Nicole Gallo
D598 – Analytics Programming
June 22, 2025
QKN1 — QKN1 Task 3: Presentation

A. Explain how the code works for the program you submitted in Task 2.
The code in the Python program in Task 2 follows a logical data analysis sequence:

1. **Importing the dataset**
   The Excel file (D598_Data_Set.xlsx) is imported into an open-source Python library called *pandas* to allow our data to be manipulated. A try/except block is used to handle file loading errors.

```
import pandas as pd
import numpy as np

# 1.IMPORT the dataset from "D598 Data Set.xlsx" into a data frame called df
try:
    df = pd.read_excel("D598_Data_Set.xlsx", sheet_name="1-150 V2")
except Exception as e:
    raise FileNotFoundError(f"Error loading Excel file: {e}")
```

2. **Identifying duplicates (and removing duplicates if necessary)**
   To ensure data quality, rows in the dataset are checked for duplication using the *df.duplicated()* function. The number of duplicates in the dataset is printed.
   If duplicates are present, there is a function called, *df.drop_duplicates()*, to remove any duplicated rows.

```
# 2.IDENTIFY any duplicate rows in df
# CALL df.duplicated() to detect duplicates
# STORE duplicates in a separate data frame if needed
duplicates = df[df.duplicated()]
print("Duplicate rows found:", len(duplicates))
# REMOVE duplicates if any
if not duplicates.empty:
    df = df.drop_duplicates()
    print("Duplicates removed.")
```

3. **Grouping by state and calculating descriptive statistics**
   The program defines "required_columns" that will be used for analysis. If those columns exist, the data is grouped by the "Business State" column. The descriptive statistics (mean, median, min, max) are then calculated for columns containing numbers using the *.agg()* function.

```
# 3.GROUP df BY "Business State"
required_columns = [
    'Total Revenue', 'Total Long-term Debt', 'Total Equity',
```

```
    'Total Liabilities', 'Debt to Equity', 'Profit Margin'
]

# CALCULATE descriptive statistics (mean, median, min, max)
# PRINT/STORE this grouped result as a new data frame called df_by_state
missing_cols = [col for col in required_columns if col not in df.columns]
if missing_cols:
    print(f"Warning: Missing columns for grouping: {missing_cols}")
    df_by_state = None
else:
    df_by_state = df.groupby("Business State")[required_columns].agg(['mean',
'median', 'min', 'max'])
    print("Grouped Statistics by State:")
    print(df_by_state.head())
```

4. **Filtering negative debt-to-equity**
   If the "Debt to Equity" column exists, the program will extract any rows with a negative debt-to-equity ratio, debt-to-equity ratio less than zero, into a new dataframe (*df_negative_dte*).

```
# 4.FILTER df to find rows where "Debt to Equity" is less than 0
# STORE/PRINT this filtered result as df_negative_dte
df_negative_dte = df[df["Debt to Equity"] < 0]
if "Debt to Equity" in df.columns:
    df_negative_dte = df[df["Debt to Equity"] < 0]
    print("Businesses with Negative Debt to Equity:")
    print(df_negative_dte[["Business ID", "Business State", "Debt to
Equity"]].head())
else:
    print("Warning: 'Debt to Equity' column not found.")
    df_negative_dte = pd.DataFrame()
```

5. **Creating a new column**
   A new column is created to calculate "Debt to Income Ratio". This is calculated by dividing "Total Long-term Debt" by "Total Revenue" for each business. "NaN" is assigned if "Total Revenue" equals 0 to avoid division by 0.

```
# 5. CREATE a new column "Debt to Income Ratio"
# CALCULATE by dividing "Total Long-term Debt" by "Total Revenue" for each
row
if "Total Revenue" in df.columns and "Total Long-term Debt" in df.columns:
    df["Debt to Income Ratio"] = np.where(
        df["Total Revenue"] == 0,
        np.nan, # If Total Revenue is 0, assign NaN to avoid division by zero
        df["Total Long-term Debt"] / df["Total Revenue"]
    )
else:
    print("Warning: Missing columns for Debt to Income Ratio calculation.")
    df["Debt to Income Ratio"] = np.nan
```

6. **Concatenating the new column with the original column**
   The new "Debt to Income Ratio" is added to the main df, preserving all relevant metrics in one central location in the dataset.

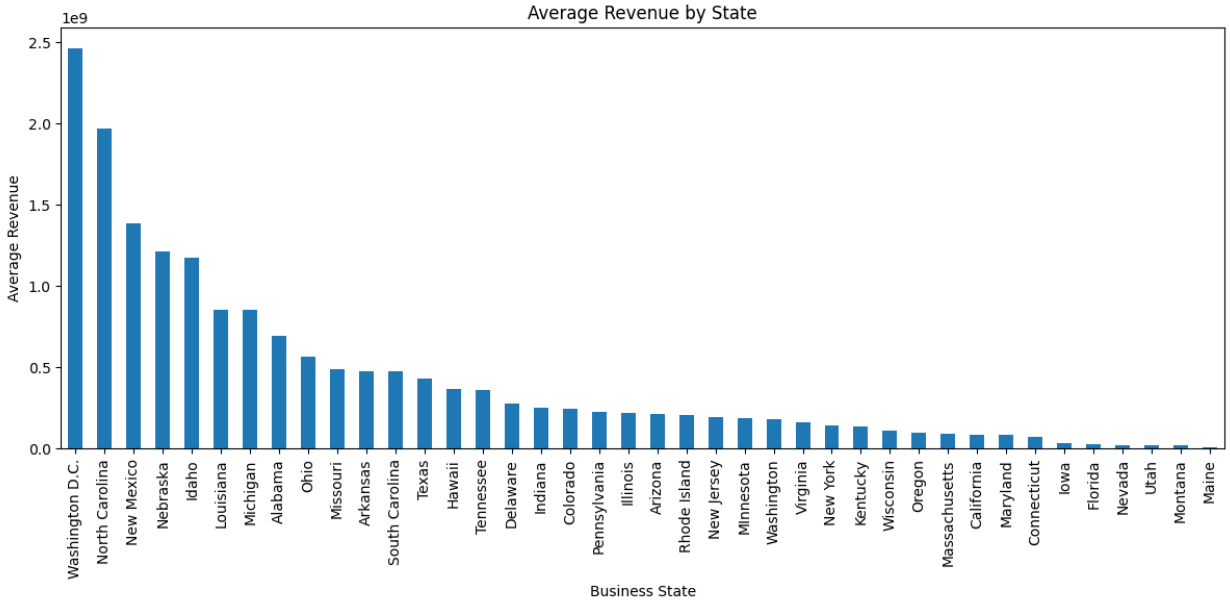
```
# 6.CONCATENATE the new column to the original df
# STORE the result as df_combined
df_combined = df.copy()
```

7. **Exporting the final dataset**
   The updated dataset is saved and exported as an Excel file (Updated_D598_Data_Set.xlsx) for further analysis or reporting.

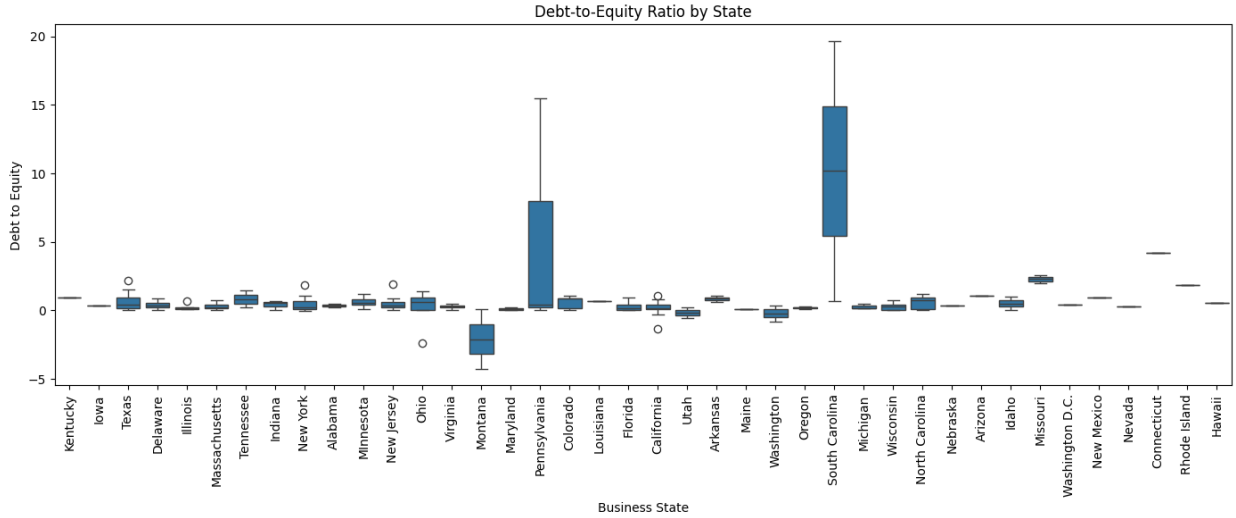
```
# 7.GENERATE updated dataset as output (e.g., export or prepare for
reporting)
df_combined.to_excel("Updated_D598_Data_Set.xlsx", index=False)

print("Updated Dataset with Debt-to-Income Ratio:")
print(df_combined.head())
```

B. Provide 4 customized data visualizations.

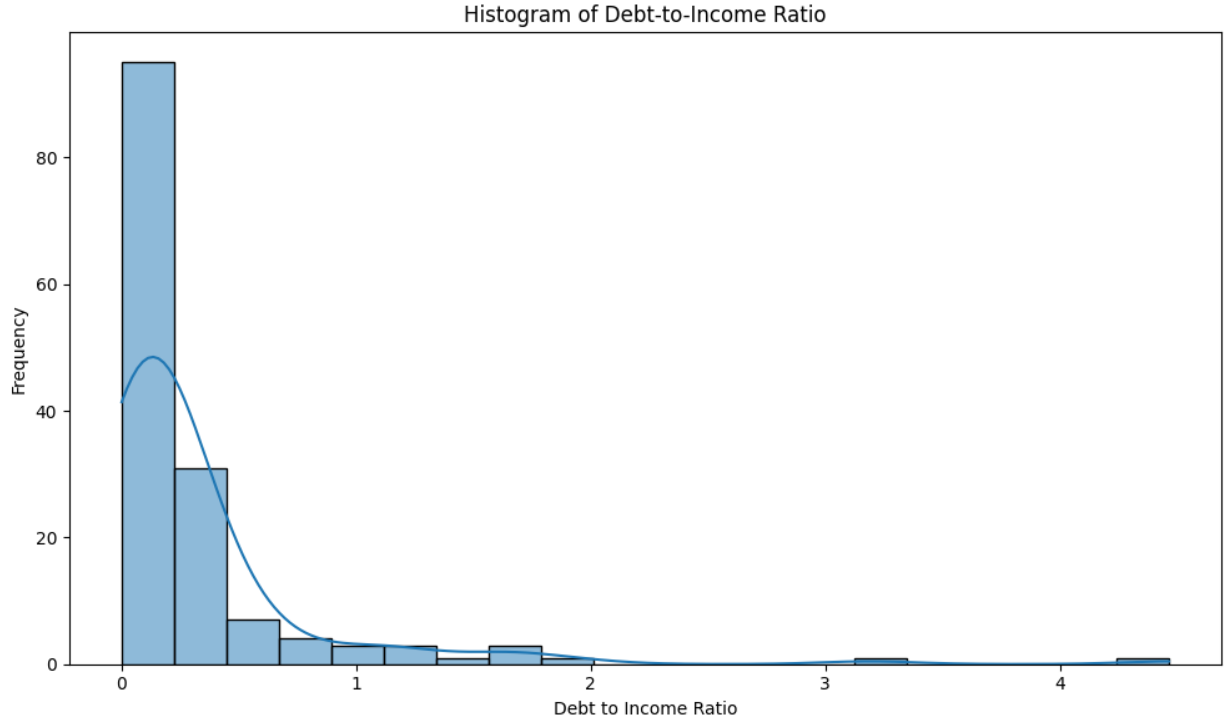
Bar Chart – Average Revenue by State



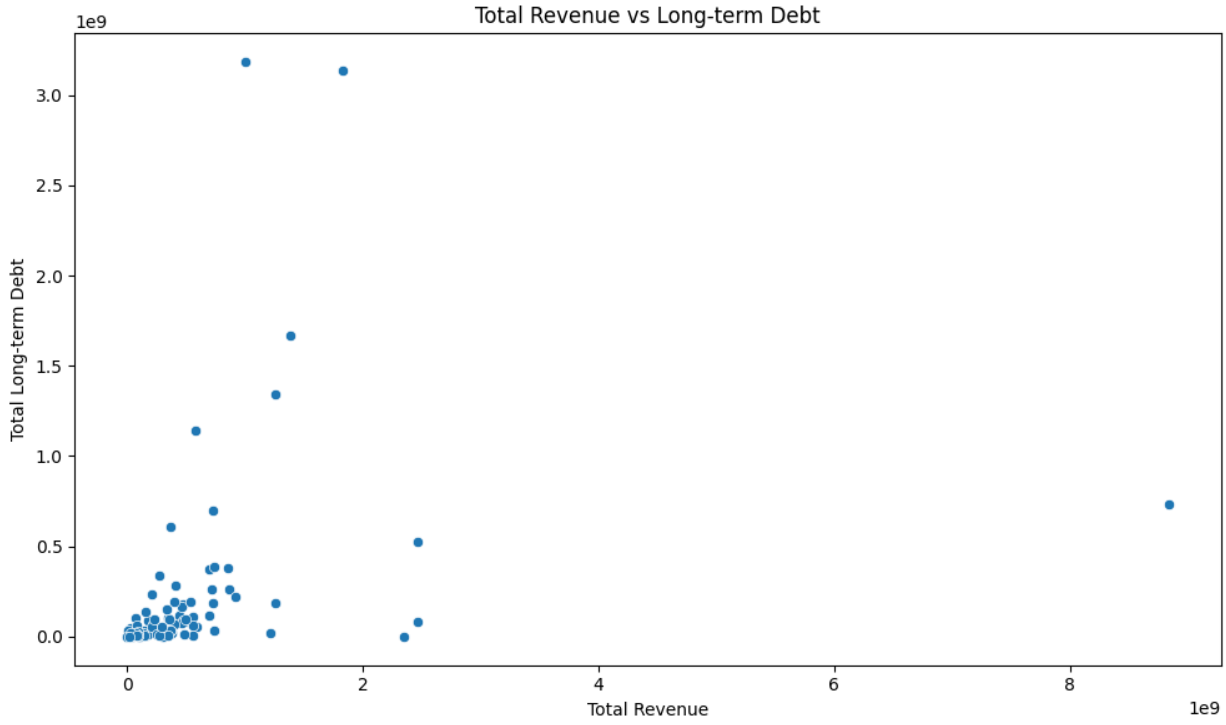
Box Plot – Debt-to-Equity Ratio by State



Histogram – Debt-to-Income Ratio



Scatter Plot – Total Revenue vs Long-term Debt



C. Explain how customized visualizations in part B were created.
First, there is Python code to set up packages, variables, and folders before creating the visualizations:

```
# Import packages to create data visualizations
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Load the dataset
df = pd.read_excel("D598_Data_Set.xlsx", sheet_name="1-150 V2")

# Create Debt to Income Ratio
df["Debt to Income Ratio"] = df["Total Long-term Debt"] / df["Total Revenue"]

# Create output folder for charts
output_dir = "task3_visuals"
os.makedirs(output_dir, exist_ok=True)
```

At the end of our program, we call a print statement that will output the visualizations as PNGs into a new folder:

```
# Export and save visualization outputs into a new folder
print(f"All charts saved to: {output_dir}/")
```

Bar Chart –
The bar chart was created by grouping the data by state, computing the mean (average) of "Total Revenue", and then using *matplotlib* to build out a sorted bar chart.
Python code:

```
# 1. Bar Chart – Average Revenue by State
plt.figure(figsize=(12,6))
avg_revenue = df.groupby("Business State")["Total
Revenue"].mean().sort_values(ascending=False)
avg_revenue.plot(kind='bar')
plt.title("Average Revenue by State")
plt.ylabel("Average Revenue")
plt.xticks(rotation=90)
plt.tight_layout()
plt.savefig(f"{output_dir}/avg_revenue_by_state.png")
plt.close()
# 2. Box Plot – Debt-to-Equity by State
plt.figure(figsize=(14,6))
sns.boxplot(data=df, x="Business State", y="Debt to Equity")
plt.title("Debt-to-Equity Ratio by State")
plt.xticks(rotation=90)
plt.tight_layout()
plt.savefig(f"{output_dir}/debt_to_equity_boxplot.png")
```

```
plt.close()
```

Box Plot –
The box plot was created by using *seaborn.boxplot()* to visualize debt-to-equity distributions across states for comparative analysis.
Python code:

```
# 2. Box Plot – Debt-to-Equity by State
plt.figure(figsize=(14,6))
sns.boxplot(data=df, x="Business State", y="Debt to Equity")
plt.title("Debt-to-Equity Ratio by State")
plt.xticks(rotation=90)
plt.tight_layout()
plt.savefig(f"{output_dir}/debt_to_equity_boxplot.png")
plt.close()
```

Histogram –
The histogram of Debt-to-Income Ratio was created by using *seaborn.histplot()* with KDE (this creates a smooth curve for a clearer visualization of the data). The histogram displays the frequency and distribution of Debt-to-Income ratios.
Python code:

```
# 3. Histogram – Debt-to-Income Ratio
plt.figure(figsize=(10,6))
sns.histplot(df["Debt to Income Ratio"].dropna(), bins=20, kde=True)
plt.title("Histogram of Debt-to-Income Ratio")
plt.xlabel("Debt to Income Ratio")
plt.ylabel("Frequency")
plt.tight_layout()
plt.savefig(f"{output_dir}/dti_histogram.png")
plt.close()
```

Scatter Plot –
The scatter plot was created using *seaborn.scatterplot()* to visually assess the correlation between total revenue and long-term debt.
Python code:

```
# 4. Scatter Plot – Revenue vs Long-term Debt
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x="Total Revenue", y="Total Long-term Debt")
plt.title("Total Revenue vs Long-term Debt")
plt.xlabel("Total Revenue")
plt.ylabel("Total Long-term Debt")
plt.tight_layout()
plt.savefig(f"{output_dir}/revenue_vs_debt_scatter.png")
plt.close()
```

D. Sources
No external sources outside of WGU coursework were used or referenced in this task.