**Task 3: Presentation**

Krescens Kok

Department of Technology, Western Governors University

D598: Analytics Programming

Keiona Middleton

2025 June 16

1. **Explain how the code works for the program you submitted in Task 2.**

The following line imports the Pandas package used in Python, and aliases it as ‘pd’, to assist in the manipulation of data structures (*Pandas*, 2025).

import pandas as pd

This uses the pandas package to read the excel file ‘D598 Data Set.xlsx’ and assigns it to a variable called ‘df.’

df = pd.read\_excel("D598 Data Set.xlsx")

Print statements are used to print text and outputs it to the console. The first print statement is just printing ‘Duplicated rows: ‘ to the console. The second print statement is printing a series of Boolean values (true/false) that represent whether or not the row is a duplicate or not.

The last line in this screenshot takes the data in the variable, ‘df’, drops the duplicates, and re-assigns the result back to the same variable, ‘df.’

print("Duplicate rows: ")  
print(df[df.duplicated()])  
  
# Remove duplicated rows  
df = df.drop\_duplicates()

This gets all the numeric data columns from the data in df and assigns a list of numeric column names to a new variable called ‘numeric\_cols.’

numeric\_cols = df.select\_dtypes(include='number').columns

This takes the data in ‘df’, groups it by the Business State column and aggregates the numeric columns, which was found in the previous step, to calculate the mean, median, min, and max.

It then prints out ‘Printing descriptive stats: ‘ to the console and then prints the dataframe with the aggregate statistics for each business state.

descriptive\_df = df.groupby("Business State")[numeric\_cols].aggregate(['mean', 'median', 'min', 'max'])  
print("Printing descriptive stats: ")  
print(descriptive\_df)

Here, the code filters the data in df, where the Debt to Equity column values are less than zero, or negative. It assigns the results from the filter into a new variable called neg\_dte\_df.

It then prints out ‘Negative debt to equity records: ‘ and prints the filtered dataframe to the console.

neg\_dte\_df = df[df['Debt to Equity'] < 0]  
print("Negative debt to equity records: ")  
print(neg\_dte\_df)

The code then calculates the debt to income ratio by taking the total long-term debt column in df and divides it by the total revenue column in df. The division is only done where the total revenue is not equal to zero and where the value is not null. The calculations are then stored into a variable called dti.

dti = (df['Total Long-term Debt'] / df['Total Revenue']).where(  
 (df['Total Revenue'] != 0) & (df['Total Revenue'].notna())  
)

The code then takes the dti variable and creates a dataframe using the Pandas package, and names the column ‘DTI Calculation’, and assigns this dataframe to a new dataframe called dti\_df.

It then prints the dataframe to the console using the print function.

dti\_df= pd.DataFrame(dti, columns=['DTI Calculation'])  
print(dti\_df)

Lastly, the code uses the concat function from the Pandas package to concatenate the original dataframe that was created in the beginning, df, and the dti\_df that was created in the previous step. Axis=1 signifies that the dataframes will be concatenated on the column axis, placing the dataframes side by side (*pandas.concat — Pandas 2.3.0 Documentation*, n.d.). The result of pd.concat is stored in the variable called final\_df.

The text, ‘Final dataframe: ‘ is then printed out to the console, followed by the concatened dataframe.

final\_df = pd.concat([df, dti\_df], axis=1)  
print("Final dataframe: ")  
print(final\_df.head())

1. **Provide 4** **customized data visualizations.**

**Bar Graph:**

**![A graph of a number of states

AI-generated content may be incorrect.]()**

**Scatterplot:**

**![A graph with blue dots

AI-generated content may be incorrect.]()**

**Box Plot:**

**![A graph with numbers and a bar

AI-generated content may be incorrect.]()**

**Density Graph:**

**![A graph of a graph

AI-generated content may be incorrect.]()**

1. **Explain how customized visualizations in part B were created.**
2. **Bar Graph:** For the bar graph, the Python library, matplotlib, was used and imported. Matplotlib is a library that allows for creating different types of graphs such as bar graphs, histograms, scatterplots, etc… The maximum total long-term debt for each state was assigned to a new variable called ‘max\_debt\_by\_state.’ This was done by using the descriptive\_df that was used earlier in the code to get aggregate statistics for each state, and since the data frame has a hierarchical structure, it was necessary to specify the column name, ‘Total Long-term Debt’, and the statistic ‘max’ value. The next line of code was actually plotting the data and specifying that it’s a bar graph, specifying the color of the bars, and the figure size of the plot. The remaining lines are adjusting details about the bar graph. The title is specified, along with the x and y labels. The x labels were rotated by 90 degrees and plt.tight\_layout() was specified to prevent overlapping. Lastly, plt.show() is what actually reveals the bar graph.

import matplotlib.pyplot as plt  
  
max\_debt\_by\_state = descriptive\_df['Total Long-term Debt']['max']  
  
max\_debt\_by\_state.plot(kind='bar', color='teal', figsize=(10, 6))  
  
plt.title('Maximum Total Long-term Debt by Business State')  
plt.xlabel('Business State')  
plt.ylabel('Max Total Long-term Debt')  
plt.xticks(rotation=90)  
plt.tight\_layout()  
plt.show()

1. **Scatterplot:** For this graph, matplotlib and seaborn Python packages were both used. Seaborn is built on top of matplotlib that allows for more attractive drawing of graphs (*Seaborn: Statistical Data Visualization — Seaborn 0.13.2 Documentation*, n.d.). The final\_df was from the existing code after concatenating the original data frame and the data frame that had the debt to income calculations. Using the seaborn library, a scatterplot function is used where the data is specified to be the final\_df and the x and y values are specified, where x is the DTI Calculation column and y is the Total Long-term Debt column. The next few lines specify the x and y labels and the title. Lastly, the plot is being shown with plt.show().

import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(data=final\_df, x='DTI Calculation', y='Total Long-term Debt')  
plt.xlabel('Debt to Income Ratio')  
plt.ylabel('Total Long-term Debt')  
plt.title('Debt to Income Ratio vs. Total Long-term Debt')  
plt.show()

1. **Box Plot:** For this graph, seaborn and matplotlib libraries are being used. Just like the previous graph, the final\_df is being used as the data. Using the boxplot function from the seaborn library, a boxplot is specified as the type of graph. The data is specified as final\_df, x is specified as the Total Revenue column, and lastly, the color of the boxplot is set to skyblue. The title and x label are added to the plot, and lastly, plt.show() is used to show the plot.

import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.boxplot(data=final\_df, x='Total Revenue', color='skyblue')  
  
plt.title('Box Plot of Total Revenue')  
plt.xlabel('Total Revenue')  
plt.show()

1. **Density Graph:** For this graph, seaborn and matplotlib are being used. For the density graph, a function called kdeplot from the seaborn package is being used. The final\_df is also used as the data, where the x axis is the Total Liabilities column, fill is set to True, meaning that the area under the curve will be filled in with the color specified, which is skyblue in this case. The title, x and y axis labels are specified, and lastly, the plot is being shown.

import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.kdeplot(data=final\_df, x='Total Liabilities', fill=True, color='skyblue')  
  
plt.title('Density Plot of Profit Margin')  
plt.xlabel('Profit Margin')  
plt.ylabel('Density')  
plt.show()

**References**

*pandas*. (2025, June 5). PyPI. <https://pypi.org/project/pandas/>

*pandas.concat — pandas 2.3.0 documentation*. (n.d.). https://pandas.pydata.org/docs/reference/api/pandas.concat.html

*seaborn: statistical data visualization — seaborn 0.13.2 documentation*. (n.d.). https://seaborn.pydata.org/