# PROG8421 - Programming for Big Data

**Final Project**

**Group 1 Members:**

|  |  |
| --- | --- |
| **Name: Sudhan Shrestha** | **Name: Jose Encarnacion** |
| **Student ID: 8889436** | **Student ID: 8982860** |

**Part I: Data Overview**

***Prerequisites***

**Loading the libraries**

**Code**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

import datetime

warnings.filterwarnings("ignore")

**Comment**

Importing all the required libraries on the first cell of the notebook.

**Screenshot**

**A screen shot of a computer program

Description automatically generated**

1. **Use the pandas library to read the data file and to create the data frame.**

**Code**

data\_path ='customers.csv' # setting the path to the data file

df = pd.read\_csv(data\_path, delimiter='\t') # reading the csv file, with delimiter as tab since data are separated by tabs

df

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Display the first 5 rows and the last 3 rows of your data.**

**Code**

df.head() # first 5 rows of the dataframe

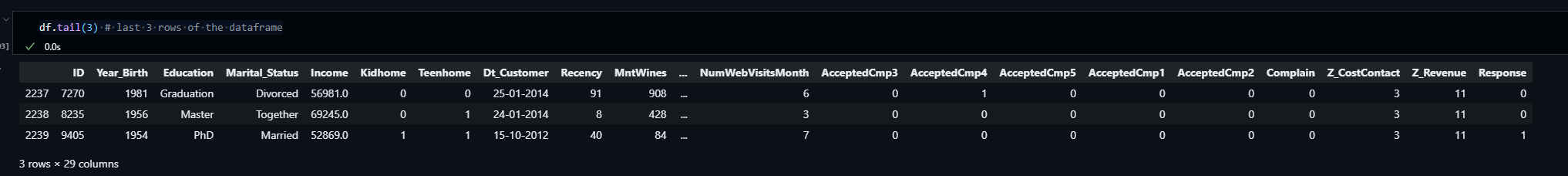
df.tail(3) # last 3 rows of the dataframe

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

****

1. **Show quick statistics of your data**

**Code**

df.describe()

**Comment**

**Screenshot**

**A screenshot of a computer screen

Description automatically generated**

1. **Show the data type of each column**

**Code**

print("Data types of all columns:")

df.dtypes

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Show how many columns and rows in your data.**

**Code**

rows = df.shape[0] # number of rows in the dataframe

columns = df.shape[1] # number of columns in the dataframe

print(f"Number of rows: {rows}")

print(f"Number of columns: {columns}")

**Comment**

**Screenshot**

**A screenshot of a computer code

Description automatically generated**

1. **Show the list of columns in your data frame.**

**Code**

columns\_list = df.columns.tolist() # convert the column names to a list

print("List of columns:")

columns\_list

**Comment**

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Show the number of duplicated rows in your data.**

**Code**

duplicate\_count = df.duplicated().sum() # count the number of duplicate rows

print(f"The number of duplicate rows is: {duplicate\_count}")

**Comment**

**Screenshot**

**A screen shot of a computer code

Description automatically generated**

**Part II: Data Preparation and Cleaning**

1. **Rename the columns belonging to Products by removing Mnt from them.**
   1. **You should have: 'Wines','Fruits','Meat','Fish','Sweet','Gold'**

**Code**

columns\_to\_rename = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']

new\_column\_names = ['Wines', 'Fruits', 'Meat', 'Fish', 'Sweet', 'Gold']

dfm = df.copy() # creating a copy of the original dataframe

dfm.rename(columns=dict(zip(columns\_to\_rename, new\_column\_names)), inplace=True) # renaming the columns

dfm[new\_column\_names].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Rename the columnsbelonging to Place by removing Num and Purchases from them.** 
   1. **You should have: 'Web ','Catalog', 'Store'**

**Code**

columns\_to\_rename = ['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases']

new\_column\_names = ['Web', 'Catalog', 'Store']  # Corrected column names needed to rename

dfm.rename(columns=dict(zip(columns\_to\_rename, new\_column\_names)), inplace=True)

dfm[new\_column\_names].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Keep only NumDealsPurchases column in the Promotion section**

**Code**

columns\_to\_keep = ['NumDealsPurchases']

dfm[columns\_to\_keep].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Delete the column NumWebVisitsMonth**

**Code**

columns\_to\_delete = ['NumWebVisitsMonth']

dfm['NumWebVisitsMonth'].head()

dfm.drop(columns=columns\_to\_delete, inplace=True)

# dfm['NumWebVisitsMonth'].head()

**Comment**

**Screenshot**

**A screen shot of a computer code

Description automatically generated**

1. **The attribute Year\_Birth is not helpful to segment your data. Instead create a new derived column named Age.**

**Code**

current\_year = datetime.datetime.now().year # getting the current year

dfm['Age'] = current\_year - dfm['Year\_Birth'] # calculating the age (current year - year of birth)

dfm['Age'].head() # displaying the first 5 rows of the 'Age' column

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Create a new column named Spending that holds the total amount spent on all products categories.**

**Code**

dfm['Spending'] = dfm['Wines'] + dfm['Fruits'] + dfm['Meat'] + dfm['Fish'] + dfm['Sweet'] + dfm['Gold']

dfm['Spending'].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Change the Dt\_Customer type from object to datetime. Use the format YYYY-MM-DD.**

**Code**

# Convert the 'Dt\_Customer' column to datetime format

dfm['Dt\_Customer'] = pd.to\_datetime(dfm['Dt\_Customer'], format='%d-%m-%Y')

dfm['Dt\_Customer'].head()

**Comment**

**Screenshot**

**A screen shot of a computer program

Description automatically generated**

1. **Show the unique values of the column data Marital\_Status.**

**Code**

print("The unique values in the 'Marital\_Status' column are:")

print(dfm['Marital\_Status'].unique())

**Comment**

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **It is better to work with lower categorical values. Hence, we’ll classify customers in two segments for their martial status: change the column Marital status as follows:** 
   1. **Change the values: Divorced, Single, Absurd and Widow to 'Alone'**
   2. **Change Married and Together to 'Not Alone'**

**Code**

new\_values = {

    'Divorced': 'Alone',

    'Single': 'Alone',

    'Absurd': 'Alone',

    'Widow': 'Alone',

    'Married': 'Not Alone',

    'Together': 'Not Alone'

}

dfm['Marital\_Status'] = dfm['Marital\_Status'].replace(new\_values)

dfm['Marital\_Status'].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Show the unique values of the column Education.**

**Code**

print("The unique values in the 'Education' column are:")

print(dfm['Education'].unique())

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **We want to segment the customers in 2 groups based on their education level: change the column Education as follows:** 
   1. **Change the values: 'Basic' and '2n Cycle' to 'Undergraduate',**
   2. **All other values to 'Postgraduate'**

**Code**

columns = ['Education']

new\_values = {

    'Basic': 'Undergraduate',

    '2n Cycle': 'Undergraduate'

}

dfm['Education'] = dfm['Education'].replace(new\_values)

dfm['Education'].loc[~dfm['Education'].isin(['Undergraduate'])] = 'Postgraduate'

dfm['Education'].loc[dfm['Education'].isin(['Undergraduate'])].head()

print("The unique values in the 'Education' column after replacing values are:")

print(dfm['Education'].unique())

**Comment**

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Have a look at column Income. Investigate the presence of outliers? Delete them accordingly.**

**Code**

columns = ['Income']

dfm[columns].describe()

# checking for outliers in the income using boxplot

sns.boxplot(dfm['Income'], color='lightgreen')

plt.title('Income Column Boxplot')

plt.show()

# removing the outliers using IQR method

q1 = dfm['Income'].quantile(0.25)

q3 = dfm['Income'].quantile(0.75)

iqr = q3 - q1

# removing the outliers by only keeping the values that are within the IQR range

dfm = dfm[(dfm['Income'] >= q1 - 1.5 \* iqr) & (dfm['Income'] <= q3 + 1.5 \* iqr)]

sns.boxplot(dfm['Income'], color='skyblue')

plt.title('Income Column Boxplot (After Outlier Removal)')

plt.show()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

1. **Show quick statistics for the column Income.**

**Code**

columns = ['Income']

df[columns].describe()

# dfm[columns].describe() # error the column was delete

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Show the number of missing values for each column.**

**Code**

columns = df.columns

missing\_values = df[columns].isnull().sum()

print(missing\_values)

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Fill the missing values with the average.**

**Code**

columns = ['Income']

df[columns] = df[columns].fillna(df[columns].mean())

print(f"The missing value in the 'Income' column is: {df['Income'].isnull().sum()}")

**Comment**

**Screenshot**

**A screen shot of a computer code

Description automatically generated**

1. **Create a new column named Children to hold the total number of children for every customer.**

**Code**

new\_column\_name = 'Children'

dfm[new\_column\_name] = dfm['Kidhome'] + dfm['Teenhome']

dfm[new\_column\_name].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Create a new column Has\_Complaint as follows: a. 'Has complaint' if the customer complained in the last 2 years, b. Otherwise 'No complaint'**

**Code**

columns = ['Complain']

dfm['Has\_Complaint'] = dfm['Complain'].apply(lambda x: 'Has complaint' if x == 1 else 'No complaint')

dfm['Has\_Complaint'].head()

dfm['Has\_Complaint'].unique() # checking the unique values to see changes

**Comment**

**Screenshot**

**A screenshot of a computer program

Description automatically generated**

1. **Show the first 5 rows of customers with ‘No complaint’**

**Code**

dfm[dfm['Has\_Complaint'] == 'No complaint'].head()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**Part III: Data Visualization & Analytics**

1. **Let’s analyse the profile of customers based on their background factors. To do so, we will use our cleaned data frame (in part II) to create a new one with the columns: Age, Spending, Marital\_Status, Education, Income, Children, Dt\_Customer and Has\_Complaint.**

**Code**

columns\_to\_add = ['Age', 'Spending', 'Marital\_Status', 'Education', 'Income', 'Children','Dt\_Customer', 'Has\_Complaint']

df\_customer\_profile = dfm[columns\_to\_add]

df\_customer\_profile

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

1. **Create histogram to show the number of children per customers in your data. What do you see?**

**Code**

# histogram to show the number of childerns per customer

sns.histplot(df\_customer\_profile['Children'],bins= range(df\_customer\_profile['Children'].min(),df\_customer\_profile['Children'].max()+1), color='#7986CB')

# creating labels and titles

plt.title('Histogram of Children per Customer')

plt.xlabel('Number of Children')

plt.ylabel('Frequency')

**Comment**

**Screenshot**

**A screenshot of a computer screen

Description automatically generated**

1. **Create a histogram and a Boxplot to show the Income of customers. What do you find?**

**Code**

# histogram to show the income of customers

sns.histplot(df\_customer\_profile['Income'], color='#7986CB', kde=True)

plt.title('Income Distribution of Customers')

plt.xlabel('Income')

plt.ylabel('Frequency')

plt.show()

sns.boxplot(df\_customer\_profile['Income'], color='lightgreen')

plt.title('Income Distribution of Customers')

plt.xlabel('Income')

plt.show()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram and a Boxplot to show the Spending of customers. What do you find?**

**Code**

# Histogram for Spending of Customers

sns.histplot(df\_customer\_profile['Spending'], color='#00838F', edgecolor='black', kde=True)

plt.title('Spending Distribution of Customers')

plt.xlabel('Spending')

plt.ylabel('Frequency')

plt.show()

# Boxplot for Spending of Customers

sns.boxplot(df\_customer\_profile['Spending'], color='lightgreen')

plt.title('Spending Distribution of Customers')

plt.xlabel('Spending')

plt.show()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram and a boxplot to analyse the customers based on their Age.**

**Code**

# Histogram for Age of Customers

sns.histplot(df\_customer\_profile['Age'], color='#00838F',bins=35)

plt.title('Age Distribution of Customers')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.show()

# Boxplot for Age of Customers

sns.boxplot(df\_customer\_profile['Age'], color='lightgreen')

plt.title('Age Distribution of Customers')

plt.xlabel('Age')

plt.show()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Create a histogram to analyse the Education level of the company’s customers. What do you find?** 
   1. **The column Education is defined as Object and has 2 values (Undergraduate, Postgraduate). You need to shape the values as numeric to be able to plot the histogram**

**Code**

# histogram to show the education level of the company’s customers

sns.histplot(df\_customer\_profile['Education'],color='teal')

plt.title("Histogram of Education level of the Company's Customer")

plt.xlabel('Education Level')

plt.ylabel('Frequency')

plt.show()

# # convert to categorical data to numeric

df\_customer\_profile['Education'] = df\_customer\_profile['Education'].map({'Postgraduate': 1, 'Undergraduate': 0})

sns.histplot(df\_customer\_profile['Education'],color='teal', bins=5)

plt.title("Histogram of Education level of the Company's Customer (After converting to numeric)")

plt.xlabel('Education Level')

plt.ylabel('Frequency')

plt.show()

**Comment**

**Screenshot**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**