# **Course-End Project Solution**

**Applied Data Science with Python**

## **Sales Analysis**

### **Problem Statement:**

AAL, established in 2000, is a well-known brand in Australia, particularly recognized for its clothing business. It has opened branches in various states, metropolises, and tier-1 and tier-2 cities across the country.

The brand caters to all age groups, from kids to the elderly.

Currently experiencing a surge in business, AAL is actively pursuing expansion opportunities. To facilitate informed investment decisions, the CEO has assigned the responsibility to the head of AAL’s sales and marketing (S&M) department. The specific tasks include:

* Identify the states that are generating the highest revenues.
* Develop sales programs for states with lower revenues. The head of sales and marketing has requested your assistance with this task.

Analyze the sales data of the company for the fourth quarter in Australia, examining it on a state-by-state basis. Provide insights to assist the company in making data-driven decisions for the upcoming year.

### **Steps to Perform:**

1. **• Data wrangling**

* **Ensure that the data is clean and free from any missing or incorrect entries.**
  + **Inspect the data manually to identify missing or incorrect information using the functions isna() and notna().**

import pandas as pd

# Load the CSV again if needed

file\_path = 'AusApparalSales4thQrt2020.csv'

df = pd.read\_csv(file\_path)

# Check for missing values (True = missing)

print("🔍 Missing Values Summary (isna()):")

print(df.isna().sum())

# Check for non-missing (valid) data entries

print("\n✅ Non-missing value check (notna()):")

print(df.notna().sum())

1. **Recommendations for missing rows :**

* This is based on the columns , if dates are missing it is better to drop the rows, it is preferred to drop the rows or fill it with median
* Never keep impossible values (e.g., negative sales).

1. **Choose a suitable data wrangling technique—either data standardization or normalization. Execute the preferred normalization method and present the resulting data. *(Normalization is the preferred approach for this problem.)***

*from sklearn.preprocessing import MinMaxScaler*

*scaler = MinMaxScaler()*

*df[['Sales\_Normalized', 'Unit\_Normalized']] = scaler.fit\_transform(df[['Sales', 'Unit']])*

*print(df[['Sales', 'Sales\_Normalized', 'Unit', 'Unit\_Normalized']].head())*

1. ***Share your insights regarding the application of the GroupBy() function for either data chunking or merging, and offer a recommendation based on your analysis.***

The groupby() function is used to:

* **Split** the data into meaningful groups based on a key (e.g., State, Date, Demographic Group)
* **Apply** aggregation functions (like sum(), mean(), count())
* **Combine** the results into a summarized form

df.groupby('State')['Sales'].sum().sort\_values(ascending=False)

df.groupby(df['Date'].dt.to\_period('M'))['Sales'].sum()

1. **Perform descriptive statistical analysis on the data in the Sales and Unit columns. Utilize techniques such as mean, median, mode, and standard deviation for this analysis.**

# Ensure numeric columns are correctly typed

df['Sales'] = pd.to\_numeric(df['Sales'], errors='coerce')

df['Unit'] = pd.to\_numeric(df['Unit'], errors='coerce')

# Drop rows with NaNs in 'Sales' or 'Unit'

df.dropna(subset=['Sales', 'Unit'], inplace=True)

# Calculate descriptive statistics for Sales

sales\_stats = {

'mean': df['Sales'].mean(),

'median': df['Sales'].median(),

'mode': df['Sales'].mode().iloc[0] if not df['Sales'].mode().empty else None,

'std\_dev': df['Sales'].std()

}

# Calculate descriptive statistics for Unit

unit\_stats = {

'mean': df['Unit'].mean(),

'median': df['Unit'].median(),

'mode': df['Unit'].mode().iloc[0] if not df['Unit'].mode().empty else None,

'std\_dev': df['Unit'].std()

}

# Print the results

print("Sales Statistics:")

for k, v in sales\_stats.items():

print(f"{k.title()}: {v:.2f}")

print("\nUnit Statistics:")

for k, v in unit\_stats.items():

print(f"{k.title()}: {v:.2f}")

1. **Identify the group with the highest sales and the group with the lowest sales based on the data provided.**
2. **Identify the group with the highest and lowest sales based on the data provided.**

# Ensure 'Sales' column is numeric

df['Sales'] = pd.to\_numeric(df['Sales'], errors='coerce')

# Group by 'Group' and sum the Sales

group\_sales = df.groupby('Group')['Sales'].sum().reset\_index()

# Identify group with highest sales

highest\_sales\_group = group\_sales.loc[group\_sales['Sales'].idxmax()]

# Identify group with lowest sales

lowest\_sales\_group = group\_sales.loc[group\_sales['Sales'].idxmin()]

# Print results

print("Total Sales by Group:\n", group\_sales)

print("\nGroup with Highest Sales:")

print(highest\_sales\_group)

print("\nGroup with Lowest Sales:")

print(lowest\_sales\_group)

1. **Generate weekly, monthly, and quarterly reports to document and present the results of the analysis conducted.**

***(Use suitable libraries such as NumPy, Pandas, and SciPy for performing the analysis.)***

|  |
| --- |