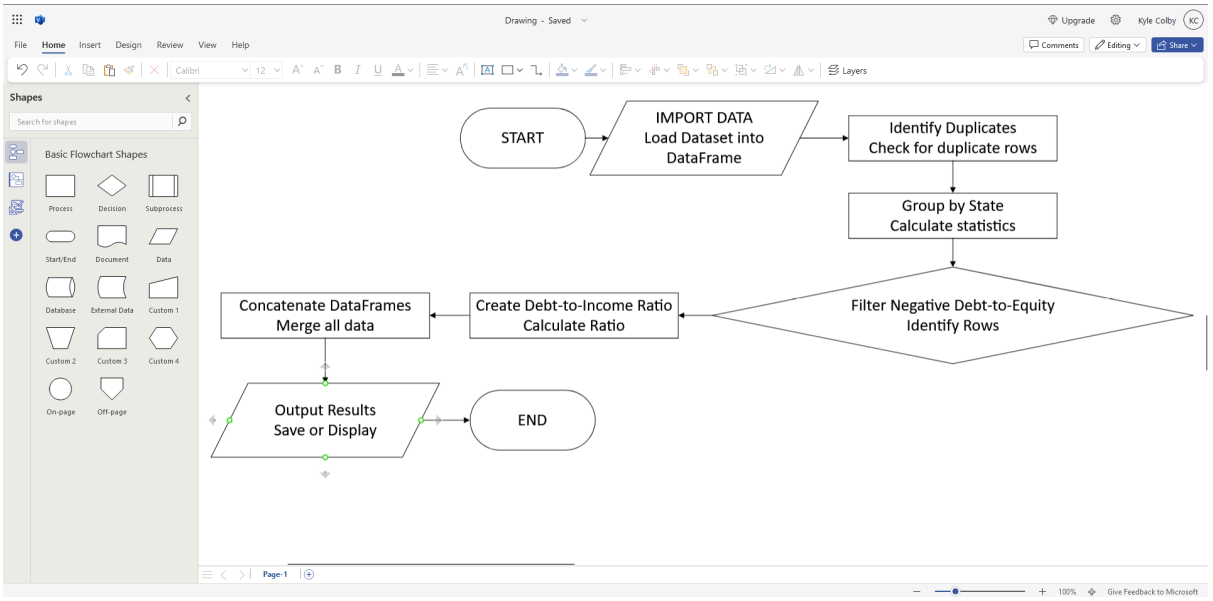

Flowchart for Equity Fund Analysis

1. **Start**
    - Initialize the process.
2. **Import Data**
    - Load the dataset into a DataFrame.
3. **Identify Duplicates**
    - Check for duplicate rows in the DataFrame.
4. **Group by State**
    - Group businesses by state.
    - For all numeric columns, calculate descriptive statistics (mean, median, min, max).
    - Store the results in a new DataFrame.
5. **Filter for Negative Debt-to-Equity Ratios**
    - Filter rows where the Debt-to-Equity ratio is negative.
    - Create a new DataFrame with these rows.
6. **Create Debt-to-Income Ratio**
    - Calculate the Debt-to-Income ratio for each business:
        - Formula: Total Long-term Debt / Total Revenue.
    - Store this data in a new DataFrame.
7. **Concatenate DataFrames**
    - Merge the Debt-to-Income ratio DataFrame with the original DataFrame.
8. **Output Results**
    - Save or display the results for further analysis.
9. **End**
    - Terminate the program.

    a.

Pseudocode for Equity Fund Analysis

1. **Start**

- Initialize the script and set up the environment.

2. **Import Libraries and Dataset**
   - Import necessary libraries: pandas, numpy.
   - Load the dataset into a pandas DataFrame.
3. **Check for Duplicates**
   - Use `DataFrame.duplicated()` to identify duplicate rows.
   - Print or store the duplicate rows for review.
4. **Group by State and Calculate Statistics**
   - Group the DataFrame by the 'Business State' column using `groupby()`.
   - Calculate mean, median, min, and max for numeric columns.
   - Store the grouped statistics in a new DataFrame.
5. **Filter for Negative Debt-to-Equity Ratios**
   - Use a condition to filter rows where 'Debt to Equity' < 0.
   - Save the filtered rows into a new DataFrame.
6. **Calculate Debt-to-Income Ratio**
   - Create a new column in the original DataFrame:
     - Formula: `Debt-to-Income = Total Long-term Debt / Total Revenue`.
   - Handle cases where Total Revenue is zero to avoid division errors.
7. **Merge DataFrames**
   - Concatenate the Debt-to-Income ratio column with the original DataFrame.
8. **Save or Display Results**
   - Export the final DataFrame to a CSV or Excel file.
   - Print key results for validation.
9. **End**
   - Complete the script execution and exit.

Python > D598 - Task 1.py > ...

```python
import pandas as pd

# Step 1: Import Data
data_file = 'C:/Users/kyleh/Downloads/D598 Data Set.xlsx'
df = pd.read_excel(data_file)

# Step 2: Identify Duplicate Rows
duplicates = df[df.duplicated()]
print("Duplicate Rows:")
print(duplicates)

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

print("Grouped Statistics by State:")
print(grouped_stats)

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

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

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

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

```

Explanation of Flowchart and Pseudocode

1. **Logic in the Flowchart**:
    - The flowchart provides a high-level visual representation of the program's steps, from initializing the process to outputting results. It is designed to simplify the sequence of operations.
    - Each step in the flowchart corresponds to a logical task, such as importing data, checking for duplicates, grouping by state, and calculating statistics.
2. **Alignment with Pseudocode**:
    - The pseudocode provides detailed instructions for each step in the flowchart. For example:
        - The flowchart mentions "Import Data," the pseudocode specifies importing pandas and reading the dataset using `pd.read_excel()`.
        - The flowchart includes "Group by State," and the pseudocode details how to use `group by ()` to calculate descriptive statistics.
    - Each flowchart component maps directly to a corresponding segment in the pseudocode, ensuring a clear and logical flow from visual representation to implementation.
3. **Key Relationship**:
    - The flowchart serves as a blueprint for the pseudocode, breaking down the tasks into actionable items. The pseudocode translates the flowchart's high-level steps into structured, executable logic.

Course |. (n.d.). https://apps.cgp-oex.wgu.edu/wgulearning/course/course-v1:WGUx+OEX0343+v01/block-v1:WGUx+OEX0343+v01+type@sequential+block@543516264ee84ad4b89a1449a5d2db90/block-v1:WGUx+OEX0343+v01+type@vertical+block@63e2765421dc476a8611a55973cafd50

3.13.1 documentation. (n.d.). https://docs.python.org/3/

Documentation for Visual Studio code. (2021, November 3). https://code.visualstudio.com/Docs