
7557  30-Dec-2020     Evening   TAS       Men    15  37500
7558  30-Dec-2020     Evening   TAS     Women    11  27500
7559  30-Dec-2020     Evening   TAS   Seniors    13  32500

[7560 rows x 6 columns]
```

#1 Assignment 2 Tasks:

1. Data Wrangling

   - Ensure that the data is clean and that there is no missing or incorrect data.
   - Inspect the data manually for missing/incorrect data using the functions isna(), and notna().
   - Based on your knowledge of Data Analytics, include your recommendations for treating missing data and incorrect data. (dropping the null values or filling them).

- (skip) Select an appropriate Data Wrangling approach — data standardization or data normalization. Perform the standardization or normalization and present the data. (Normalization is the preferred approach for this problem.) -
- Share your recommendation on the usage of the groupby() function for data chunking or merging.

Cleaning the Data and Checking for the missing values!

Checked for Null. (Null and Not a Number)

[3]:
```
# Checking for NaN values in dataframe.

sales_df.isna().sum()
```

[3]:
```
Date     0
Time     0
State    0
Group    0
Unit     0
Sales    0
dtype: int64
```

[7]:
```
# Data Type before conversion to DateTime object
sales_df['Date'].dtype
```

[7]:
```
dtype('O')
```

[4]:
```
# Making Date consitent to correct datatype

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

[9]:
```
# Data Type after conversion to DateTime object
sales_df['Date'].dtype
```

[9]:
```
dtype('<M8[ns]')
```

[38]:
```
#Adding Month Number of Q4

sales_df['Month'] = sales_df['Date'].dt.month
sales_df['Month'].dtype
```

[38]:
```
dtype('int64')
```

[40]:
```
#Adding Week Numbers in Q4

sales_df['Week'] = sales_df['Date'].dt.isocalendar().week
sales_df['Week'].dtype
```

[40]:
```
UInt32Dtype()
```

[14]:
```
display(sales_df)
```

	Date	Time	State	Group	Unit	Sales	Month	Week
0	2020-10-01	Morning	WA	Kids	8	20000	10	40
1	2020-10-01	Morning	WA	Men	8	20000	10	40
2	2020-10-01	Morning	WA	Women	4	10000	10	40
3	2020-10-01	Morning	WA	Seniors	15	37500	10	40
4	2020-10-01	Afternoon	WA	Kids	3	7500	10	40
…	…	…	…	…	…	…	…	…
7555	2020-12-30	Afternoon	TAS	Seniors	14	35000	12	53
7556	2020-12-30	Evening	TAS	Kids	15	37500	12	53
7557	2020-12-30	Evening	TAS	Men	15	37500	12	53
7558	2020-12-30	Evening	TAS	Women	11	27500	12	53
7559	2020-12-30	Evening	TAS	Seniors	13	32500	12	53

[7560 rows x 8 columns]

[7]:
```
#Cheking Month Numbers

sales_df['Month'].unique()
```

[7]:
```
array([10, 11, 12])
```

[8]:
```
#Checking week numbers

sales_df['Week'].unique()
```

[8]:
```
<IntegerArray>
[40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]
Length: 14, dtype: UInt32
```

[11]:
```
# Copying the dataframe for normalization
norm_sales_df = sales_df.copy()

sales_df.columns
```

[11]:
```
Index(['Date', 'Time', 'State', 'Group', 'Unit', 'Sales', 'Month', 'Week'],
dtype='object')
```

[12]:
```
# Skipped Normalization (Scaling wrt max)

norm_sales_df['Unit'] = norm_sales_df['Unit']  / norm_sales_df['Unit'].abs().
 ↪max()
norm_sales_df['Sales'] = norm_sales_df['Sales']  / norm_sales_df['Sales'].abs().
 ↪max()
```

```
# view normalized data
display(norm_sales_df)
```

```
           Date        Time State     Group      Unit     Sales  Month  Week
0    2020-10-01     Morning    WA      Kids  0.123077  0.123077     10    40
1    2020-10-01     Morning    WA       Men  0.123077  0.123077     10    40
2    2020-10-01     Morning    WA     Women  0.061538  0.061538     10    40
3    2020-10-01     Morning    WA   Seniors  0.230769  0.230769     10    40
4    2020-10-01   Afternoon    WA      Kids  0.046154  0.046154     10    40
…           …         …   …       …       …       …    …   …
7555 2020-12-30   Afternoon   TAS   Seniors  0.215385  0.215385     12    53
7556 2020-12-30     Evening   TAS      Kids  0.230769  0.230769     12    53
7557 2020-12-30     Evening   TAS       Men  0.230769  0.230769     12    53
7558 2020-12-30     Evening   TAS     Women  0.169231  0.169231     12    53
7559 2020-12-30     Evening   TAS   Seniors  0.200000  0.200000     12    53

[7560 rows x 8 columns]
```

Suggested Measures for treating Missing and Incorrect Data.

0.0.2 No Missing or Incorrect Data Was Found, however we can perform:

1. Drop the row or Fill the Missing Data by dropna() or fillna() | sales_df['Sales'].fillna(sa
2. Remove the duplicates by dropping the rows using drop_duplicates()
3. Inconsistent values wrt expected data type in column, using data formating.
    converted Date from O type to DateTime

Data Chunking and Merging using group by

1. Grouping the States and Person Groups and Minimum Units Sold in Each State

[ ]:
```
# Chunking the data by state and group, minimum units sold in each state

display(sales_df.groupby(["State", "Group"])['Unit'].min())
```

```
State  Group
 NSW    Kids       12
        Men        12
        Seniors    12
        Women      12
 NT     Kids        2
        Men         2
        Seniors     2
        Women       2
 QLD    Kids        3
        Men         3
        Seniors     3
```

```
        Women       3
 SA     Kids       10
        Men        10
        Seniors    10
        Women      10
 TAS    Kids        2
        Men         2
        Seniors     2
        Women       2
 VIC    Kids       20
        Men        20
        Seniors    20
        Women      20
 WA     Kids        2
        Men         2
        Seniors     2
        Women       2
Name: Unit, dtype: int64
```

2. Grouping the States and Person Groups and "Maximum" Units Sold in Each State

[19]:
```
# Chunking the data by state and group, maximum units sold in each state

display(sales_df.groupby(["State", "Group"])['Unit'].max())
```

```
State  Group
 NSW    Kids       45
        Men        45
        Seniors    45
        Women      45
 NT     Kids       15
        Men        15
        Seniors    15
        Women      15
 QLD    Kids       25
        Men        25
        Seniors    25
        Women      25
 SA     Kids       35
        Men        35
        Seniors    35
        Women      35
 TAS    Kids       15
        Men        15
        Seniors    15
        Women      15
 VIC    Kids       65
        Men        64
```

```
        Seniors    65
        Women      65
 WA     Kids       15
        Men        15
        Seniors    15
        Women      15
Name: Unit, dtype: int64
```

3. Grouping the States and Person Groups and Average Sales in Each State wrt Each Group

[20]:
```
# Average Sales Data of Groups across various states.

display(pd.DataFrame(sales_df.groupby(['Group','State'], as_index =␣
 ↪False)['Sales'].mean()))
```

```
       Group State         Sales
0       Kids   NSW  68842.592593
1       Kids    NT  21111.111111
2       Kids   QLD  31518.518519
3       Kids    SA  53759.259259
4       Kids   TAS  21388.888889
5       Kids   VIC  97629.629630
6       Kids    WA  20833.333333
7        Men   NSW  70453.703704
8        Men    NT  21342.592593
9        Men   QLD  31083.333333
10       Men    SA  54277.777778
11       Men   TAS  21324.074074
12       Men   VIC  97805.555556
13       Men    WA  21305.555556
14   Seniors   NSW  67361.111111
15   Seniors    NT  20240.740741
16   Seniors   QLD  30333.333333
17   Seniors    SA  54509.259259
18   Seniors   TAS  20925.925926
19   Seniors   VIC  97462.962963
20   Seniors    WA  20416.666667
21     Women   NSW  71009.259259
22     Women    NT  20935.185185
23     Women   QLD  30833.333333
24     Women    SA  55444.444444
25     Women   TAS  20657.407407
26     Women   VIC  98083.333333
27     Women    WA  19490.740741
```

4. Grouping the States and Time of the Day and Maximum Units Sold in Each State

[5]:
```
# Chunking the data by time and maximum units sold at various states.
timeod_state_max_units = pd.DataFrame(sales_df.groupby(["Time", "State"],␣
 ↪as_index=False)['Unit'].max())
display(timeod_state_max_units)
```

	Time	State	Unit
0	Afternoon	NSW	44
1	Afternoon	NT	15
2	Afternoon	QLD	25
3	Afternoon	SA	35
4	Afternoon	TAS	15
5	Afternoon	VIC	65
6	Afternoon	WA	15
7	Evening	NSW	45
8	Evening	NT	15
9	Evening	QLD	25
10	Evening	SA	35
11	Evening	TAS	15
12	Evening	VIC	65
13	Evening	WA	15
14	Morning	NSW	45
15	Morning	NT	15
16	Morning	QLD	25
17	Morning	SA	35
18	Morning	TAS	15
19	Morning	VIC	65
20	Morning	WA	15

5. Grouping the Time of the Day and Person Groups and Maximum Units Sold in Each Group

[22]:
```
# Chunking the data by time and maximum units purchased by various groups.
timeod_group_max_units = pd.DataFrame(sales_df.groupby(["Time", "Group"],␣
 ↪as_index=False)['Unit'].max())
display(timeod_group_max_units)
```

	Time	Group	Unit
0	Afternoon	Kids	65
1	Afternoon	Men	63
2	Afternoon	Seniors	65
3	Afternoon	Women	65
4	Evening	Kids	60
5	Evening	Men	64
6	Evening	Seniors	65
7	Evening	Women	65
8	Morning	Kids	63
9	Morning	Men	64
10	Morning	Seniors	65

```
11      Morning      Women     64
```

0.0.3 2. Data Analysis

* Perform descriptive statistical analysis on the data (Sales and Unit columns) (Techniques s
* Determine which group is generating the highest sales, and which group is generating the lo
* Determine which state is generating the highest sales, and which state is generating the lo
* Generate weekly, monthly and quarterly reports for the analysis made.
  (Use suitable libraries such as NumPy, Pandas, SciPy etc. for performing the analysis.)

2.1 Descriptive Statistical Analysis

[23]:
```
# Primary Descriptive Statistical Analysis using DataFrame describe()
sales_df.describe()
```

[23]:

	Unit	Sales	Month	Week
count	7560.000000	7560.000000	7560.000000	7560.0
mean	18.005423	45013.558201	11.000000	46.455556
std	12.901403	32253.506944	0.816551	3.786662
min	2.000000	5000.000000	10.000000	40.0
25%	8.000000	20000.000000	10.000000	43.0
50%	14.000000	35000.000000	11.000000	46.5
75%	26.000000	65000.000000	12.000000	50.0
max	65.000000	162500.000000	12.000000	53.0

[24]:
```
# Meadian and Mode of the Sales Data Frame
print("Median of Sold Units:" ,sales_df['Unit'].median())
print("Median of Sales Made: ",sales_df['Sales'].median())
print("Mode of Sold Units:" ,sales_df['Unit'].mode())
print("Mode of Sales Made: ",sales_df['Sales'].mode())
```

```
Median of Sold Units: 14.0
Median of Sales Made:  35000.0
Mode of Sold Units: 0    9
Name: Unit, dtype: int64
Mode of Sales Made:  0    22500
Name: Sales, dtype: int64
```

2.2 Which group is generating the highest and lowest sales.

[6]:
```
# Which group is producing the highest and lowest sales.

print("Sales Data by Person Group: \n")
group_group = pd.DataFrame(sales_df.groupby('Group', as_index = False)['Sales'].
 ↪sum()).sort_values(by=['Sales'],ascending=False)
display(group_group)
display("Highest Sales: ",group_group.iloc[0,:],"Lowest Sales: ",group_group.
 ↪iloc[-1,:])
```

```
Sales Data by Person Group:


      Group     Sales
1       Men  85750000
3     Women  85442500
0      Kids  85072500
2   Seniors  84037500

'Highest Sales: '

Group         Men
Sales    85750000
Name: 1, dtype: object

'Lowest Sales: '

Group     Seniors
Sales    84037500
Name: 2, dtype: object
```

2.3 Which state is generating the highest and lowest sales.

[32]:
```
# Which state is producing the highest and lowest sales.

print("Sales Data by State: \n")
state_group = pd.DataFrame(sales_df.groupby('State')['Sales'].sum()).
 ↪sort_values(by=['Sales'],ascending=False)
#display(state_group)
display("Highest Sales: ",state_group.iloc[0,:],"Lowest Sales: ",state_group.
 ↪iloc[-1,:])
```

```
Sales Data by State:


'Highest Sales: '

Sales    105565000
Name:  VIC, dtype: int64

'Lowest Sales: '

Sales    22152500
Name:  WA, dtype: int64
```

2.4 Weekly and Monthly Reports of Highest and Lowest Sales wrt Groups and State.

2.4.1 Weekly Reports of Highest and Lowest Sales Made by various Person Groups

[13]:
```
# Weekly Reports of Sales Data - Sales made by Groups
weekly_group_sales = pd.DataFrame(sales_df.groupby(['Week','Group'],␣
 ↪as_index=False)['Sales'].sum().sort_values(by='Sales', ascending=False))
#display(weekly_group_sales)
```

```
display("Highest Sales: ",weekly_group_sales.iloc[0,:],"Lowest Sales:␣
  ↪",weekly_group_sales.iloc[-1,:])
```

```
'Highest Sales: '
```

```
Week          52
Group        Men
Sales    8337500
Name: 49, dtype: object
```

```
'Lowest Sales: '
```

```
Week          53
Group        Men
Sales    3402500
Name: 53, dtype: object
```

2.4.2 Weekly Reports of Highest and Lowest Sales Made across different States

[14]:
```
# Weekly Reports of Sales Data - Sales made in States
weekly_state_sales = pd.DataFrame(sales_df.groupby(['Week','State'],␣
  ↪as_index=False)['Sales'].sum().sort_values(by='Sales', ascending=False))
#display(weekly_state_sales)
display("Highest Sales: ",weekly_state_sales.iloc[0,:],"Lowest Sales:␣
  ↪",weekly_state_sales.iloc[-1,:])
```

```
'Highest Sales: '
```

```
Week           52
State         VIC
Sales    10345000
Name: 89, dtype: object
```

```
'Lowest Sales: '
```

```
Week         53
State        WA
Sales    925000
Name: 97, dtype: object
```

2.4.3 Monthly Reports of Highest and Lowest Sales Made by various Person Groups

[23]:
```
# Monthly Reports of Sales Data - Sales made by Groups
monthly_group_sales = pd.DataFrame(sales_df.groupby(['Month','Group'],␣
  ↪as_index=False)['Sales'].sum())
#display(monthly_group_sales)
display("Highest Sales: ",monthly_group_sales.iloc[0,:],"Lowest Sales:␣
  ↪",monthly_group_sales.iloc[-1,:])
```

```
'Highest Sales: '
```

```
Month          10
Group        Kids
Sales    28635000
Name: 0, dtype: object
```

```
'Lowest Sales: '
```

```
Month          12
Group       Women
Sales    34375000
Name: 11, dtype: object
```

2.4.4 Monthly Reports of Highest and Lowest Sales Made across different States

[24]:
```
# Monthly Reports of Sales Data - Sales made in State
monthly_state_sales = pd.DataFrame(sales_df.groupby(['Month','State'],␣
 ↪as_index=False)['Sales'].sum().sort_values(by='Sales', ascending=False))
#display(monthly_state_sales)
display("Highest Sales: ",monthly_state_sales.iloc[0,:],"Lowest Sales:␣
 ↪",monthly_state_sales.iloc[-1,:])
```

```
'Highest Sales: '
```

```
Month          12
State         VIC
Sales    42592500
Name: 19, dtype: object
```

```
'Lowest Sales: '
```

```
Month          11
State          WA
Sales    5217500
Name: 13, dtype: object
```

2.4.5 Quarterly Reports of Highest and Lowest Sales Made by various Person Groups

[17]:
```
q4_sales_df = sales_df.copy()

q4_sales_df['Quarter'] = 'Q4'
```

[19]:
```
# Quarterly Reports of Sales Data - Sales made in State
quarterly_group_sales = pd.DataFrame(q4_sales_df.groupby('Group',␣
 ↪as_index=False)['Sales'].sum().sort_values(by='Sales', ascending=False))
#display(quarterly_state_sales)
display("Highest Sales: ",quarterly_group_sales.iloc[0,:],"Lowest Sales:␣
 ↪",quarterly_group_sales.iloc[-1,:])
```

```
'Highest Sales: '
```

```
Group         Men
Sales    85750000
```

```
Name: 1, dtype: object
```

```
'Lowest Sales: '
```

```
Group     Seniors
Sales    84037500
Name: 2, dtype: object
```

2.4.6 Quarterly Reports of Highest and Lowest Sales Made across Different States

[20]:
```
# Quarterly Reports of Sales Data - Sales made in State
quarterly_state_sales = pd.DataFrame(q4_sales_df.groupby('State',␣
 ↪as_index=False)['Sales'].sum().sort_values(by='Sales', ascending=False))
#display(quarterly_state_sales)
display("Highest Sales: ",quarterly_state_sales.iloc[0,:],"Lowest Sales:␣
 ↪",quarterly_state_sales.iloc[-1,:])
```

```
'Highest Sales: '
```

```
State          VIC
Sales    105565000
Name: 5, dtype: object
```

```
'Lowest Sales: '
```

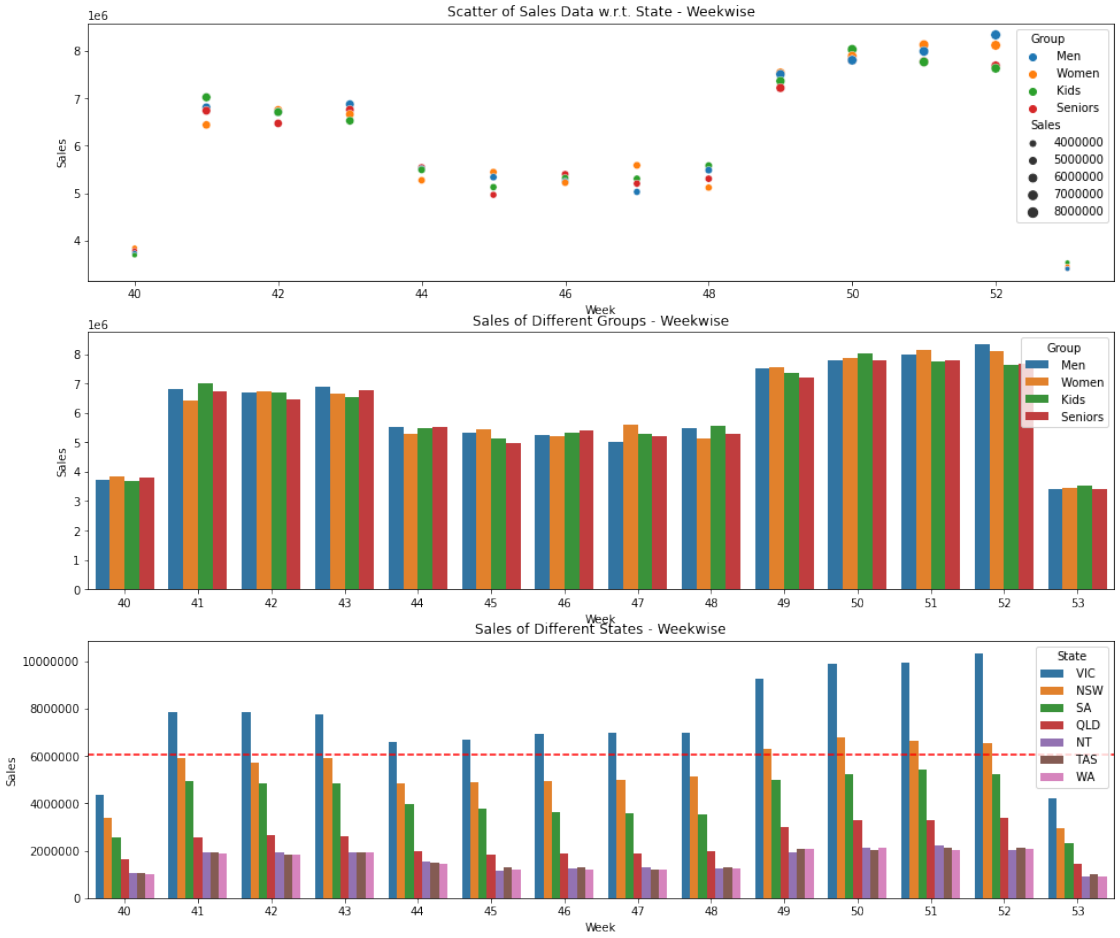
```
State          WA
Sales    22152500
Name: 6, dtype: object
```

2.5 Weekly, Monthly and Quartely Analysis of Above reports?? Using seaborn

2.5.1 Visualization of Weekly Sales Report

[21]:
```
# Using 2 stacked Seaborn Subplots

fig, axes = plt.subplots(3,1, figsize=(16,14))

# Weekly Group All Sales Scatter

sbn.scatterplot(ax=axes[0], data= weekly_group_sales, x = 'Week', y = 'Sales',␣
 ↪hue = 'Group', size='Sales')
plt.axhline(y=weekly_group_sales.Sales.mean(), color='red', ls='--')
axes[0].set_title("Scatter of Sales Data w.r.t. State - Weekwise")

sbn.barplot(ax=axes[1],data=weekly_group_sales, x='Week', y='Sales',␣
 ↪hue='Group')
axes[1].set_title("Sales of Different Groups - Weekwise")
plt.ticklabel_format(style='plain', axis='y')

sbn.barplot(ax=axes[2],data=weekly_state_sales, x='Week', y='Sales',␣
 ↪hue='State')
```

```
axes[2].set_title("Sales of Different States - Weekwise")


plt.show()
```



2.5.1.1 Interpretation:

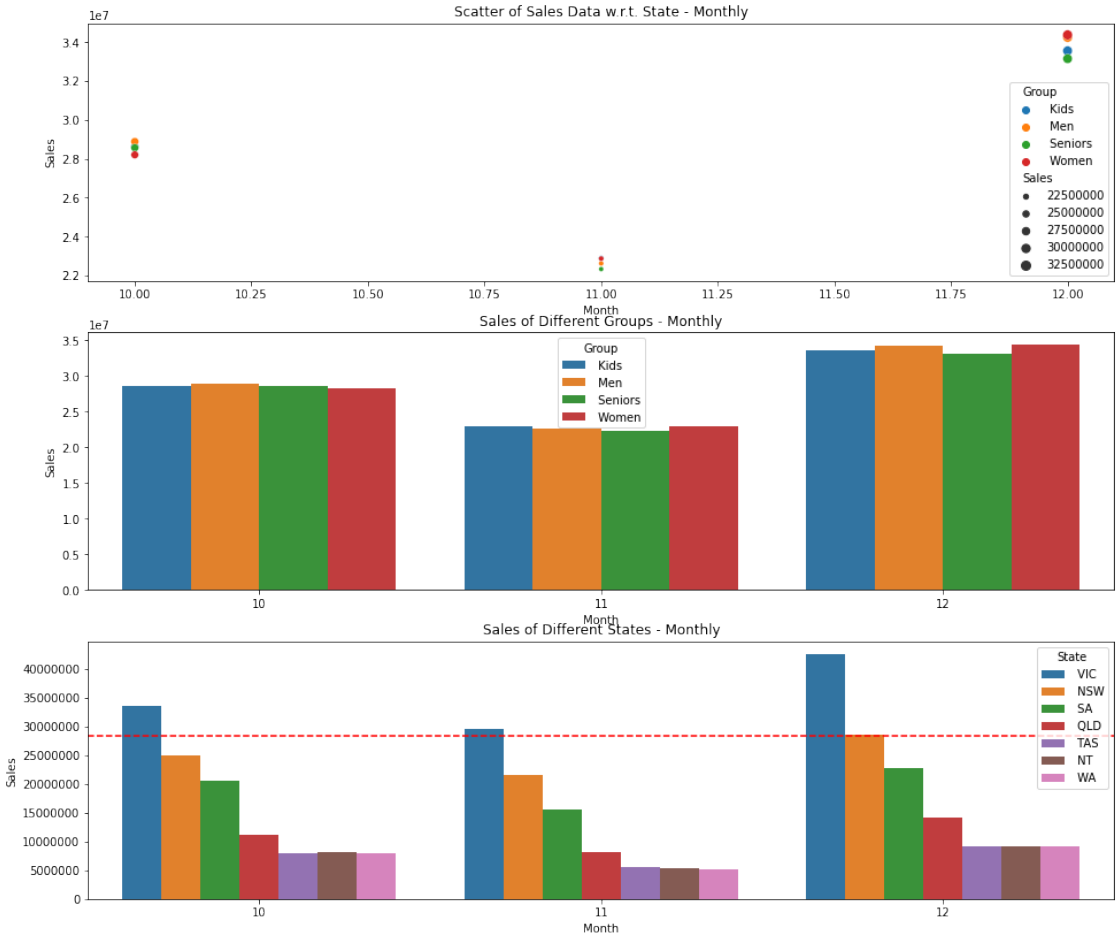
Graph 1: For Each week, the sales made by each group is scattered, higher the point, more the sales. This gives us which group is making highest and which group is making lowest sales.

Graph 2: For Each week, the sales made by each group is shown in bar, higher the bar, more the sales. This gives us which group is making highest and which group is making lowest sales in different weeks.

Graph 3: For Each week, the sales made by each state is shown in bar, higher the bar, more the sales. This gives us which state is making highest and which state is making lowest sales in different weeks.

2.5.2 Visualization of Monthly Sales Report

[26]:
```
# Using 2 stacked Seaborn Subplots

fig, axes = plt.subplots(3,1, figsize=(16,14))

# Weekly Group All Sales Scatter

sbn.scatterplot(ax=axes[0], data= monthly_group_sales, x = 'Month', y =␣
 ↪'Sales', hue = 'Group', size='Sales')
plt.axhline(y=monthly_group_sales.Sales.mean(), color='red', ls='--')
axes[0].set_title("Scatter of Sales Data w.r.t. State - Monthly")

sbn.barplot(ax=axes[1],data=monthly_group_sales, x='Month', y='Sales',␣
 ↪hue='Group')
axes[1].set_title("Sales of Different Groups - Monthly")
plt.ticklabel_format(style='plain', axis='y')

sbn.barplot(ax=axes[2],data=monthly_state_sales, x='Month', y='Sales',␣
 ↪hue='State')
axes[2].set_title("Sales of Different States - Monthly")


plt.show()
```



2.5.2.1 Interpretation:

Graph 1: For Each month, the sales made by each group is scattered, higher the point, more the sales. This gives us which group is making highest and which group is making lowest sales.

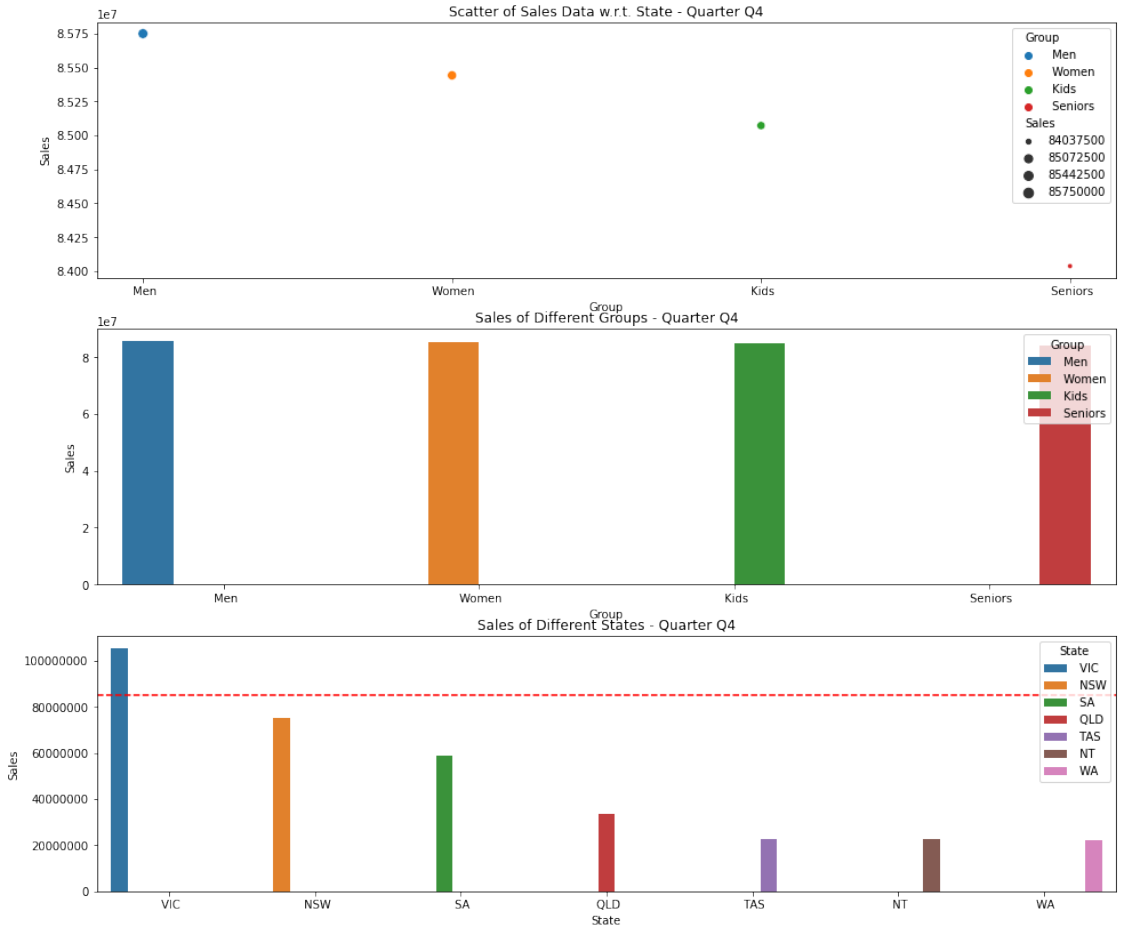
Graph 2: For Each month, the sales made by each group is shown in bar, higher the bar, more the sales. This gives us which group is making highest and which group is making lowest sales in different months.

Graph 3: For Each month, the sales made by each state is shown in bar, higher the bar, more the sales. This gives us which state is making highest and which state is making lowest sales in different months.

2.5.3 Visualization of Quarterly Sales Report

[33]:
```
# Using 2 stacked Seaborn Subplots

fig, axes = plt.subplots(3,1, figsize=(16,14))

# Weekly Group All Sales Scatter
```

```
sbn.scatterplot(ax=axes[0], data= quarterly_group_sales, x = 'Group', y =␣
 ↪'Sales', hue = 'Group', size='Sales')
plt.axhline(y=quarterly_group_sales.Sales.mean(), color='red', ls='--')
axes[0].set_title("Scatter of Sales Data w.r.t. State - Quarter Q4")

sbn.barplot(ax=axes[1],data=quarterly_group_sales, x='Group', y='Sales',␣
 ↪hue='Group')
axes[1].set_title("Sales of Different Groups - Quarter Q4")
plt.ticklabel_format(style='plain', axis='y')

sbn.barplot(ax=axes[2],data=quarterly_state_sales, x='State', y='Sales',␣
 ↪hue='State')
axes[2].set_title("Sales of Different States - Quarter Q4")


plt.show()
```



2.5.3.1 Interpretation:

Graph 1: For Each quarter, the sales made by each group is scattered, higher the point, more the sales. This gives us which group is making highest and which group is making lowest sales.

Graph 2: For Each quarter, the sales made by each group is shown in bar, higher the bar, more the sales. This gives us which group is making highest and which group is making lowest sales in this quarter.

Graph 3: For Each quarter, the sales made by each state is shown in bar, higher the bar, more the sales. This gives us which state is making highest and which state is making lowest sales in this quarter.

0.1 3. Data Visualization (Asked to Skip) –> See 3.2 For Plotly Visualizations

Use appropriate data visualization libraries to build a dashboard for the Head of S&M that includes for the key parameters like

* State-wise sales analysis for different groups (kids, women, men, and seniors)
* Group-wise sales analysis (kids, women, men, and seniors) across different states.
* Time-of-the-day analysis: during which time of the day are sales the highest, and during w
  [This helps S&M teams design programs for increasing sales such as hyper-personalization a
* The dashboard must contain daily, weekly, monthly and quarterly charts.
* (Any visualization library can be used for this purpose. However, since statistical analys

0.1.1 3.1 HSM Dashboard (Using Seaborn)

Box Plot - Statistical Representation

Scatter Plot - Data Distribution

3.1.1 HSM Dashboard - Daily Data Trends and Statistics

[55]:
```
# Subplots for Daily Data Trends
# State Wise
# Group Wise


fig, axes = plt.subplots(2,1, figsize=(16,14))

# Weekly Group All Sales Scatter

sbn.lineplot(ax=axes[0], data = sales_df, x="Date", y="Sales", hue="Group",␣
 ↪sort = True, estimator = "mean")
#plt.axhline(y=sales_df.Sales.mean(), color='red', ls='--')
axes[0].set_title("Daily Trend of Sales wrt Person Groups")

sbn.lineplot(ax=axes[1], data = sales_df, x="Time", y="Sales", hue="Time", sort␣
 ↪= True, estimator = "mean")
```

```
#plt.axhline(y=sales_df.Sales.mean(), color='red', ls='--')
axes[1].set_title("Daily Trend of Sales wrt Time of the Day")

plt.show()
```

0.1.2 3.2 HSM Dashboard (Using Plotly Express & Graph Objects)

Box Plot - Statistical Representation

Scatter Plot - Data Distribution

[ ]:

[32]:
```
# Library import

import plotly.express as px
```

[18]:
```
state_group_min_sales = pd.DataFrame(sales_df.groupby(['State','Group'],␣
 ↪as_index = False)['Sales'].min())
#display(state_group_min_sales)
```

[19]:
```
# State Group Maximum Sales
state_group_max_sales = pd.DataFrame(sales_df.groupby(['State','Group'],␣
 ↪as_index = False)['Sales'].max())
#display(state_group_max_sales)
```

[20]:
```
# State Group Average sales
state_group_avg_sales = pd.DataFrame(sales_df.groupby(['State','Group'],␣
 ↪as_index = False)['Sales'].mean())
#display(state_group_avg_sales)
```

Selecting the Visualization Library (plotly):

0.1.3 Plotly proved detailed and interactive in nature.

The choice between Matplotlib, Seaborn, and Plotly ultimately depends on your project requirements, familiarity with coding, and the type of visualizations you aim to create. Matplotlib offers extensive customization but demands more code, Seaborn simplifies statistical plots with built-in themes, and Plotly excels at creating dynamic and interactive visualizations.

If you prefer precise control over plot aesthetics and are comfortable writing code, Matplotlib might be your choice. If you're aiming for informative statistical plots with less effort, Seaborn could be the go-to. For interactive dashboards and web applications that engage users, Plotly offers a powerful solution.

Building Dashboard using Subplots.

The Subplot has:

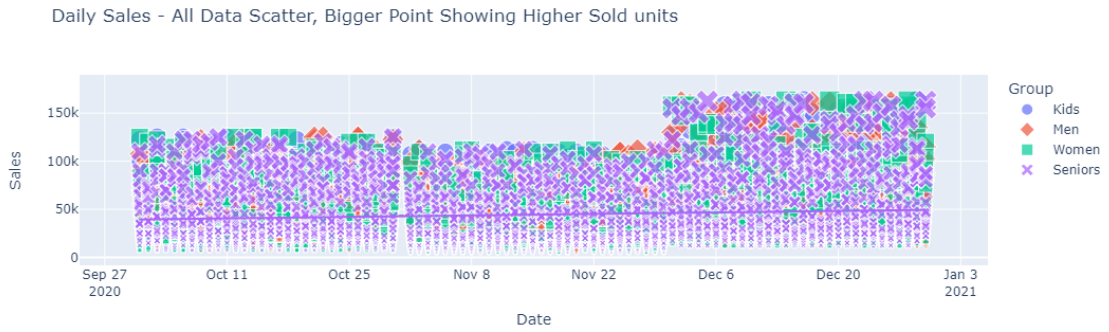
```
1. State wise analysis of different groups.
2. Group wise analysis across different states.
3. Time of the Day analysis - Highest and Lowest Sales.
All of the above in daily, weekly, monthly and quarterly charts.
```

[42]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
import statsmodels
```

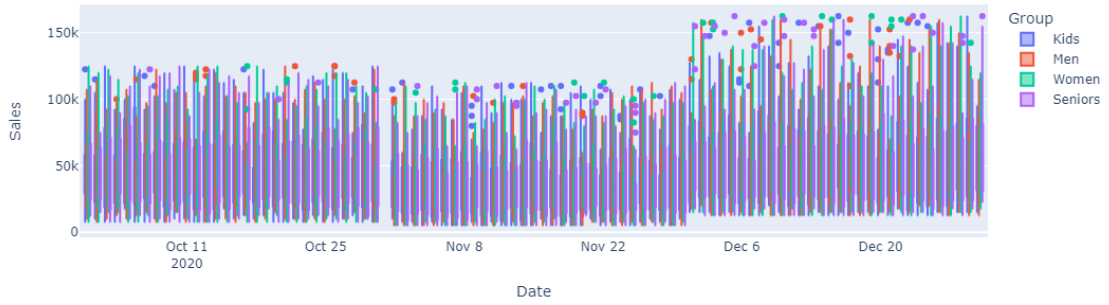
Visualization of Daily Data - All Data Scatter Plot, Bigger Point Showing Higher Sold Units (Grouping by Person Groups)

[43]:
```
# Daily Sales Data All
```

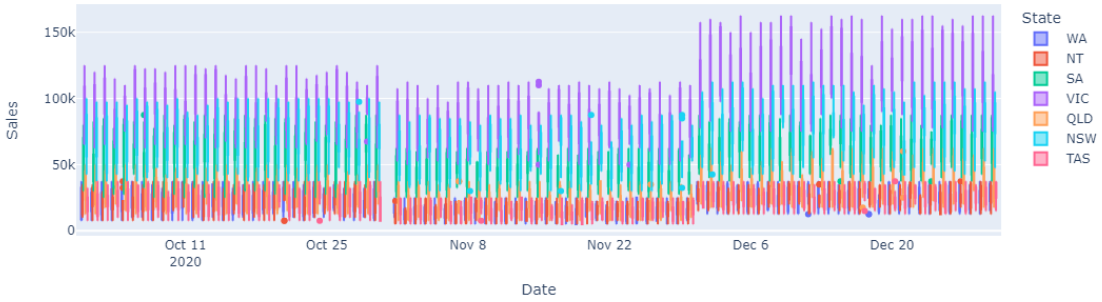
```
fig = px.scatter(sales_df, x = 'Date', y='Sales', color='Group', size='Unit',␣
 ↪symbol='Group', trendline='ols', title='Daily Sales - All Data Scatter,␣
 ↪Bigger Point Showing Higher Sold units')

fig.show()
```



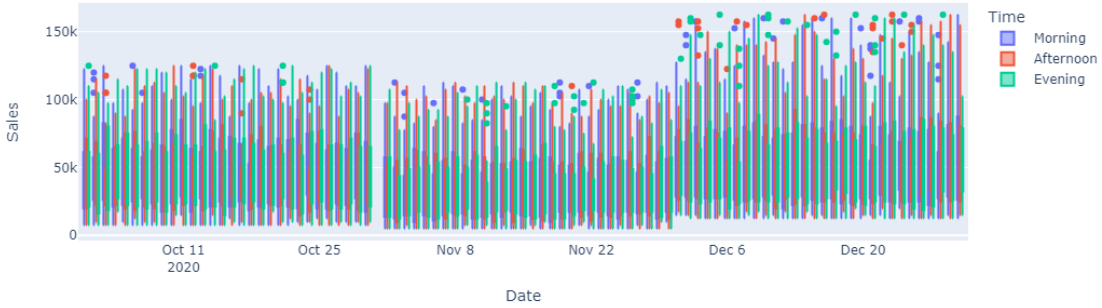
[41]:
```
# Daily Sales Statistics grouped by People Group.
fig = px.box(sales_df, x = 'Date', y='Sales', color='Group')
fig.update_traces(boxmean=True)
fig.show()
```



[58]:
```
# Daily Sales Statistics grouped by State.
fig = px.box(sales_df, x = 'Date', y='Sales', color='State')
fig.update_traces(boxmean=True)
fig.show()
```

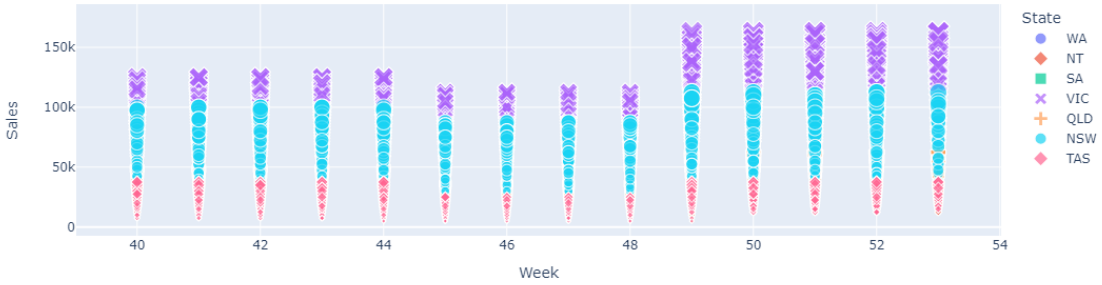


[43]:
```
# Daily Sales Statistics grouped by Time of the Day.
fig = px.box(sales_df, x = 'Date', y='Sales', color='Time')
fig.update_traces(boxmean=True)
fig.show()
```

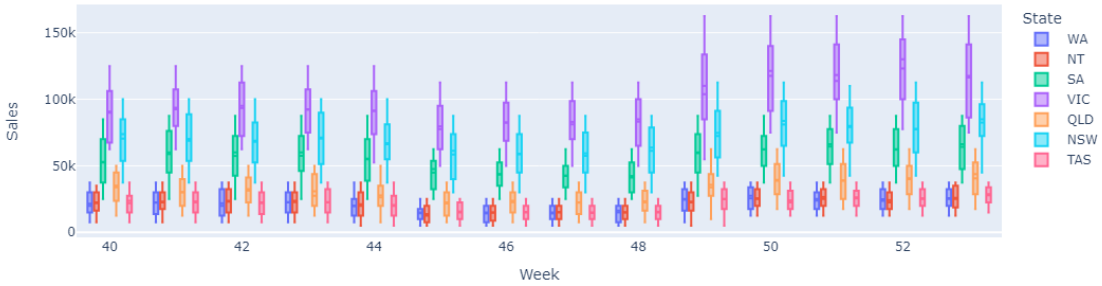


Visualization of Weekly Data (State Wise (among groups), Group Wise (across states) and Time of Day)

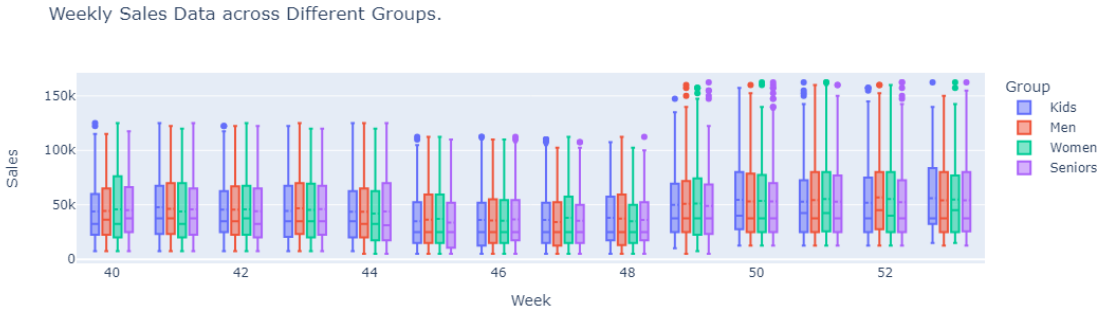
[46]:
```
# Weekly Sales All Data
fig = px.scatter(sales_df, x = 'Week', y='Sales', color='State', size='Unit',␣
 ↪symbol='State')

fig.show()
```



[48]:
```
# All Weekly Data Statitics wrt State
fig = px.box(sales_df, x = 'Week', y='Sales', color='State')
fig.update_traces(boxmean=True)
fig.show()
```



[50]:
```
# All Weekly Data Statitics wrt Group
fig = px.box(sales_df, x = 'Week', y='Sales', color='Group', title= "Weekly␣
↪Sales Data across Different Groups.")
fig.update_traces(boxmean=True)
fig.show()
```



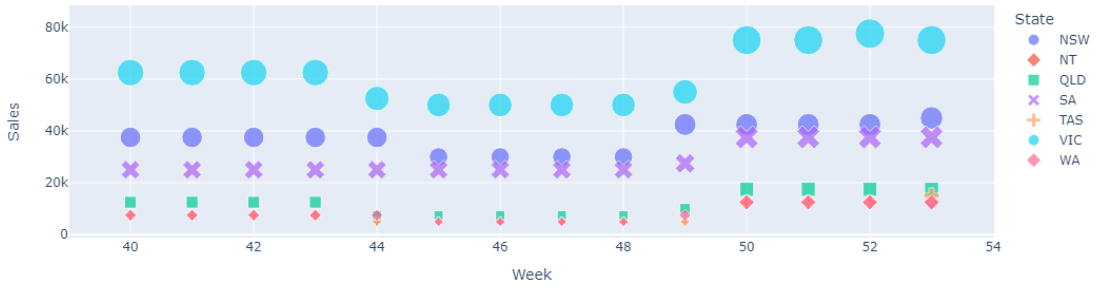
Data Frames for Line Graphs

[51]:
```
# Weekly Data Holding DataFrames

weekly_state_min_sales = pd.DataFrame(sales_df.groupby(['Week','State'],␣
 ↪as_index= False)['Sales'].min())
weekly_state_avg_sales = pd.DataFrame(sales_df.groupby(['Week','State'],␣
 ↪as_index= False)['Sales'].mean())
weekly_state_max_sales = pd.DataFrame(sales_df.groupby(['Week','State'],␣
 ↪as_index= False)['Sales'].max())

weekly_group_min_sales = pd.DataFrame(sales_df.groupby(['Week','Group'],␣
 ↪as_index= False)['Sales'].min())
weekly_group_avg_sales = pd.DataFrame(sales_df.groupby(['Week','Group'],␣
 ↪as_index= False)['Sales'].mean())
weekly_group_max_sales = pd.DataFrame(sales_df.groupby(['Week','Group'],␣
 ↪as_index= False)['Sales'].max())

weekly_tod_min_sales = pd.DataFrame(sales_df.groupby(['Week','Time'], as_index=␣
 ↪False)['Sales'].min())
weekly_tod_avg_sales = pd.DataFrame(sales_df.groupby(['Week','Time'], as_index=␣
 ↪False)['Sales'].mean())
weekly_tod_max_sales = pd.DataFrame(sales_df.groupby(['Week','Time'], as_index=␣
 ↪False)['Sales'].max())
```

[52]:
```
# Creating Aggregated DataFrame for Weekly State Data
weekly_state_sales_data = weekly_state_min_sales.copy()
weekly_state_sales_data.rename(columns = {'Sales':'MinSales'}, inplace=True)
weekly_state_sales_data['AvgSales'] = weekly_state_avg_sales['Sales']
weekly_state_sales_data['MaxSales'] = weekly_state_max_sales['Sales']

# Creating Aggregated DataFrame for Weekly Group Data
weekly_group_sales_data = weekly_group_min_sales.copy()
```

```
weekly_group_sales_data.rename(columns = {'Sales':'MinSales'}, inplace=True)
weekly_group_sales_data['AvgSales'] = weekly_group_avg_sales['Sales']
weekly_group_sales_data['MaxSales'] = weekly_group_max_sales['Sales']

# Creating Aggregated DataFrame for Weekly TOD Data
weekly_tod_sales_data = weekly_tod_min_sales.copy()
weekly_tod_sales_data.rename(columns = {'Sales':'MinSales'}, inplace=True)
weekly_tod_sales_data['AvgSales'] = weekly_tod_avg_sales['Sales']
weekly_tod_sales_data['MaxSales'] = weekly_tod_max_sales['Sales']
```

[57]:
```
# Weekly Minimum Sales Data across State.
fig = px.scatter(weekly_state_min_sales, x = 'Week', y='Sales', color='State',␣
 ↪size='Sales', symbol='State')
fig.show()
```



Combined Subplot Weekly State wise data.

[ ]:
```
fig = make_subplots(rows=2, cols=1)

fig.add_trace(go.Bar(x=monthly_group_sales['Month'],␣
 ↪y=monthly_group_sales['Sales'], color = monthly_group_sales['Group'],␣
 ↪barmode = 'group'),
              row=1, col=1)
fig.update_xaxes(title_text="Month", row=1, col=1)
fig.update_yaxes(title_text="Sales by Group", row=1, col=1)

fig.add_trace(go.Bar(x=monthly_state_sales['State'],␣
 ↪y=monthly_state_sales['Sales']),
              row=2, col=1)
fig.update_xaxes(title_text="Month", row=2, col=1)
fig.update_yaxes(title_text="Sales Made by States", row=2, col=1)

fig.update_layout(height=600, width=600, title_text="Monthly Data")
```

```
fig.show()
```

[ ]:
```
fig = make_subplots(rows = 3, cols=1)

fig.add_trace(go.scatter)
```

[225]:
```
# Weekly Minimum Sales Data in Group
fig = px.scatter(weekly_group_min_sales, x = 'Week', y='Sales', color='Group',␣
 ↪size='Sales', symbol='Group', trendline='ols')

fig.show()
```

[227]:
```
# Weekly Minimum Sales Data in Group
fig = px.scatter(weekly_tod_min_sales, x = 'Week', y='Sales', color='Time',␣
 ↪size='Sales', symbol='Time', trendline='ols')

fig.show()
```

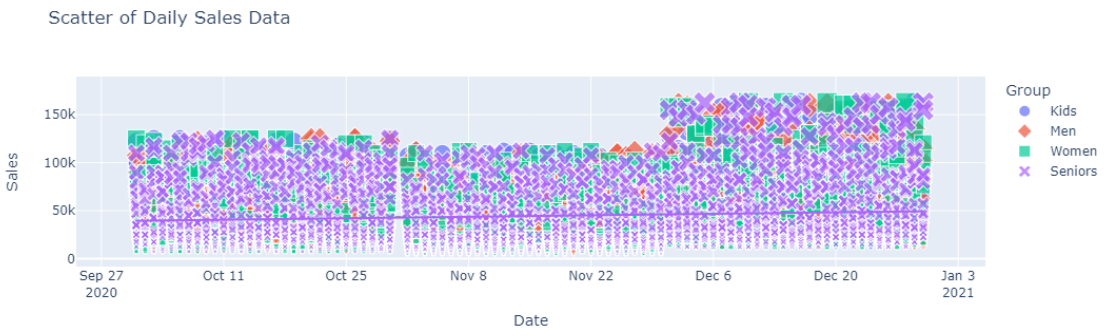
[ ]:
```

```

Sales and Marketing Head - Dashboard (wip)

Drawing Individual Plots (Individual Scatter and Box Plots)

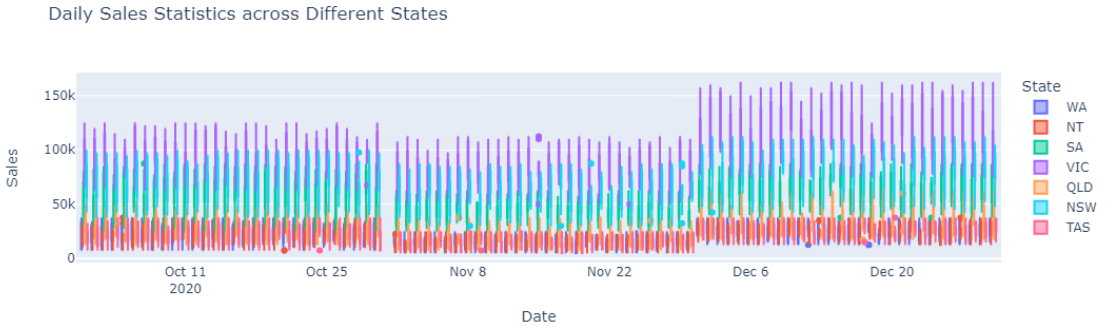
Daily Sales Data Visualization

[60]:
```
# Daily Sales Data All
fig = px.scatter(sales_df, x = 'Date', y='Sales', color='Group', size='Unit',␣
 ↪symbol='Group', trendline='ols', title="Scatter of Daily Sales Data")

fig.show()
```



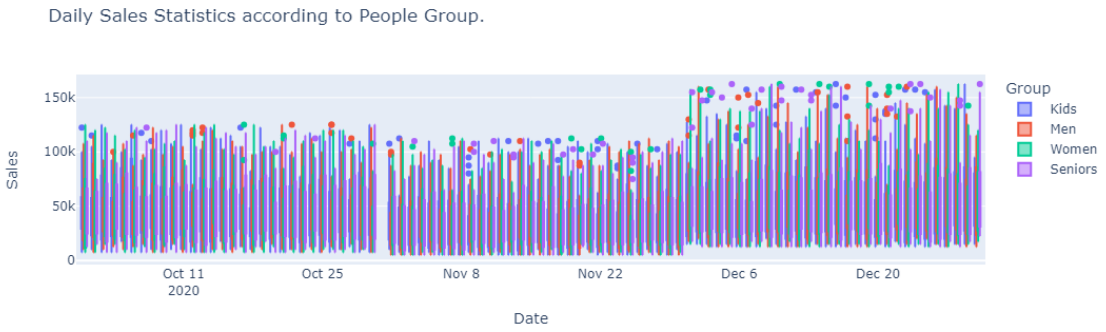
Daily Sales Statistics across Different States

[62]:
```
# Daily Sales Statistics across Different States.
fig = px.box(sales_df, x = 'Date', y='Sales', color='State', title="Daily Sales␣
 ↪Statistics across Different States")
fig.update_traces(boxmean=True)
fig.show()
```



Daily Sales Statistics according to People Group.

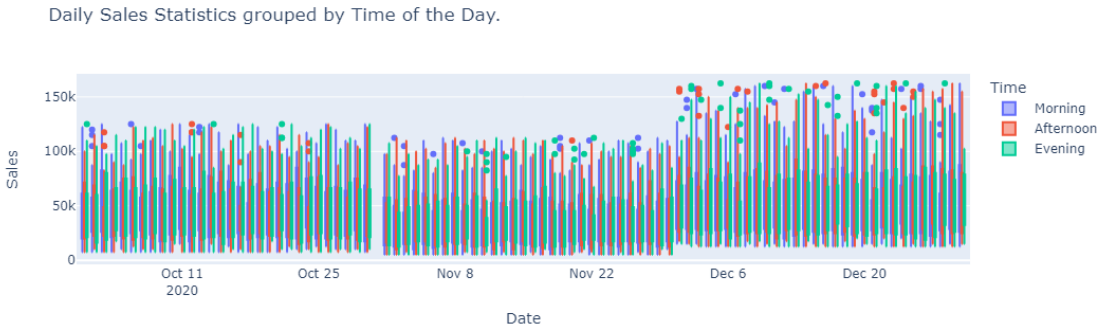
[63]:
```
# Daily Sales Statistics according to People Group.
fig = px.box(sales_df, x = 'Date', y='Sales', color='Group', title="Daily Sales␣
 ↪Statistics according to People Group.")
fig.update_traces(boxmean=True)
fig.show()
```



Daily Sales Statistics grouped by Time of the Day.

[64]:
```
# Daily Sales Statistics grouped by Time of the Day.
fig = px.box(sales_df, x = 'Date', y='Sales', color='Time', title="Daily Sales␣
 ↪Statistics grouped by Time of the Day.")
fig.update_traces(boxmean=True)
```

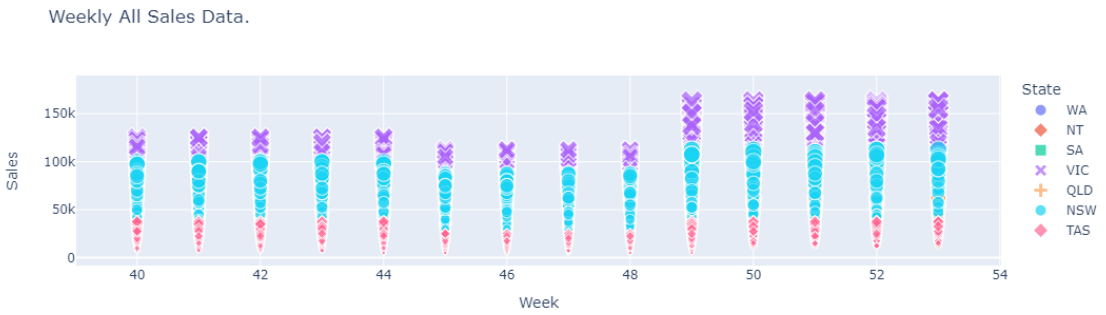
```
fig.show()
```



0.1.4 Weekly Sales Data Visualization

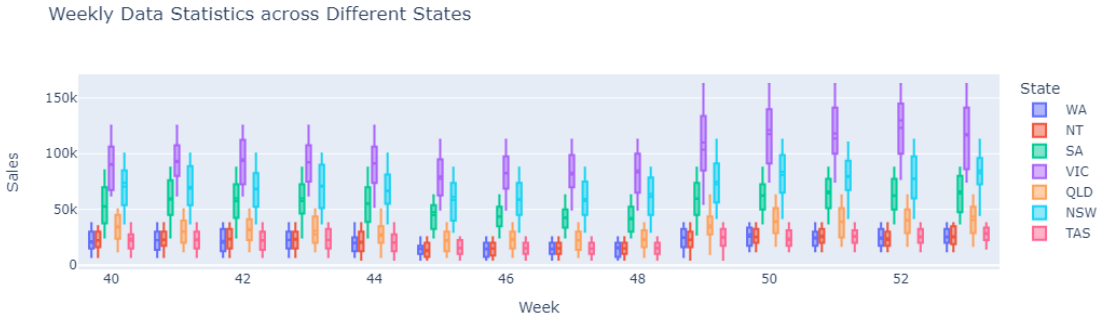
Weekly All Sales Data Scatter

[66]:
```
# Weekly Sales All Data
fig = px.scatter(sales_df, x = 'Week', y='Sales', color='State', size='Unit',␣
 ↪symbol='State', title="Weekly All Sales Data.")

fig.show()
```



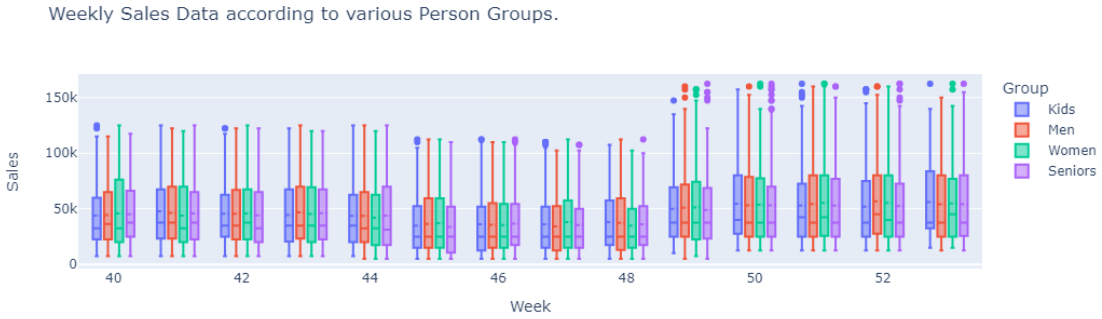
Weekly Data Statistics across Different States (Box Plot)

[67]:
```
# Weekly Data Statitics wrt State
fig = px.box(sales_df, x = 'Week', y='Sales', color='State', title="Weekly Data␣
 ↪Statistics across Different States")
fig.update_traces(boxmean=True)
fig.show()
```



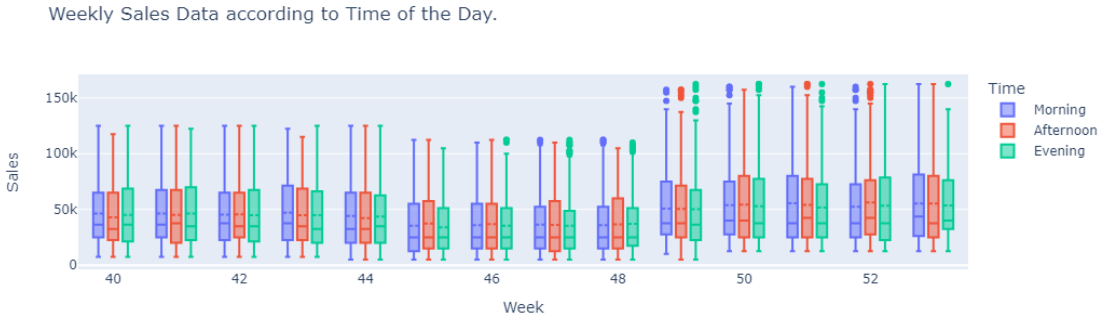
Weekly Data Statistics according to various Person Groups.

[68]:
```
# All Weekly Data Statitics wrt Group
fig = px.box(sales_df, x = 'Week', y='Sales', color='Group', title= "Weekly␣
 ↪Sales Data according to various Person Groups.")
fig.update_traces(boxmean=True)
fig.show()
```



Weekly Sales Data Statistics according to Time of The Day.

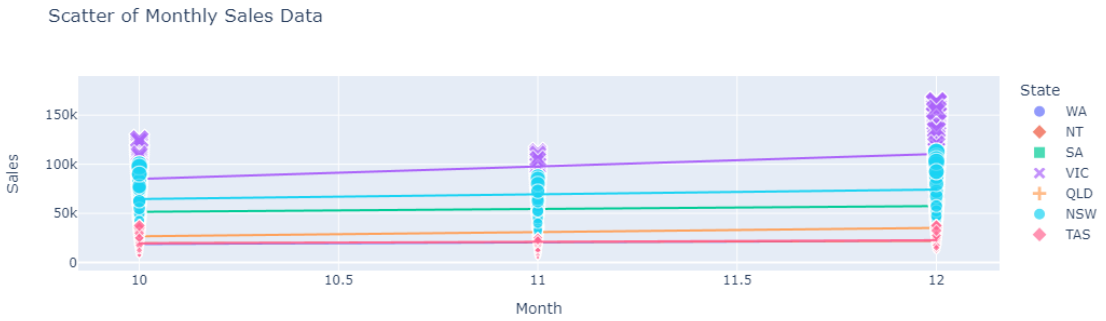
[69]:
```
# All Weekly Data Statitics wrt Group
fig = px.box(sales_df, x = 'Week', y='Sales', color='Time', title= "Weekly␣
 ↪Sales Data according to Time of the Day.")
fig.update_traces(boxmean=True)
fig.show()
```



0.1.5 Monthly Sales Data Visualization

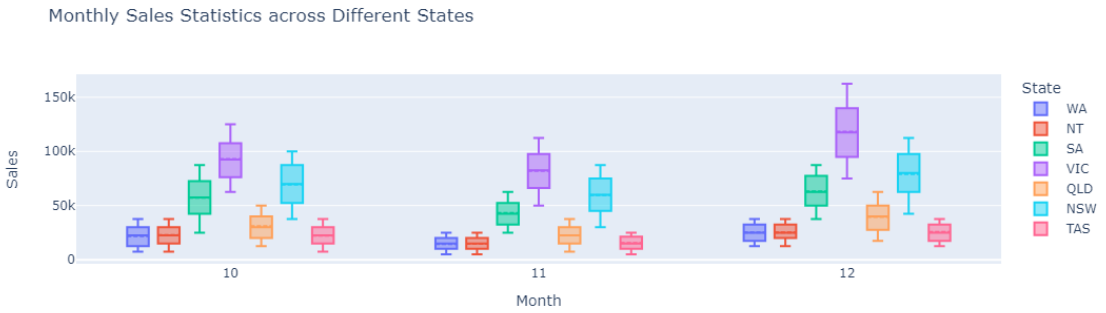
Scatter of Monthly Sales Data

[71]:
```
# Monthly Sales Data All
fig = px.scatter(sales_df, x = 'Month', y='Sales', color='State', size='Sales',␣
 ↪symbol='State', trendline='ols', title="Scatter of Monthly Sales Data")

fig.show()
```



Monthly Sales Statistics across Different States

[72]:
```
# Monthly Sales Statistics across Different States.
fig = px.box(sales_df, x = 'Month', y='Sales', color='State', title="Monthly␣
 ↪Sales Statistics across Different States")
fig.update_traces(boxmean=True)
fig.show()
```

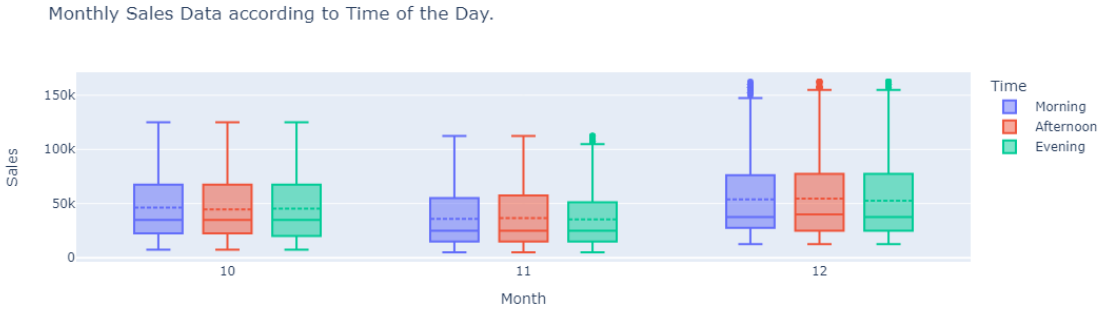


Monthly Sales Statistics according to People Group

[73]:
```
# Monthly Sales Statistics according to People Group
fig = px.box(sales_df, x = 'Month', y='Sales', color='Group', title="Monthly␣
 ↪Sales Statistics according to People Group.")
fig.update_traces(boxmean=True)
fig.show()
```

Monthly Sales Statistics according to Time of The Day.

[74]:
```
fig = px.box(sales_df, x = 'Month', y='Sales', color='Time', title= "Monthly␣
 ↪Sales Data according to Time of the Day.")
fig.update_traces(boxmean=True)
fig.show()
```

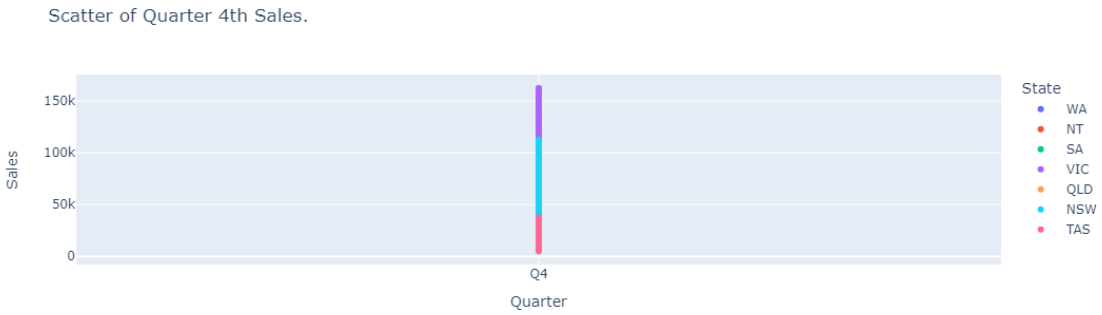


0.1.6 Quaterly Sales Data Visualization

Scatter of Quarterly Sales Data

[75]:
```
q4_sales_df = sales_df.copy()

q4_sales_df['Quarter'] = 'Q4'
```

[76]:
```
fig = px.scatter(q4_sales_df, x = 'Quarter', y='Sales', color='State',␣
↪title="Scatter of Quarter 4th Sales.")

fig.show()
```

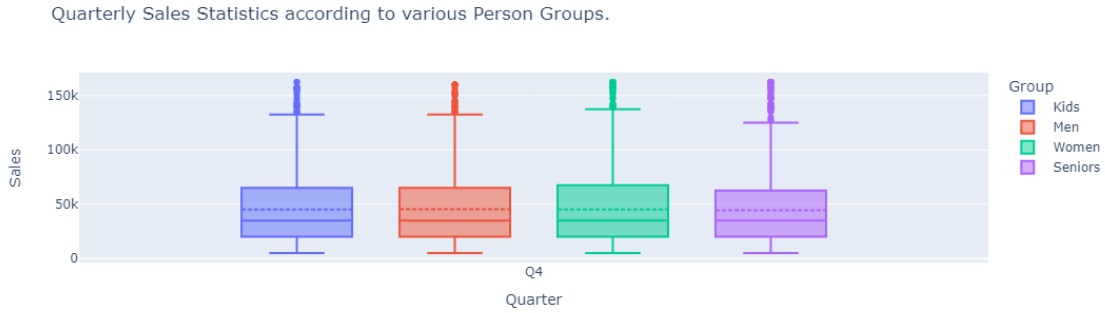


Quarterly Sales Statistics across Different States

[77]:
```
fig = px.box(q4_sales_df, x = 'Quarter', y='Sales', color='State',␣
↪title="Quarterly Sales Statistics across Different States.")
fig.update_traces(boxmean=True)
fig.show()
```

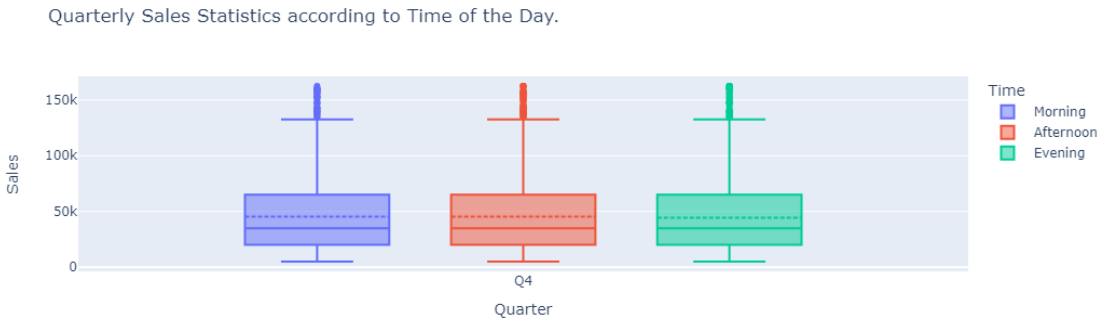
Quarterly Sales Statistics according to various Person Groups.

[78]:
```
fig = px.box(q4_sales_df, x = 'Quarter', y='Sales', color='Group',␣
 ↪title="Quarterly Sales Statistics according to various Person Groups.")
fig.update_traces(boxmean=True)
fig.show()
```



Quarterly Sales Statistics according to Time of The Day.

[79]:
```
fig = px.box(q4_sales_df, x = 'Quarter', y='Sales', color='Time',␣
 ↪title="Quarterly Sales Statistics according to Time of the Day.")
fig.update_traces(boxmean=True)
fig.show()
```



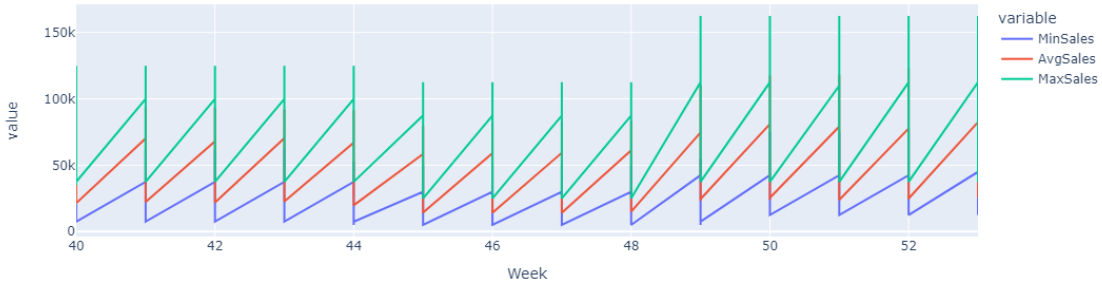
Detailed Stacked Panel. (Using Plotly Graph Objects Subplots -> Overly Complex, Collapsed)

[272]:
```
# #Initializing figure and assigning plot matrix map

# fig = make_subplots(
#     rows=6, cols=2,
#     specs=[
#             [{"type":"scatter", "colspan":2 }, None],
#           [{"type":"box"},{"type":"box"}],
#             [{"type":"scatter", "colspan":2}, None],
#           [{"type":"box"},{"type":"box"}],
#             [{"type":"scatter", "colspan":2}, None],
#           [{"type":"box"},{"type":"box"}]
#     ])

# # Adding Subplots

# #First Full Subplot, Overall Daily Sales
# fig.add_trace(go.Scatter(x=sales_df['Date'], y=sales_df['Sales']), row=1,␣
 ↪col=1)

# # Second Left Subplot, Daily Sales across Different States
# fig.add_trace(go.Box(x=sales_df['Date'], y=sales_df['Sales'], boxmean='sd'),␣
 ↪row=2, col=1)

# # Second Right Subplot, Daily Sales among Different Groups
# fig.add_trace(go.Box(x=sales_df['Date'], y=sales_df['Sales'], boxmean='sd'),␣
 ↪row=2, col=2)

# fig.update_layout(height=1080, width=2160, title_text="Side By Side Subplots")
# fig.show()
```

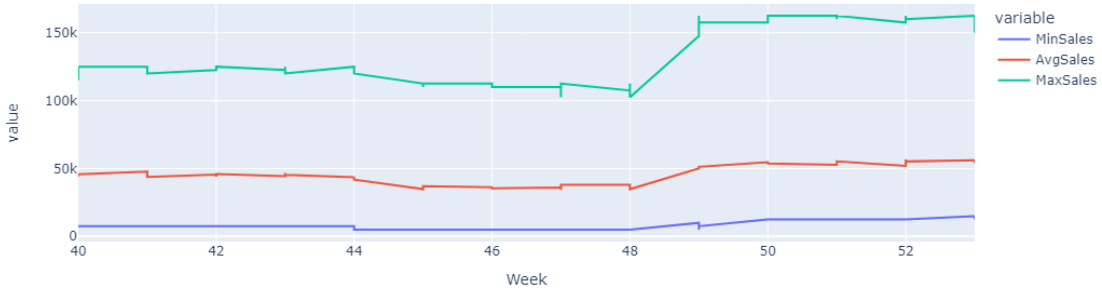
0.1.7 Agregated Line Plots for Analysis

Weekly Sales Data Visualization in Line Graphs

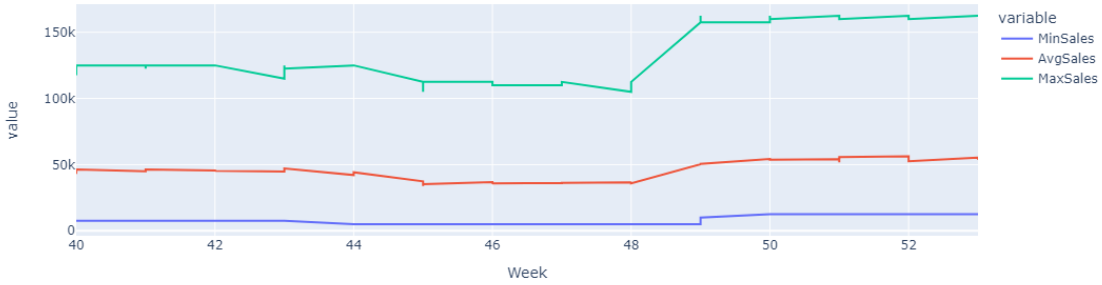
[53]:
```
fig = px.line(weekly_state_sales_data, x = 'Week', y =␣
 ↪['MinSales','AvgSales','MaxSales'])
#fig = px.line(sales_df, x = 'Week', y = 'Sales', color='State')

fig.show()
```



[54]:
```
fig = px.line(weekly_group_sales_data, x = 'Week', y =␣
 ↪['MinSales','AvgSales','MaxSales'])

fig.show()
```



[55]:
```
fig = px.line(weekly_tod_sales_data, x = 'Week', y =␣
 ↪['MinSales','AvgSales','MaxSales'])

fig.show()
```



Monthly Sales Data Visualization in Line Graphs

[ ]:

[ ]:

Quarterly Sales Data Visualization in Line Graphs

[ ]:

[ ]:

[ ]:

[61]:
```
import plotly.io as pio
pio.renderers.default = "notebook_connected"
```