Step 1: Import Data

```
Python > D598 - Task 1.py > ...
1   import pandas as pd
2
3   # Step 1: Import Data
4   data_file = 'C:/Users/kyleh/Downloads/D598 Data Set.xlsx'
5   df = pd.read_excel(data_file)
```

Purpose: Load the Excel data into a Pandas DataFrame (`df`).
Explanation: The `pd.read_excel` function reads the Excel file located at the specified path into a DataFrame, a structured, table-like data format used in Python for analysis.

Step 2: Identify Duplicate Rows

```
7   # Step 2: Identify Duplicate Rows
8   duplicates = df[df.duplicated()]
9   print("Duplicate Rows:")
10  print(duplicates)
```

Purpose: Identify and display rows in the dataset that are exact duplicates of other rows.
Explanation: The `duplicated()` method returns a Boolean Series indicating whether each row is a duplicate of a previous row. Rows where `True` is returned are filtered out and stored in the `duplicate` DataFrame for inspection.

Step 3: Group by State and Calculate Descriptive Statistics

```
12  # Step 3: Group by State and Calculate Descriptive Statistics
13  grouped_stats = df.groupby('Business State').agg({
14      'Total Long-term Debt': ['mean', 'median', 'min', 'max'],
15      'Total Equity': ['mean', 'median', 'min', 'max'],
16      'Total Liabilities': ['mean', 'median', 'min', 'max'],
17      'Total Revenue': ['mean', 'median', 'min', 'max'],
18      'Profit Margin': ['mean', 'median', 'min', 'max']
19  }).reset_index()
```

Purpose: Calculate summary statistics (mean, median, minimum, and maximum) for specific columns grouped by the `'Business State'` column.
Explanation:

- The `groupby` method groups the data by the `'Business State'` column.
- The `agg` method applies aggregation functions (mean, median, min, max) to the specified columns.
- The result is a new DataFrame (`grouped_stats`) containing these statistics for each state.

Step 4: Filter Businesses with Negative Debt-to-Equity Ratios

```
24  # Step 4: Filter Businesses with Negative Debt-to-Equity Ratios
25  negative_debt_to_equity = df[df['Debt to Equity'] < 0]
26  print("Businesses with Negative Debt-to-Equity Ratios:")
27  print(negative_debt_to_equity)
```

Purpose: Add a new column to the DataFrame that calculates the Debt-to-Income ratio for each business.
Explanation: This ratio is computed by dividing the `'Total Long-term Debt'` by `'Total Revenue'`, providing insight into how leveraged a business is relative to its income.

Step 5: Create Debt-to-Income Ratio

```
29  # Step 5: Create Debt-to-Income Ratio
30  df['Debt-to-Income'] = df['Total Long-term Debt'] / df['Total Revenue']
```

Purpose: Add a new column to the DataFrame that calculates the Debt-to-Income ratio for each business.
Explanation: This ratio is computed by dividing the `'Total Long-term Debt'` by `'Total Revenue'`, providing insight into how leveraged a business is relative to its income.

Step 6: Concatenate the New DataFrame

```
32  # Step 6: Concatenate the New DataFrame
33  # Creating a new DataFrame with only Debt-to-Income Ratio
34  debt_to_income_df = df[['Business ID', 'Debt-to-Income']]
35  final_df = pd.merge(df, debt_to_income_df, on='Business ID')
```

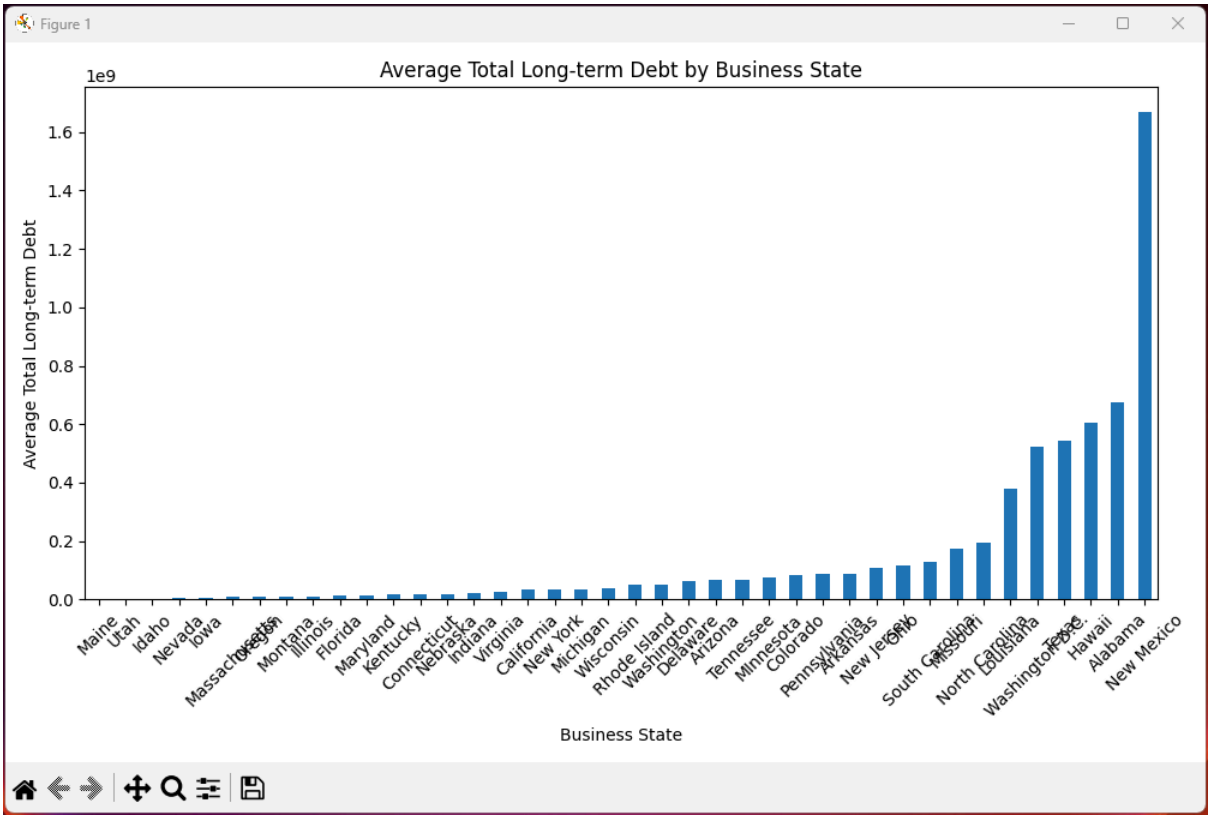
Purpose: Merge the existing DataFrame with a new DataFrame that includes only the `'Business ID'` and `'Debt-to-Income'` columns.
Explanation:

- The `merge` method combines `df` with `debt_to_income_df` using `'Business ID'` as the key.
- This ensures the final DataFrame (`final_df`) contains all original columns and retains the Debt-to-Income ratio as part of its structure.

Step 7: Save the Final DataFrame

```
37  # Step 7: Save the Final DataFrame
38  final_df.to_excel('Final_Analysis_Output.xlsx', index=False)
39  print("Analysis Completed. Results saved as 'Final_Analysis_Output.xlsx'.")
```

Purpose: Export the final DataFrame to a new Excel file.
Explanation: The `to_excel` method saves the DataFrame as an Excel file named `Final_Analysis_Output.xlsx` in the current working directory. The `index=False` argument ensures the row index is not included in the output.

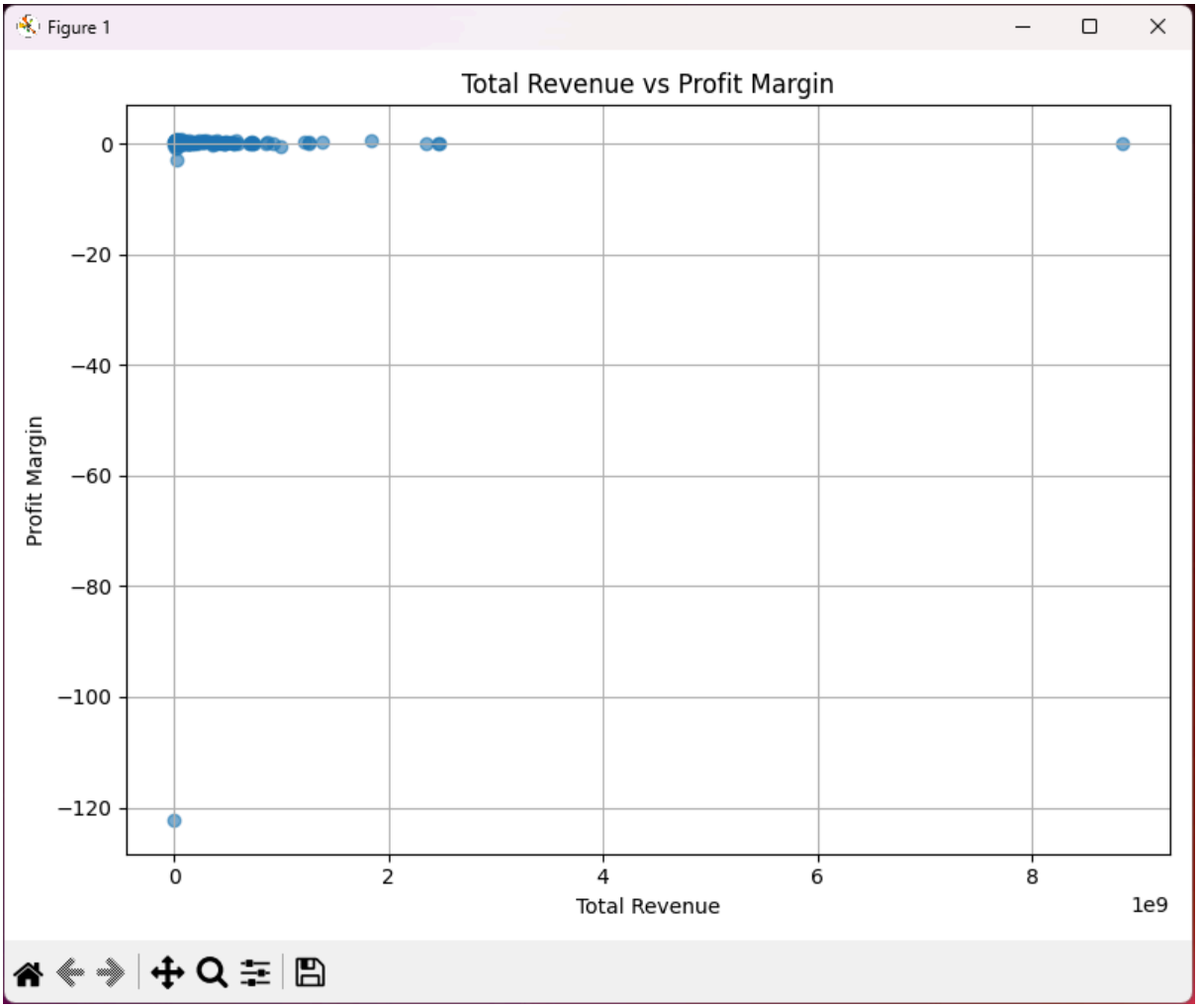


1. Bar Chart: Average Total Long-term Debt by Business State

```
: > Users > kyleh > OneDrive > Desktop > D598 - Task 1 > D598 - Dataset - Visualizations.py > ...
1   import pandas as pd
2   import matplotlib.pyplot as plt
3
4   # Load the dataset
5   file_path = 'C:/Users/kyleh/Downloads/D598 Data Set.xlsx'
6   df = pd.read_excel(file_path)
7
8   # 1. Bar Chart: Average Total Long-term Debt by Business State
9   avg_debt_by_state = df.groupby('Business State')['Total Long-term Debt'].mean().sort_values()
10  plt.figure(figsize=(10, 6))
11  avg_debt_by_state.plot(kind='bar', title='Average Total Long-term Debt by Business State')
12  plt.ylabel('Average Total Long-term Debt')
13  plt.xlabel('Business State')
14  plt.xticks(rotation=45)
15  plt.tight_layout()
16  plt.show()
```

Purpose: Visualize the average `Total Long-term Debt` for businesses in each state.
Steps:

1. The data was grouped by the column `'Business State'` using `groupby`.
2. Calculated the mean of the `'Total Long-term Debt'` for each group.
3. Sorted the values for better visual organization.
4. Plotted the data as a bar chart using the Pandas `plot` method.
5. Enhanced readability with labels, a title, and proper layout adjustments.



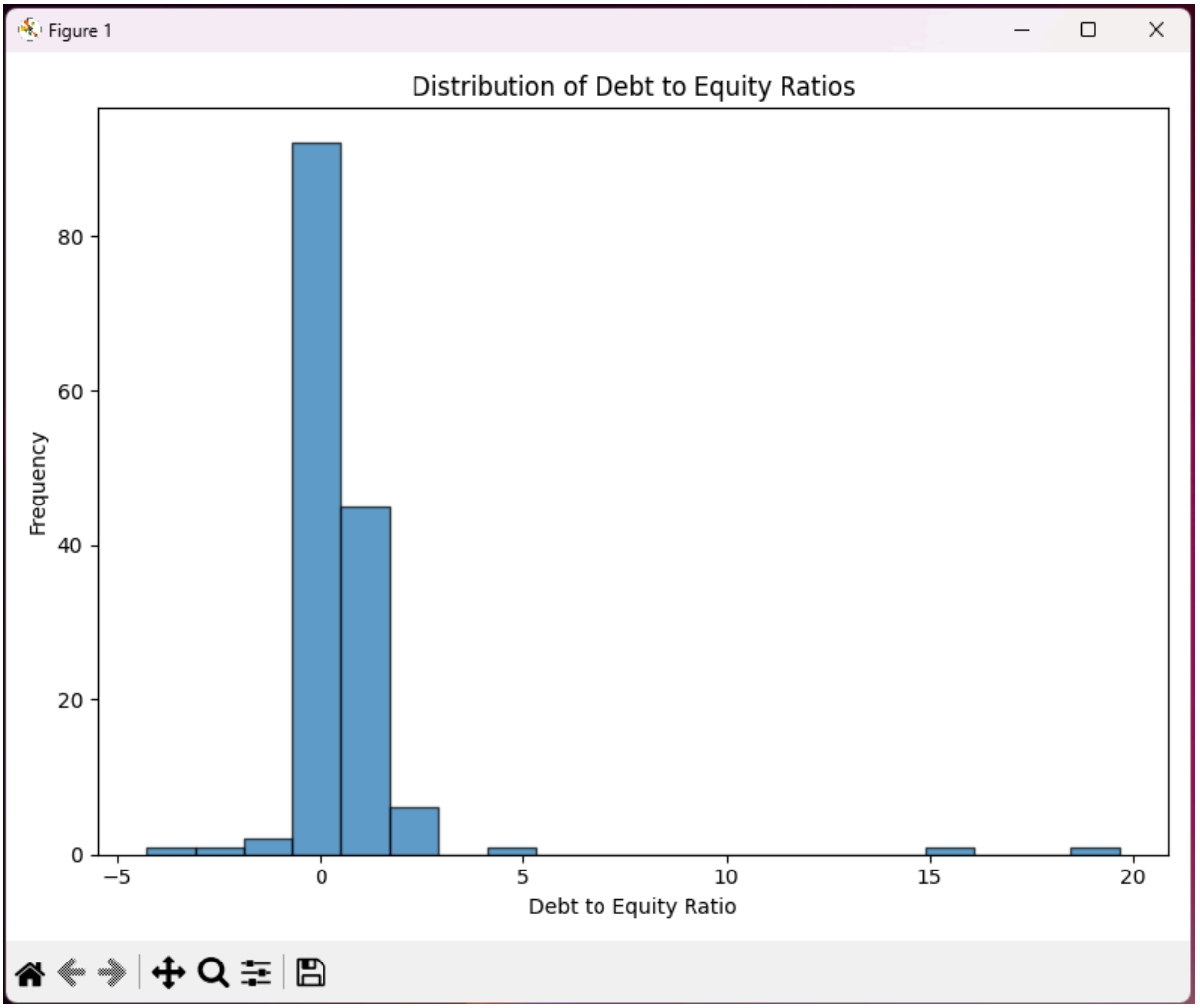
2. Scatter Plot: Relationship between Total Revenue and Profit Margin

```
# 2. Scatter Plot: Relationship between Total Revenue and Profit Margin
plt.figure(figsize=(8, 6))
plt.scatter(df['Total Revenue'], df['Profit Margin'], alpha=0.6)
plt.title('Total Revenue vs Profit Margin')
plt.xlabel('Total Revenue')
plt.ylabel('Profit Margin')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Purpose: Show the relationship between a business's `Total Revenue` and its `Profit Margin`.

Steps:

1. Used `plt.scatter` to create a scatter plot.
2. Mapped `Total Revenue` to the x-axis and `Profit Margin` to the y-axis.
3. Adjusted transparency with `alpha` for better visualization of overlapping points.
4. Added gridlines and axis labels for clarity.

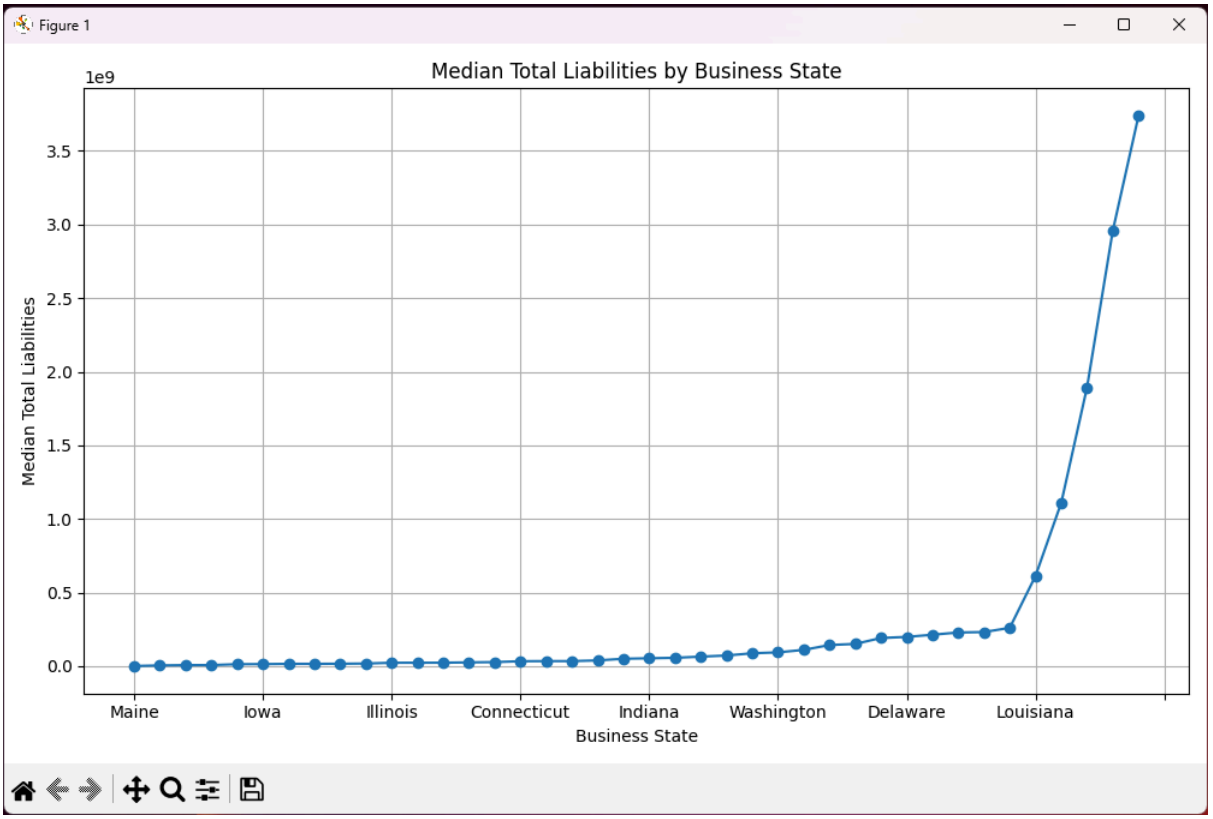


3. Histogram: Distribution of Debt-to-Equity Ratios

```
# 3. Histogram: Distribution of Debt to Equity Ratios
plt.figure(figsize=(8, 6))
plt.hist(df['Debt to Equity'], bins=20, edgecolor='black', alpha=0.7)
plt.title('Distribution of Debt to Equity Ratios')
plt.xlabel('Debt to Equity Ratio')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

Purpose: Display the frequency distribution of `Debt to Equity` ratios across businesses.
Steps:

1. Used `plt.hist` to create a histogram of the `'Debt to Equity'` column.
2. Set the number of bins to 20 for more granularity.
3. Added edge color and adjusted transparency for better aesthetics.
4. Enhanced with labels and a title for better interpretation.



4. Line Chart: Median Total Liabilities for Each Business State

```
# 4. Line Chart: Median Total Liabilities for each Business State
median_liabilities_by_state = df.groupby('Business State')['Total Liabilities'].median().sort_values()
plt.figure(figsize=(10, 6))
median_liabilities_by_state.plot(kind='line', marker='o', title='Median Total Liabilities by Business State')
plt.ylabel('Median Total Liabilities')
plt.xlabel('Business State')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Purpose: Show the median `Total Liabilities` for businesses in each state as a trend over the sorted states.
Steps:

1. Grouped the data by `'Business State'` and calculated the median of `'Total Liabilities'`.
2. Sorted the results for logical ordering.
3. Used a line chart (`kind='line'`) with markers (`marker='o'`) to make data points visible.
4. Added gridlines, labels, and a title for better readability.

References

- The Pandas Development Team. (n.d.). *Group by: Split-apply-combine*. Retrieved January 6, 2025, from https://pandas.pydata.org/docs/user_guide/groupby.html
- The Matplotlib Development Team. (n.d.). *Pyplot tutorial*. Retrieved January 6, 2025, from https://matplotlib.org/stable/tutorials/introductory/pyplot.html
- D598 Course Materials. (2025). *Master of Data Engineering Task 2 assignment: Financial data analysis and visualizations.* Instructional content provided by Western Governors University.
- Colby, K. (2025). *Python code for data analysis and visualizations* [Personal knowledge and experience].