```
# %% [markdown]
# # Project to Sales Analysis of AAL Company

# %%
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import pandas as pd
import os

# %%
#load Dataset
df = pd.read_csv('1716984926_ausapparalsales4thqrt2020/AusApparalSales4thQrt2020.csv')

# %%
# Print few head rows from dataset
print(df.head())


# %%

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


# %%

# Providing information about the DataFrame, including data types and non-null counts
print(df.info())


# %%
# Displaying descriptive statistics of the DataFrame, such as mean, std, min, max, and so
on.
print(df.describe())



# %%
# Displaying datatypes of the columns
df.dtypes


# %% [markdown]
# ## Data wrangling

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

# %%
# Checking for naN values
missing_values = df.notna().sum()
print("Missing Values per Column:")
print(missing_values)

# %%
# Handling missing values
df = df.fillna(df["Sales"].mean())


# %%
# Removing duplicate records
df = df.drop_duplicates()
```

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

# %% [markdown]
# # Data Transformation

# %%
df['Sales_Normalized'] = (df['Sales'] - df['Sales'].min()) / (df['Sales'].max() -
df['Sales'].min())
df

# %% [markdown]
# # Grouping Operation

# %%
# Average sales for each combination of State and Group.
df_grouped = df.groupby(['State', 'Group'])['Sales'].mean().reset_index()
print(df_grouped)

# %% [markdown]
# ## Identify States with Highest Revenues

# %%

# Group by state and sum the sales
df_state_sales = df.groupby('State')['Sales'].sum().reset_index()

# Sort states by total sales in descending order
df_state_sales_sorted = df_state_sales.sort_values(by='Sales', ascending=False)
print(df_state_sales_sorted)

# %% [markdown]
# ## Identify States with Lowest Revenue

# %%
# Sort states by total sales in descending order
df_state_sales_sorted = df_state_sales.sort_values(by='Sales', ascending=True)
print(df_state_sales_sorted)

# %%
print("State with Highest Revenues: \n", df_state_sales_sorted.tail(1))
print("State with Lowest Revenues: \n", df_state_sales_sorted.head(1))

# %% [markdown]
# # Data analysis
#

# %%
sales_state = df[['Sales', 'Unit']].describe()
sales_state

# %%
# Calculate additional statistics
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("Sales Descriptive Stats:")
print(sales_state)

# %%
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}")

# %%
df['Sales'].hist()

# %% [markdown]
# ## Identify the Group with the Highest and Lowest Sales

# %%
sales_by_group = df.groupby('Group')['Sales'].sum().reset_index()
sales_by_group

# %%
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)

# %% [markdown]
# ## Identify the Group with the Highest and Lowest Sales (Based on Units)

# %%
sales_by_group = df.groupby('Group')['Unit'].sum().reset_index()
highest_unit_sales = sales_by_group.sort_values('Unit', ascending=False).iloc[0]
lowest_unit_sales = sales_by_group.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 om units):")
print(lowest_unit_sales)

# %% [markdown]
# ##  Generate Weekly, Monthly, and Quarterly Reports

# %%
df = df.reset_index()
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Generate weekly reports
weekly_report = df.resample('W').agg({'Sales': 'sum', 'Unit': 'sum'}).reset_index()

# Generate monthly reports
```

```
monthly_report = df.resample('ME').agg({'Sales': 'sum', 'Unit': 'sum'}).reset_index()

# Generate Quaterly reports
quarterly_report = df.resample('QE').agg({'Sales': 'sum', 'Unit': 'sum'}).reset_index()

# %%
# Print or save reports
print("Weekly Report:")
print(weekly_report.head())

print("\nMonthly Report:")
print(monthly_report.head())

print("\nQuarterly Report:")
print(quarterly_report.head())

# %% [markdown]
# # Data Visualization

# %%
!pip install seaborn

# %%
import seaborn as sns
import matplotlib.pyplot as plt

# %%

# Assuming 'df' contains a 'State', 'Group', and 'Sales' column
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")

# Correct way to set the title for FacetGrid
g.fig.suptitle("State-wise Sales Analysis for Different Demographic Groups", fontsize=16)

plt.tight_layout()
plt.show()

# %%
# Pivot the data for a 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()

# %%
# Assuming 'Time' is the column indicating morning/afternoon/evening
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()
```

```
# %%


# %%
# Ensure the 'Date' column is in datetime format
df = df.reset_index()
df['Date'] = pd.to_datetime(df['Date'])

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

# Resample for daily, weekly, monthly, and quarterly data
daily_sales = df.resample('D').agg({'Sales': 'sum'})
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("Daily Sales Report:")
print(daily_sales.head())

print("Weekly Sales Report:")
print(weekly_sales.head())

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

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

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

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

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

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

# Quarterly sales plot
plt.subplot(2, 2, 4)
sns.lineplot(data=quarterly_sales, x=quarterly_sales.index, y='Sales')
plt.title('Quarterly Sales')

plt.tight_layout()
plt.show()

# %% [markdown]
# ## **Recommendations**
#
# ## **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.
#
# ## **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.
#
# ## **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 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.

# %% [markdown]
#
```