Project: Sales Analysis of AAL Company (Fourth Quarter 2020)

This project aims to analyze sales data for AAL, a clothing company, and provide insights for strategic decision-making. We will perform data wrangling, descriptive statistics, and visualization to help AAL make informed decisions about expanding into states with lower revenues and improving sales strategies across demographics.

Step 1: Data Wrangling

1.1 Loading and Inspecting the Data

First, we will load the dataset and inspect it to understand its structure and check for any inconsistencies.

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Load the dataset
df =
pd.read_csv('1716984926_ausapparalsales4thqrt2020/AusApparalSales4thQrt2020
.csv')

# Print first few rows of the dataset
print(df.head())

# Print the last few rows of the dataset
print(df.tail())

# Provide an overview of the dataset's structure
print(df.info())

# Display descriptive statistics
print(df.describe())

# Display data types of the columns
print(df.dtypes)
```

1.2 Missing Value Detection and Treatment

Check for missing values and handle them appropriately.

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

# Handle missing values by filling with the mean of the 'Sales' column
df = df.fillna(df["Sales"].mean())

# Remove any duplicates
df = df.drop_duplicates()

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

1.3 Data Normalization

Normalize the Sales column using Min-Max scaling.

```
# Normalize the 'Sales' column
df['Sales_Normalized'] = (df['Sales'] - df['Sales'].min()) /
(df['Sales'].max() - df['Sales'].min())
print(df[['Sales', 'Sales_Normalized']].head())
```

Step 2: Data Analysis

We will perform descriptive statistics and identify key insights related to sales performance.

2.1 Descriptive Statistical Analysis

```
# Descriptive statistics for Sales and Unit columns
sales_state = df[['Sales', 'Unit']].describe()
print(sales_state)

# Calculate additional statistics for Sales and Units
sales_mean = df['Sales'].mean()
sales_median = df['Sales'].median()
sales_mode = df['Sales'].mode()[0]
sales_std = df['Sales'].std()

unit_mean = df['Unit'].mean()
unit_median = df['Unit'].median()
unit_mode = df['Unit'].mode()[0]
unit_std = df['Unit'].std()

# Print statistical details
print(f"\nSales Mean: {sales_mean}")
print(f"Sales Median: {sales_median}")
print(f"Sales Mode: {sales_mode}")
print(f"Sales Standard Deviation: {sales_std}")
print(f"\nUnit Mean: {unit_mean}")
print(f"Unit Median: {unit_median}")
print(f"Unit Mode: {unit_mode}")
print(f"Unit Standard Deviation: {unit_std}")
```

2.2 Identifying the Group with the Highest and Lowest Sales

```
# Grouping sales by demographics (Group)
sales_by_group = df.groupby('Group')['Sales'].sum().reset_index()
print(sales_by_group)

# Identify group with highest and lowest sales
highest_sales = sales_by_group.sort_values('Sales',
ascending=False).iloc[0]
lowest_sales = sales_by_group.sort_values('Sales', ascending=True).iloc[0]

print("Group with Highest Sales:")
print(highest_sales)

print("\nGroup with Lowest Sales:")
print(lowest_sales)
```

2.3 Identifying the Group with the Highest and Lowest Sales (Based on Units)

```
# Grouping by Units and identifying highest and lowest sales based on units
sales_by_group_units = df.groupby('Group')['Unit'].sum().reset_index()
highest_unit_sales = sales_by_group_units.sort_values('Unit',
ascending=False).iloc[0]
lowest_unit_sales = sales_by_group_units.sort_values('Unit',
ascending=True).iloc[0]

print("Group with Highest Sales (based on units):")
print(highest_unit_sales)

print("\nGroup with Lowest Sale (based on units):")
print(lowest_unit_sales)
```

2.4 Generating Weekly, Monthly, and Quarterly Reports

```
# Resetting the index and setting 'Date' as datetime format
df = df.reset_index()
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Generate weekly, monthly, and quarterly reports
weekly_sales = df.resample('W').agg({'Sales': 'sum'}).reset_index()
```

```
monthly_sales = df.resample('M').agg({'Sales': 'sum'}).reset_index()
quarterly_sales = df.resample('Q').agg({'Sales': 'sum'}).reset_index()

# Display the first few rows of each report
print("Weekly Sales Report:")
print(weekly_sales.head())

print("\nMonthly Sales Report:")
print(monthly_sales.head())

print("\nQuarterly Sales Report:")
print(quarterly_sales.head())
```

Step 3: Data Visualization

We will create visualizations to provide a clear overview of sales data and trends.

3.1 Visualizing Sales by State for Different Demographics

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

# State-wise sales analysis for different demographic groups
plt.figure(figsize=(14, 8))
g = sns.catplot(x='State', y='Sales', hue='Group', kind='bar', data=df,
height=6, aspect=2)
g.set_xticklabels(rotation=90)
g.set_axis_labels("State", "Sales")
g.fig.suptitle("State-wise Sales Analysis for Different Demographic
Groups", fontsize=16)
plt.tight_layout()
plt.show()
```

3.2 Group-wise Sales Analysis Across States (Heatmap)

```
python
# Pivot the data for heatmap
group_state_sales = df.pivot_table(values='Sales', index='State',
columns='Group', aggfunc='sum')

plt.figure(figsize=(14, 8))
sns.heatmap(group_state_sales, annot=True, cmap="YlGnBu", fmt=".1f",
linewidths=.5)
plt.title('Group-wise Sales Analysis Across States')
plt.ylabel('State')
plt.xlabel('Group')
plt.tight_layout()
plt.show()
```

3.3 Time-of-the-Day Sales Analysis

```
python
# Assuming there is a 'Time' column for morning/afternoon/evening sales
time_of_day_sales = df.groupby('Time')['Sales'].sum().reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Time', y='Sales', data=time_of_day_sales, palette='viridis')
plt.title('Time-of-the-Day Sales Analysis')
plt.xlabel('Time of Day')
plt.ylabel('Total Sales')
plt.tight_layout()
plt.show()
```

3.4 Sales Trend Over Time (Daily, Weekly, Monthly, Quarterly)

```
python
# Plotting the sales trends for each time period
plt.figure(figsize=(14, 8))

# Weekly sales plot
plt.subplot(2, 2, 1)
sns.lineplot(data=weekly_sales, x=weekly_sales.index, y='Sales')
plt.title('Weekly Sales')

# Monthly sales plot
plt.subplot(2, 2, 2)
sns.lineplot(data=monthly_sales, x=monthly_sales.index, y='Sales')
plt.title('Monthly Sales')

# Quarterly sales plot
plt.subplot(2, 2, 3)
sns.lineplot(data=quarterly_sales, x=quarterly_sales.index, y='Sales')
plt.title('Quarterly Sales')

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

Step 4: Recommendations

4.1 States with the Highest and Lowest Revenues

- Based on the state-wise sales analysis:
  - States with Highest Revenues: These states should continue with their current strategies and potentially expand their presence further, especially focusing on high-performing groups (e.g., Women, Men).

  - States with Lowest Revenues: Target these states for special marketing campaigns, promotions, and discounts to increase sales. Consider offering special deals tailored to local preferences to boost engagement.

4.2 Group Analysis Recommendations

- The Group with the Highest Sales (e.g., Men, Women) should see increased focus, with personalized offerings and exclusive product launches.

- The Group with the Lowest Sales (e.g., Seniors, Kids) requires focused attention through tailored products, discounts, and targeted advertising.

4.3 Time-of-the-Day Sales Recommendations

- Peak Sales Periods: Leverage the peak sales periods (e.g., evening hours) for flash sales, real-time offers, or even next-best offers to enhance customer engagement.
- Off-Peak Periods: For off-peak hours, consider offering discounted flash sales to drive traffic during quieter times.

4.4 General Recommendations

- Expansion Strategy: For states with lower sales, consider further market research to understand local preferences, and deploy targeted marketing strategies.
- Diversification: The company should look into expanding its product lines or introducing seasonal collections to maintain high engagement across all demographics.