# AAI_Sales_new

December 14, 2025

[1]:
```
import pandas as pd
df = pd.read_csv("AusApparalSales4thQrt2020.csv")
df.head()
```

[1]:

| | 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 |

[2]:
```
print("Missing Values in each column: ")
print(df.isna().sum())
print("\n Non-missing values in each column: ")
print(df.notna().sum())
```

```
Missing Values in each column:
Date     0
Time     0
State    0
Group    0
Unit     0
Sales    0
dtype: int64

 Non-missing values in each column:
Date     7560
Time     7560
State    7560
Group    7560
Unit     7560
Sales    7560
dtype: int64
```

[3]:
```
df.dropna(how='all',inplace=True)
df.info()
df['Sales'] = df['Sales'].fillna(df['Sales'].mean())
df['Sales']
```

```
<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   object
 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: int64(2), object(4)
memory usage: 354.5+ KB
```

[3]:
```
0       20000
1       20000
2       10000
3       37500
4        7500
        ...
7555    35000
7556    37500
7557    37500
7558    27500
7559    32500
Name: Sales, Length: 7560, dtype: int64
```

[4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['Sales','Unit']]=scaler.fit_transform(df[['Sales','Unit']])
df[['Sales','Unit']]
```

[4]:

| | Sales | Unit |
|---|---|---|
| 0 | 0.095238 | 0.095238 |
| 1 | 0.095238 | 0.095238 |
| 2 | 0.031746 | 0.031746 |
| 3 | 0.206349 | 0.206349 |
| 4 | 0.015873 | 0.015873 |
| ... | ... | ... |
| 7555 | 0.190476 | 0.190476 |
| 7556 | 0.206349 | 0.206349 |
| 7557 | 0.206349 | 0.206349 |
| 7558 | 0.142857 | 0.142857 |
| 7559 | 0.174603 | 0.174603 |

[7560 rows x 2 columns]

[5]:
```
#Taks 1-d
State_Sales= df.groupby("State")["Sales"].sum().reset_index()
print(State_Sales)
State_Sales.sort_values(by="Sales", ascending=False)
```

```
  State       Sales
0   NSW  441.714286
1    NT  109.079365
2   QLD  177.888889
3    SA  339.412698
4   TAS  110.222222
5   VIC  635.968254
6    WA  106.365079
```

[5]:

| | State | Sales |
|---|---|---|
| 5 | VIC | 635.968254 |
| 0 | NSW | 441.714286 |
| 3 | SA | 339.412698 |
| 2 | QLD | 177.888889 |
| 4 | TAS | 110.222222 |
| 1 | NT | 109.079365 |
| 6 | WA | 106.365079 |

[6]:
```
#task 2-a
print(df[["Sales", "Unit"]].describe())
print(df[["Sales", "Unit"]].mode())
```

```
             Sales         Unit
count  7560.000000  7560.000000
mean      0.254054     0.254054
std       0.204784     0.204784
min       0.000000     0.000000
25%       0.095238     0.095238
50%       0.190476     0.190476
75%       0.380952     0.380952
max       1.000000     1.000000
      Sales      Unit
0  0.111111  0.111111
```

[8]:
```
#Task 2-b and 2-c
top_sales_group = df.groupby("Group")["Sales"].sum().idxmax()
bottom_sales_group = df.groupby("Group")["Sales"].sum().idxmin()
print(f"Highest Sales by Group: {top_sales_group}")
print(f"Lowest Sales by Group: {bottom_sales_group}")
```

```
Highest Sales by Group:  Men
Lowest Sales by Group:  Seniors
```

[11]:
```
# Task 2-d
df['date'] = pd.to_datetime(df['Date'])
df['Week'] = df['date'].dt.isocalendar().week
df['Month'] = df['date'].dt.month
df['Quarter'] = df['date'].dt.quarter

df.head()
```

[11]:

| | Date | Time | State | Group | Unit | Sales | date | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 1-Oct-2020 | Morning | WA | Kids | 0.095238 | 0.095238 | 2020-10-01 | |
| 1 | 1-Oct-2020 | Morning | WA | Men | 0.095238 | 0.095238 | 2020-10-01 | |
| 2 | 1-Oct-2020 | Morning | WA | Women | 0.031746 | 0.031746 | 2020-10-01 | |
| 3 | 1-Oct-2020 | Morning | WA | Seniors | 0.206349 | 0.206349 | 2020-10-01 | |
| 4 | 1-Oct-2020 | Afternoon | WA | Kids | 0.015873 | 0.015873 | 2020-10-01 | |

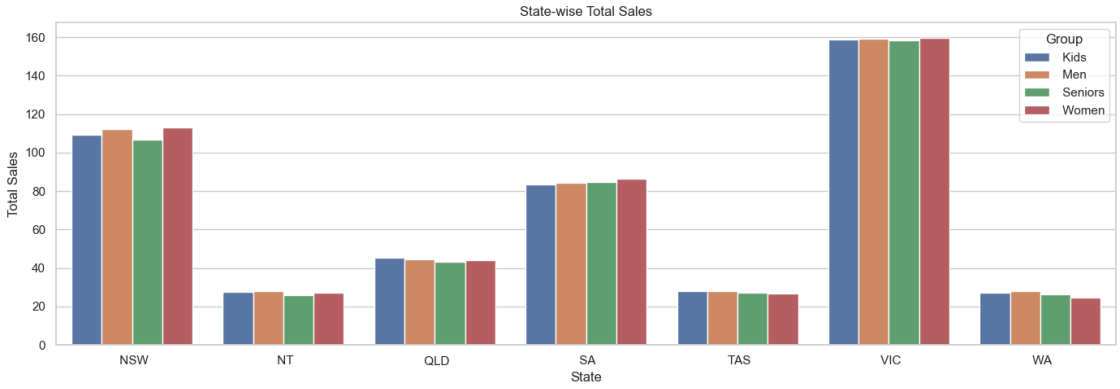
| | week | month | quarter | Week | Month | Quarter |
|---|---|---|---|---|---|---|
| 0 | 40 | 10 | 4 | 40 | 10 | 4 |
| 1 | 40 | 10 | 4 | 40 | 10 | 4 |
| 2 | 40 | 10 | 4 | 40 | 10 | 4 |
| 3 | 40 | 10 | 4 | 40 | 10 | 4 |
| 4 | 40 | 10 | 4 | 40 | 10 | 4 |

[12]:
```
#weekly sales trend
weekly_sales = df.groupby('Week')['Sales'].sum().reset_index()
print(weekly_sales)

#monthly sales trend
monthly_sales = df.groupby('Month')['Sales'].sum().reset_index()
print(monthly_sales)

#quarterly sales trend
quarterly_sales = df.groupby('Quarter')['Sales'].sum().reset_index()
print(quarterly_sales)
```

| | Week | Sales |
|---|---|---|
| 0 | 40 | 84.857143 |
| 1 | 41 | 152.777778 |
| 2 | 42 | 150.476190 |
| 3 | 43 | 151.587302 |
| 4 | 44 | 122.460317 |
| 5 | 45 | 113.809524 |
| 6 | 46 | 115.761905 |
| 7 | 47 | 115.380952 |
| 8 | 48 | 117.698413 |
| 9 | 49 | 169.412698 |
| 10 | 50 | 181.492063 |
| 11 | 51 | 182.317460 |
| 12 | 52 | 183.047619 |

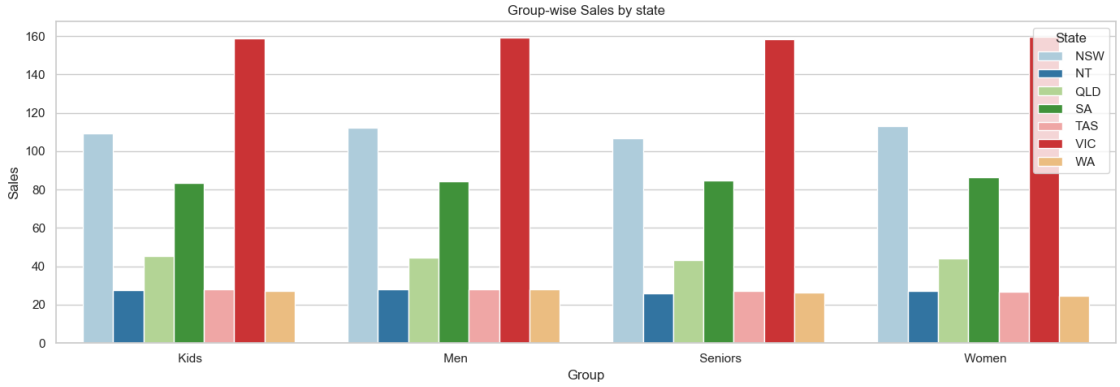
```
13     53   79.571429
   Month       Sales
0     10  645.650794
1     11  495.761905
2     12  779.238095
   Quarter        Sales
0        4  1920.650794
```

[17]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

sns.set_theme(style="whitegrid")

#Task 3-a-1
# Group-wise sales bar chart
State_group_Sales = df.groupby(['State','Group'])['Sales'].sum().reset_index()
plt.figure(figsize=(14,5))
sns.barplot(data=State_group_Sales, x='State', y='Sales', hue='Group')
plt.ylabel("Total Sales")
plt.title("State-wise Total Sales")
plt.tight_layout()
plt.show()
```

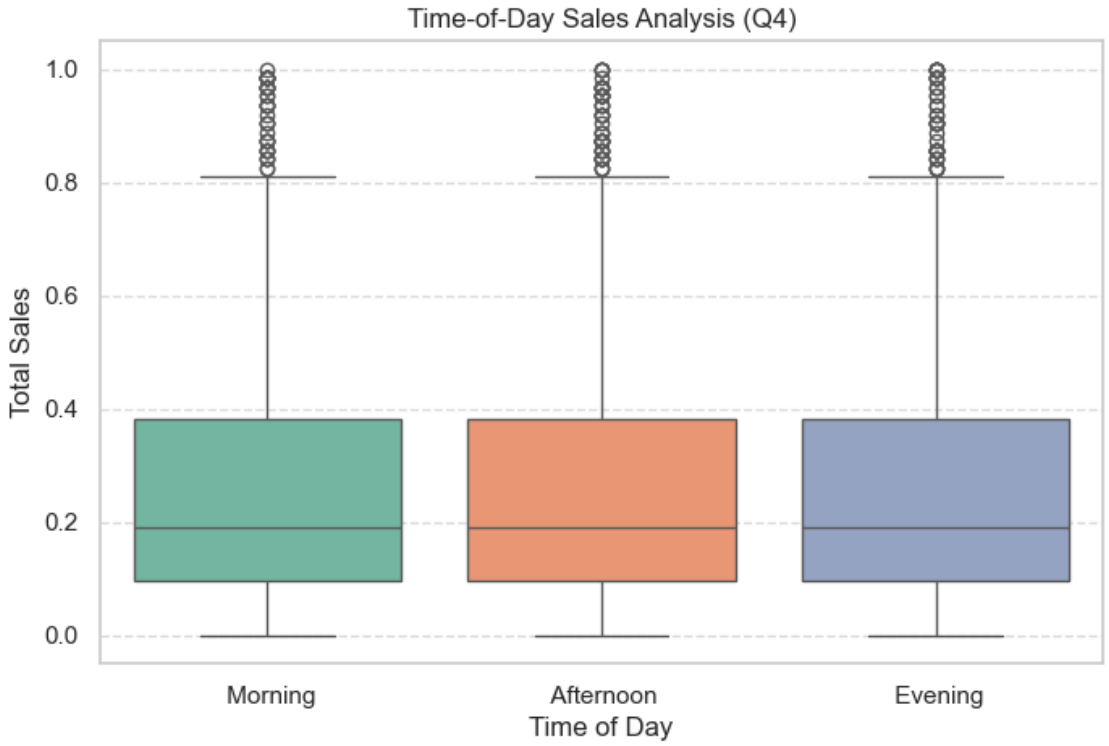



[26]:
```
#Task 3-a-2
# Group-wise sales by State bar chart
group_state_Sales = df.groupby(['Group','State'])['Sales'].sum().reset_index()
plt.figure(figsize=(14,5))
sns.barplot(data=group_state_Sales, x='Group', y='Sales',␣
 ↪hue='State',palette='Paired')
plt.ylabel("Sales")
plt.title("Group-wise Sales by state")
```

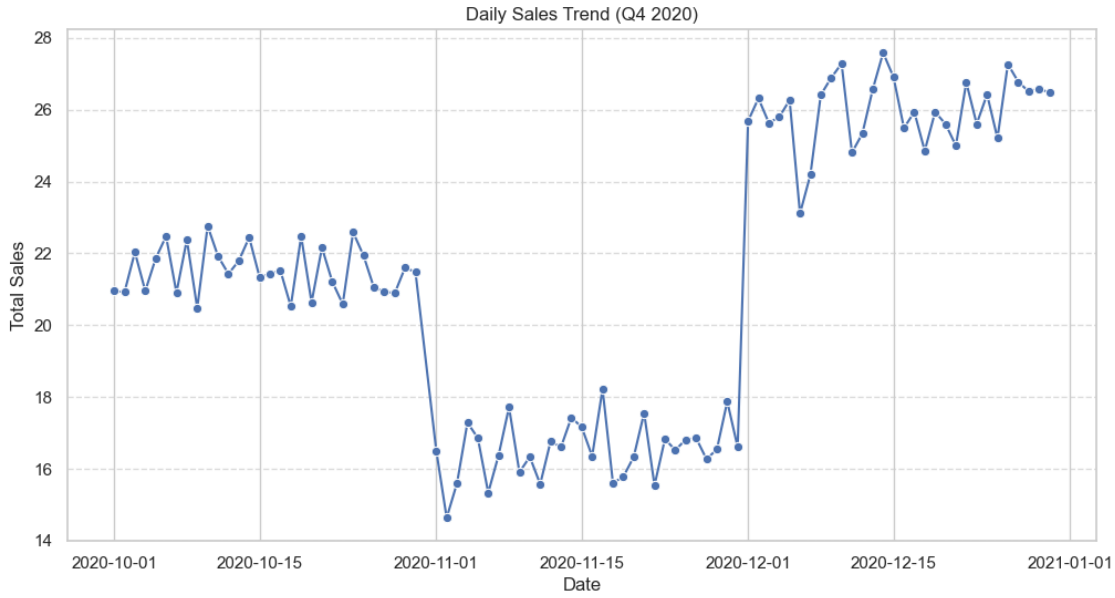
```
plt.tight_layout()
plt.show()
```

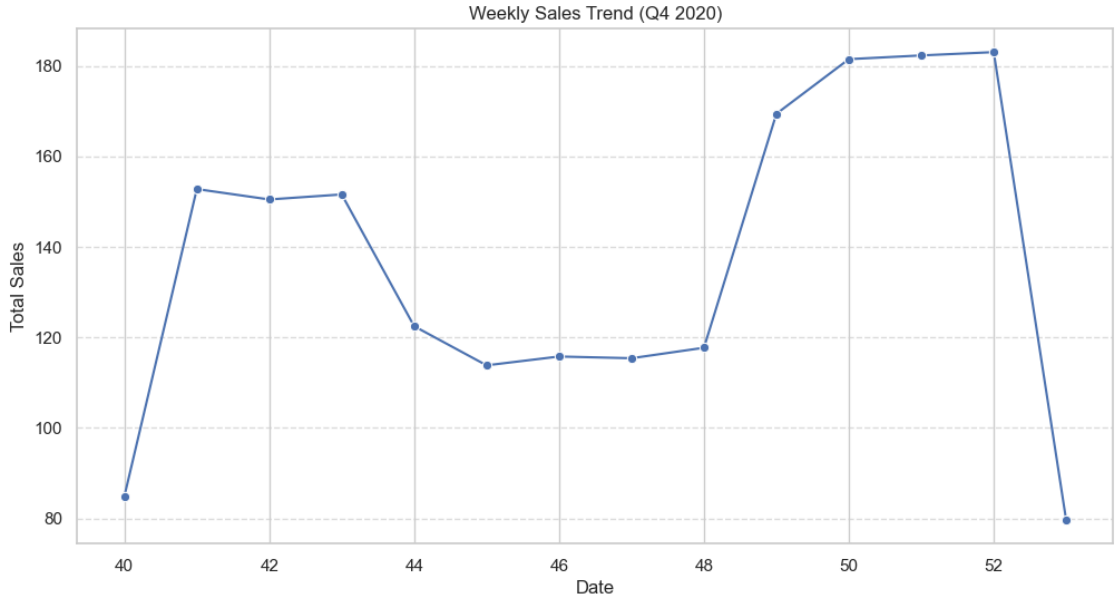



[25]:
```
#Task 3-a-3
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x="Time", y="Sales", palette="Set2",hue="Time")

plt.title("Time-of-Day Sales Analysis (Q4)")
plt.xlabel("Time of Day")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle="--", alpha=0.7)
plt.show()
```



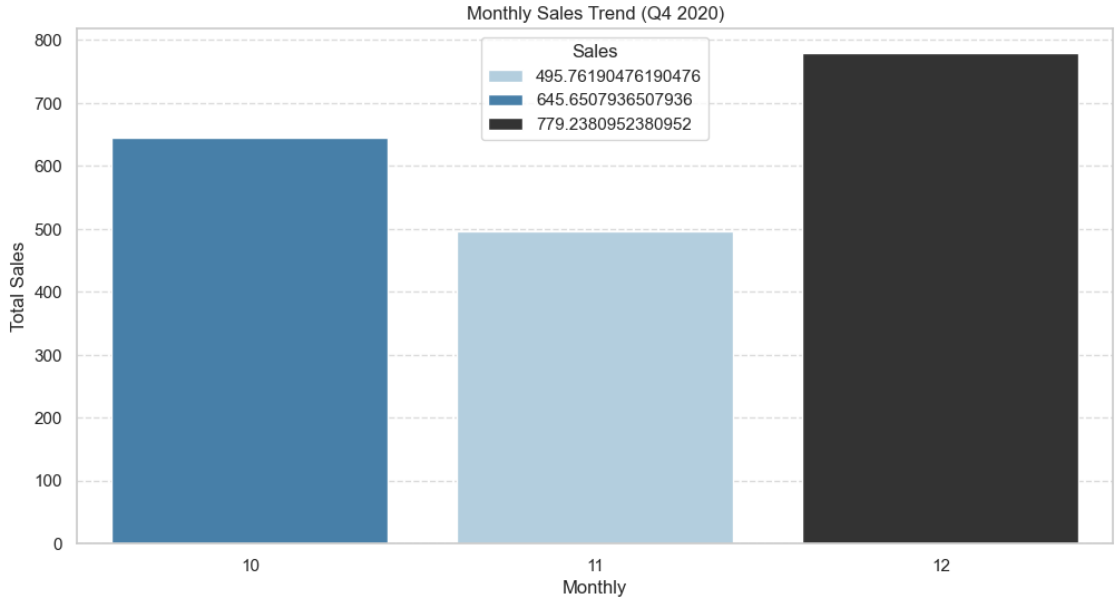

[ ]:
```
# Task 3-b daily sales trend line
daily_sales = df.groupby('date')['Sales'].sum().reset_index()
plt.figure(figsize=(12,6))
sns.lineplot(data=daily_sales, x='date', y='Sales', marker='o')
plt.title("Daily Sales Trend (Q4 2020)")

plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle="--", alpha=0.7)
plt.show()

# Task 3-b weekly sales trend line
weekly_sales = df.groupby('Week')['Sales'].sum().reset_index()
plt.figure(figsize=(12,6))
sns.lineplot(data=weekly_sales, x='Week', y='Sales', marker='o',estimator=None)
plt.title("Weekly Sales Trend (Q4 2020)")

plt.xlabel("Week")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle="--", alpha=0.7)
plt.show()
```




[43]:
```
# Task 3-c Monthly sales trend line
monthly_sales = df.groupby('Month')['Sales'].sum().reset_index()
plt.figure(figsize=(12,6))
sns.barplot(data=monthly_sales, x='Month',␣
 ↪y='Sales',palette='Blues_d',hue='Sales')
plt.title("Monthly Sales Trend (Q4 2020)")
```

```
plt.xlabel("Monthly")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle="--", alpha=0.7)
plt.show()
```




[46]:
```
#Task 3-b # Quarterly sales
plt.figure(figsize=(12, 5))
sns.barplot(data=quarterly_sales, x='Quarter', y='Sales')
plt.title("Quaterly Sales Trend")
plt.xlabel("Quarter")
plt.ylabel("Total Sales")
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()
```


Task 3C: Recommendation – Visualization Package For this project, the recommended visualization package is Seaborn. Reason: Seaborn is built on top of Matplotlib and provides: * Clean and professional plots with minimal code. * Better support for statistical visualizations (bar plots, box plots, line plots). * Readable styles and built-in themes, making the charts clear for business dashboards. Therefore, Seaborn is the preferred choice for building the sales analysis dashboard in this project.

[47]:
```
# Task 4: Report Generation
report = """
Sales Analysis Report (Q4)
1. State-wise Sales Analysis:
- Top states: VIC
- Lowest performing states: WA
2. Sales Distribution by Group:
- Highest sales category: Men
- Lowest sales category: Seniors
3. Time-of-Day Sales Analysis:
- Peak hours: Morning
- Off-peak hours: Afternoon
4. Sales Trends:
- Daily: Sales fluctuations throughout Q4
- Weekly: Identified high and low sales weeks
- Monthly: Sales trends for each month in Q4
- Quarterly: Overall Q4 sales comparison by state
Recommendations:
- Optimize inventory for peak hours
- Improve marketing strategies in low-sales states
- Leverage Q4 sales trends for better promotions
"""
print(report)
```

Sales Analysis Report (Q4)
1. State-wise Sales Analysis:
- Top states: VIC
- Lowest performing states: WA
2. Sales Distribution by Group:
- Highest sales category: Men
- Lowest sales category: Seniors
3. Time-of-Day Sales Analysis:
- Peak hours: Morning
- Off-peak hours: Afternoon
4. Sales Trends:
- Daily: Sales fluctuations throughout Q4
- Weekly: Identified high and low sales weeks
- Monthly: Sales trends for each month in Q4
- Quarterly: Overall Q4 sales comparison by state
Recommendations:
- Optimize inventory for peak hours
- Improve marketing strategies in low-sales states
- Leverage Q4 sales trends for better promotions