Capstone Project

"Comprehensive Analysis of Fortune 1000 Companies: Unveiling Insights from Revenue, Profit Change, and Workforce Dynamics"

Members: Sania Bandekar, Brendan Dishion & Nathan Pham

Objective

The project aims to perform a comprehensive **analysis** of the "Fortune 1000 Companies by Revenue.csv" **dataset** using custom **Bash scripts**. The initial phase involves **extracting metadata**, including the **total number of entries**, **data types** of each **column**, **column headers**, **count of missing values**, **unique values per column**, and the **minimum-maximum value ranges** in specified columns. Subsequently, the project delves into deriving **nontrivial insights** by **calculating correlations**. Specifically, it explores the **relationship** between the **percentage change in profits** and **market value**, the **Pearson correlation coefficient** between **revenues** and the **number of employees**, and the **correlation** between **revenues** and **profits for the listed companies**. The ultimate **goal** is to provide **users with a deeper understanding** of the **dataset's characteristics and interdependencies**, facilitating **informed decision-making** and **data-driven insights**.

https://raw.githubusercontent.com/nghiapham1026/cs131/main/MiniProjectGroup10/data/Fortune%201000%20Companies%20by%20Revenue.csv

Dataset-

```
rank ,name ,revenues ,revenue_percent_change,profits ,profits_percent_change,assets,market_value ,change_in_rank,employees
1,Walmart,"$572,754 ",2.40%,"$13,673 ",1.20%,"$244,860 ","$409,795 ",-,"2,300,000"
2,Amazon,"$469,822 ",21.70%,"$33,364 ",56.40%,"$420,549 ","$1,658,807.30 ",-,"1,608,000"
3,Apple,"$365,817 ",33.30%,"$94,680 ",64.90%,"$351,002 ","$2,849,537.60 ",-,"154,000"
4,CVS Health,"$292,111 ",8.70%,"$7,910 ",10.20%,"$232,999 ","$132,839.20 ",-,"258,000"
5,UnitedHealth Group,"$287,597 ",11.80%,"$17,285 ",12.20%,"$212,206 ","$479,830.30 ",-,"350,000"
6,Exxon Mobil,"$285,640 ",57.40%,"$23,040 ",-,"$338,923 ","$349,652.40 ",4,"63,000"
7,Berkshire Hathaway,"$276,094 ",12.50%,"$89,795 ",111.20%,"$958,784 ","$779,542.30 ",-1,"372,000"
8,Alphabet,"$257,637 ",41.20%,"$76,033 ",88.80%,"$359,268 ","$1,842,326.10 ",1,"156,500"
9,McKesson,"$238,228 ",3.10%,"($4,539)",-604.30%,"$65,015 ","$45,857.80 ",-2,"67,500"
10,AmerisourceBergen,"$213,988.80 ",12.70%,"$1,539.90 ",-,"$57,337.80 ","$32,355.70 ",-2,"40,000"
11,Costco Wholesale,"$195,929 ",17.50%,"$5,007 ",25.10%,"$59,268 ","$255,230.70 ",1,"288,000"
12,Cigna,"$174,078 ",8.50%,"$5,365 ",-36.60%,"$154,889 ","$76,286.30 ",1,"72,963"
```

Metadata #1

- **Idea of Script**: To count the total number of lines in the "Fortune 1000 Companies by Revenue.csv" file, which represents the total number of entries, and then subtract one to exclude the header row.
- **Execution Steps**:
  - Assign the result of `wc -l` (word count for lines) command to the variable `entries`. This command counts all lines in the CSV file.
  - Use `cut -d' ' -f1` to extract just the number part of the `wc -l` output.

```
#!/bin/bash
# Count the total number of entries in the CSV, subtracting the header row
entries=$(wc -l "./data/Fortune 1000 Companies by Revenue.csv" | cut -d' ' -f1)
echo "Total entries: $((entries - 1))"
```

```
nathan1026@LAPTOP-NCPS0E0P:/mnt/c/Users/nghia/OneDrive/Documents/GitHub/cs131/MiniProjectGroup10$ ./scripts/metadata/count_entries.sh
Total entries: 1000
```

Metadata #2

- **Idea of Script**: To infer the data types of each column in the "Fortune 1000 Companies by Revenue.csv" by examining the contents of the second row (assuming the first row is the header).
- **Execution Details**:
  - The script uses `awk` to analyze the file.
  - It sets the field separator to a comma (`,`), as it's dealing with a CSV file.
  - `NR==2` instructs `awk` to only consider the second row of the file.
  - The `for` loop iterates over each field (`$i`) in the row.
  - The `if` and `else if` statements use regular expressions to test if the field is an integer, a float, or neither.

```
Inferring data types for each column...
Column 1: Integer
Column 2: String
Column 3: String
Column 4: String
Column 5: String
Column 6: String
Column 7: String
Column 8: String
Column 9: String
Column 10: String
Column 11: String
Column 12: String
Column 13: String
Column 14: String
Column 15: Integer
Column 16: String
```

```
#!/bin/bash

echo "Inferring data types for each column..."
awk -F, 'NR==2 {
  for (i=1; i<=NF; i++) {
    if ($i ~ /^[+-]?[0-9]+$/)
      printf "Column %d: Integer\n", i;
    else if ($i ~ /^[+-]?[0-9]+(\.[0-9]+)?$/)
      printf "Column %d: Float\n", i;
    else
      printf "Column %d: String\n", i;
  }
}' "./data/Fortune 1000 Companies by Revenue.csv"
```

Metadata #3

- **Idea of Script**: To extract and display the column headers from the "Fortune 1000 Companies by Revenue.csv" file, which indicate the features included in the dataset.
- **Execution Details**:
  - The script uses the `head` command to output the first line of the CSV file.
  - `-1` option tells `head` to select only the first line, which typically contains the column names in a CSV file.

```
#!/bin/bash
# Display the first line of the CSV file to show column headers
head -1 "./data/Fortune 1000 Companies by Revenue.csv"

#./scripts/metadata/features.sh
```

```
nathan1026@LAPTOP-NCPS0E0P:/mnt/c/Users/nghia/OneDrive/Documents/GitHub/cs131/MiniProjectGroup10$ ./scripts/metadata/features.sh
rank ,name ,revenues ,revenue percent change,profits ,profits percent change,assets,market value ,change in rank,employees
```

Metadata #4

- **Idea of Script**: To count the number of unique values present in each column of the "Fortune 1000 Companies by Revenue.csv" file.
- **Execution Details**:
  - It uses a `for` loop to iterate over each column in the CSV file.
  - `seq` generates a sequence of numbers from 1 to the number of columns, which is determined by counting the commas in the header row.
  - For each column, `cut` extracts the column's data, `sort` arranges the lines, `uniq` filters out duplicate lines, and `wc -l` counts the remaining unique lines.

```
Counting unique values for each column...
Column 1:
998
Column 2:
1001
Column 3:
104
Column 4:
900
Column 5:
557
Column 6:
648
Column 7:
726
Column 8:
351
Column 9:
770
```

```bash
#!/bin/bash

echo "Counting unique values for each column..."
for i in $(seq 1 $(head -1 "./data/Fortune 1000 Companies by Revenue.csv" | grep -o "," | wc -l)); do
  echo "Column $i:"
  cut -d, -f$i "./data/Fortune 1000 Companies by Revenue.csv" | sort | uniq | wc -l
done
```

Metadata #5

- **Idea of Script**: This script is designed to find the minimum and maximum values in a specified column of the "Fortune 1000 Companies by Revenue.csv" file.
- **Execution Details**:
  - `cut` is used to extract the specified column's data.
  - `sed '1d'` removes the header row from the output.
  - `tr -d '"'` removes any double-quote characters that may be present.
  - `sed 's/\$//g'` strips out dollar sign characters.
  - `tr -d ','` removes commas, which are often used as thousand separators in numbers.
  - `sort -n` sorts the numbers in ascending order.
  - `head -1` and `tail -1` are used to get the minimum and maximum values, respectively.

```
#!/bin/bash
# Print min and max values for a specified column number
column=$1
min=$(cut -d',' -f$column "./data/Fortune 1000 Companies by Revenue.csv" | sed '1d' | tr -d '"' | sed 's/\$//g' | tr -d ',' | sort -n | head -1)
max=$(cut -d',' -f$column "./data/Fortune 1000 Companies by Revenue.csv" | sed '1d' | tr -d '"' | sed 's/\$//g' | tr -d ',' | sort -n | tail -1)
echo "Column $column: Min value is $min, Max value is $max"

#Run ./scripts/metadata/value_ranges.sh 2
#Change the last number to the column you want to extract
```

Nontrivial #1

- **Idea of Script**: To calculate the correlation coefficient between the percent change in profits and the market value of companies listed in the "Fortune 1000 Companies by Revenue.csv" file.
- **Execution Details**:
  - The `pd.to_numeric` function is used to convert the 'profits_percent_change' and 'market_value' columns to numeric types, with non-numeric strings converted to `NaN` (using `errors='coerce'` to avoid conversion errors).
  - `regex=True` is used within the `replace` function to apply regular expressions for removing percentage signs and currency symbols.
  - The `corr` method from pandas is used to calculate the Pearson correlation coefficient between the two columns, which measures the linear relationship between them.

```bash
#!/bin/bash

# Path to the CSV file
file_path="./data/Fortune 1000 Companies by Revenue.csv"

# Use Python to calculate the correlation
python3 - <<EOF
import pandas as pd

df = pd.read_csv("$file_path")
# Replace non-numeric strings with NaN and convert the column to floats
df['profits_percent_change'] = pd.to_numeric(df['profits_percent_change'].replace('[\%,]', '', regex=True), errors='coerce')
df['market_value '] = pd.to_numeric(df['market_value '].replace('[\$,]', '', regex=True), errors='coerce')

# Calculate the correlation, ignoring NaN values
correlation = df['profits_percent_change'].corr(df['market_value '])
print(f"Correlation between Profits Percent Change and Market Value: {correlation}")
EOF

#chmod +x ./scripts/nontrivial/correlation_profit_change_market_value.sh
#./scripts/nontrivial/correlation_profit_change_market_value.sh
```

```
Correlation between Profits Percent Change and Market Value: -0.021741427111873768
```

Nontrivial #2

- **Idea of Script**: To find the correlation between the revenues and the number of employees of companies listed in the "Fortune 1000 Companies by Revenue.csv" file.
- **Execution Details**:
  - The revenues and employees columns are cleaned using `str.replace` to remove commas and dollar signs, which are common in currency representations.
  - The `pd.to_numeric` function with `errors='coerce'` parameter converts the cleaned string values to numeric, turning non-numeric strings into NaNs (useful for handling missing data).
  - The `corr` function from pandas is used to compute the Pearson correlation coefficient between the revenues and employees columns, providing a measure of the linear relationship between the two variables.

```
#!/bin/bash

# Path to the CSV file, assuming the script is run from the MiniProjectGroup10 directory
file_path="data/Fortune 1000 Companies by Revenue.csv"

# Use Python to calculate the correlation
python3 - <<EOF
import pandas as pd

df = pd.read_csv("$file_path")

df['revenues '] = pd.to_numeric(df['revenues '].str.replace('[\$,]', '', regex=True), errors='coerce')
df['employees '] = pd.to_numeric(df['employees '].str.replace('[\$,]', '', regex=True), errors='coerce')

correlation = df['revenues '].corr(df['employees '])
print(f"Correlation between Revenues and Employees: {correlation}")
EOF

#chmod +x ./scripts/nontrivial/correlation_revenue_employees.sh
#./scripts/nontrivial/correlation_revenue_employees.sh
```

```
Correlation between Revenues and Employees: 0.7322144580826426
```

Nontrivial #3

```
#!/bin/bash

# Path to the CSV file
file_path="./data/Fortune 1000 Companies by Revenue.csv"

# Use Python to calculate the correlation
python3 - <<EOF
import pandas as pd
import numpy as np

# Function to convert currency strings to floats
def convert_currency(val):
    if val == '-':  # Check if the value is a dash, which indicates missing data
        return np.nan
    new_val = val.replace(',', '').replace('$', '').replace('(', '-').replace(')', '')
    try:
        return float(new_val)
    except ValueError:  # In case there are any other strings that cannot be converted
        return np.nan

# Read the CSV file
df = pd.read_csv("$file_path")

# Clean the column names by stripping whitespace
df.columns = df.columns.str.strip()

# Apply the conversion function to the revenues and profits columns
df['revenues'] = df['revenues'].apply(convert_currency)
df['profits'] = df['profits'].apply(convert_currency)

# Calculate the correlation, ignoring NaN values
correlation = df['revenues'].corr(df['profits'])
print(f"Correlation between Revenues and Profits: {correlation}")
EOF
```

Nontrivial #3

- **Idea of Script**: To calculate the correlation between company revenues and profits from the "Fortune 1000 Companies by Revenue.csv" file.
- **Execution Details**:
  - The `convert_currency` function is defined to convert currency-formatted strings (including handling of missing values represented by '-') to float values.
  - Column names are cleaned of any leading/trailing whitespace to avoid issues with column name mismatches.
  - The `convert_currency` function is applied to the 'revenues' and 'profits' columns to clean and convert the data to numeric format, handling missing values by converting them to `np.nan`.
  - The Pearson correlation coefficient between the 'revenues' and 'profits' columns is calculated using the pandas `corr` method.

```
Correlation between Revenues and Profits: 0.6513867739810506
```

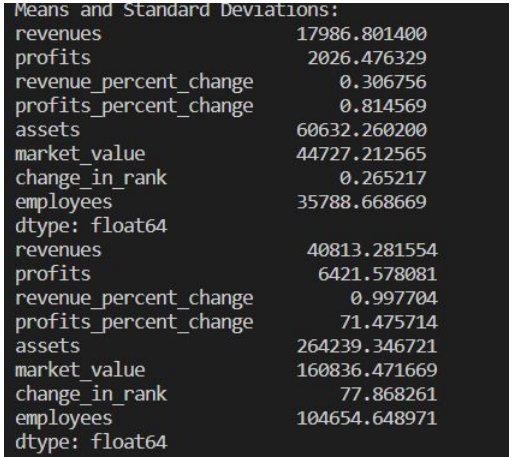
Nontrivial #4

```
#!/bin/bash

# Path to the CSV file
file_path="./data/Fortune 1000 Companies by Revenue.csv"

# Use Python to calculate mean and standard deviation
python3 - <<EOF
import pandas as pd
import numpy as np

# Function to convert strings with commas and potential negative signs to float
def convert_to_float(val):
    if isinstance(val, str):
        val = val.replace(',', '').replace('$', '').replace('(', '-').replace(')', '')
        if val.strip() == '-' or val.strip() == '':
            return np.nan  # Convert missing value indicators to NaN
        try:
            return float(val)
        except ValueError:
            return np.nan  # Convert other non-numeric strings to NaN
    return val

# Convert percentage strings to floats
def convert_percentage(val):
    if isinstance(val, str):
        val = val.replace('%', '').replace('(', '-').replace(')', '')
        if val.strip() == '-' or val.strip() == '':
            return np.nan
        try:
            return float(val) / 100
        except ValueError:
            return np.nan
    return val

# Read the CSV file
df = pd.read_csv("$file_path")
```

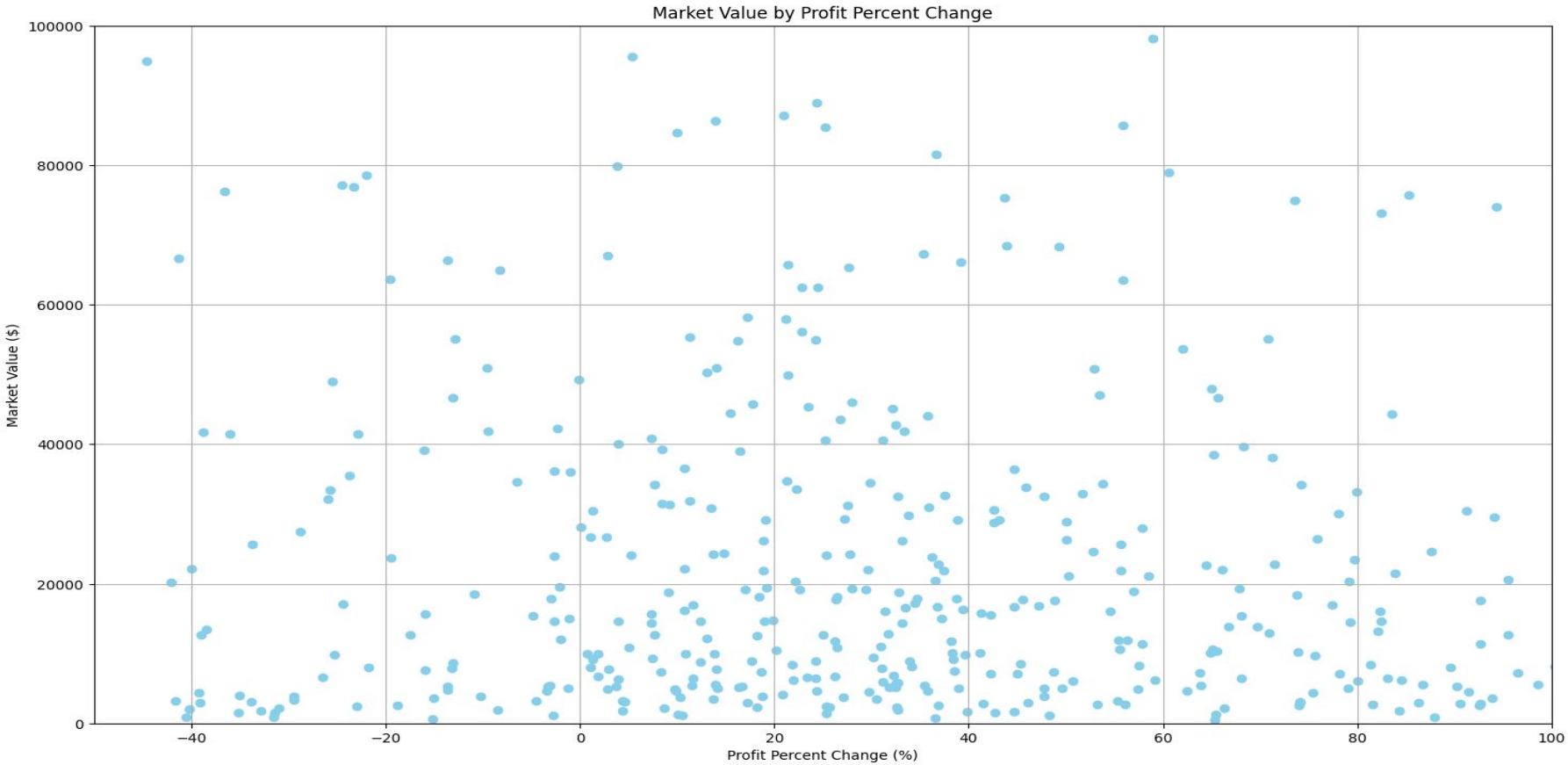
```
# Clean the columns and convert to float
df['revenues '] = df['revenues '].apply(convert_to_float)
df['profits '] = df['profits '].apply(convert_to_float)
df['revenue_percent_change'] = df['revenue_percent_change'].apply(convert_percentage)
df['profits_percent_change'] = df['profits_percent_change'].apply(convert_percentage)
df['assets'] = df['assets'].apply(convert_to_float)
df['market_value '] = df['market_value '].apply(convert_to_float)
df['change_in_rank'] = pd.to_numeric(df['change_in_rank'], errors='coerce')  # Convert directly, handling errors
df['employees '] = pd.to_numeric(df['employees '].str.replace(',', ''), errors='coerce')  # Remove commas and conver

# Handle missing values in 'change_in_rank'
df['change_in_rank'] = pd.to_numeric(df['change_in_rank'].replace('-', np.nan), errors='coerce')

# Calculate the mean and standard deviation for each column
means = df[['revenues ', 'profits ', 'revenue_percent_change', 'profits_percent_change',
            'assets', 'market_value ', 'change_in_rank', 'employees ']].mean()
std_devs = df[['revenues ', 'profits ', 'revenue_percent_change', 'profits_percent_change',
               'assets', 'market_value ', 'change_in_rank', 'employees ']].std()

# Print out the results
print("Means and Standard Deviations:")
print(means)
print(std_devs)
EOF
```

Nontrivial #4

- **Idea of Script**: The script aims to compute the mean (average) and standard deviation (a measure of the amount of variation or dispersion) for selected numerical columns in the "Fortune 1000 Companies by Revenue.csv" file.
- **Execution Details**:
  - Defines two functions in Python to handle currency and percentage conversion:
    - `convert_to_float` cleans and converts currency-formatted strings to float, accounting for missing or special characters.
    - `convert_percentage` converts percentage strings into float values, handling special cases and missing values.
  - Applies the conversion functions to appropriate columns to standardize the data for numerical operations.
  - Calculates the mean and standard deviation for the cleaned columns using pandas methods.



Market Value ($) vs Profit Change (%) - Plot 1



Input file created using linux commands - Plot 1

- awk 'BEGIN {FPAT = "\"[^\"]+\"|[^,]+"; OFS=","} {gsub(/\"/,"",$6); gsub(/\"/,"",$8); print $6, $8}' "Fortune 1000 Companies by Revenue.csv" > processed.csv
  - This command invokes awk
  - FPAT tells awk how to recognize fields in the input
    - Sets regular expression that matches either a quote or sequence of characters
    - `”[^\”]+”` - quote followed by any characters except a quote, followed by a quote
    - `[^,]+` - sequence of characters that don’t include comma
  - OFS=”,” - sets output field separator to comma
  - gsub(/\"/,"",$6) - replaces all quotation symbols in 8th column
  - gsub(/\"/,"",$8) - replaces all quotation symbols in 8th column
  - print $6, $8 - prints 6th and 8th columns
  - Output is directed to processed.csv

Script for Market Value vs. Profit Change - Plot 1

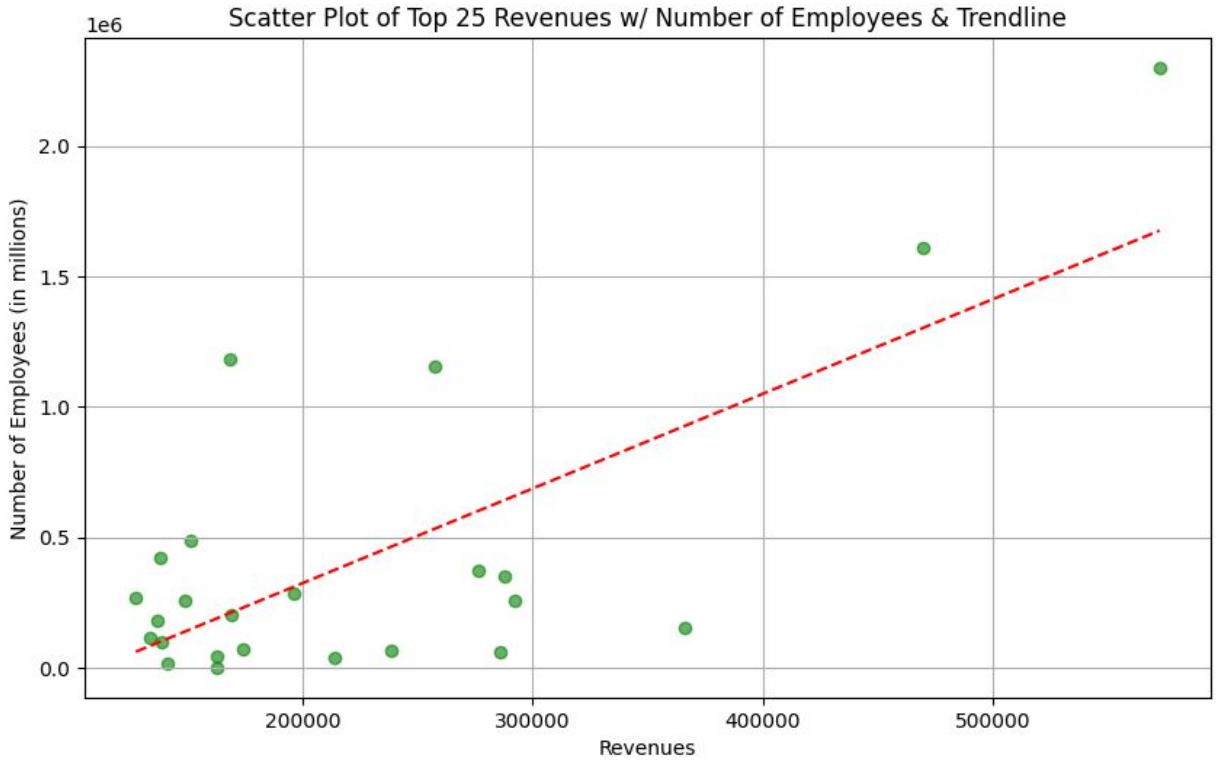
Top 10 columns/rows from processed.csv, the file continues for 990 other entries. This only shows the first 10 entries for the processed.csv file that we created using linux commands

```
profits_percent_change,market_value
1.20%,"$409,795 "
56.40%,"$1,658,807.30 "
64.90%,"$2,849,537.60 "
10.20%,"$132,839.20 "
12.20%,"$479,830.30 "
-,"$349,652.40 "
111.20%,"$779,542.30 "
88.80%,"$1,842,326.10 "
604.30%,"$45,857.80 "
-,"$32,355.70 "
```

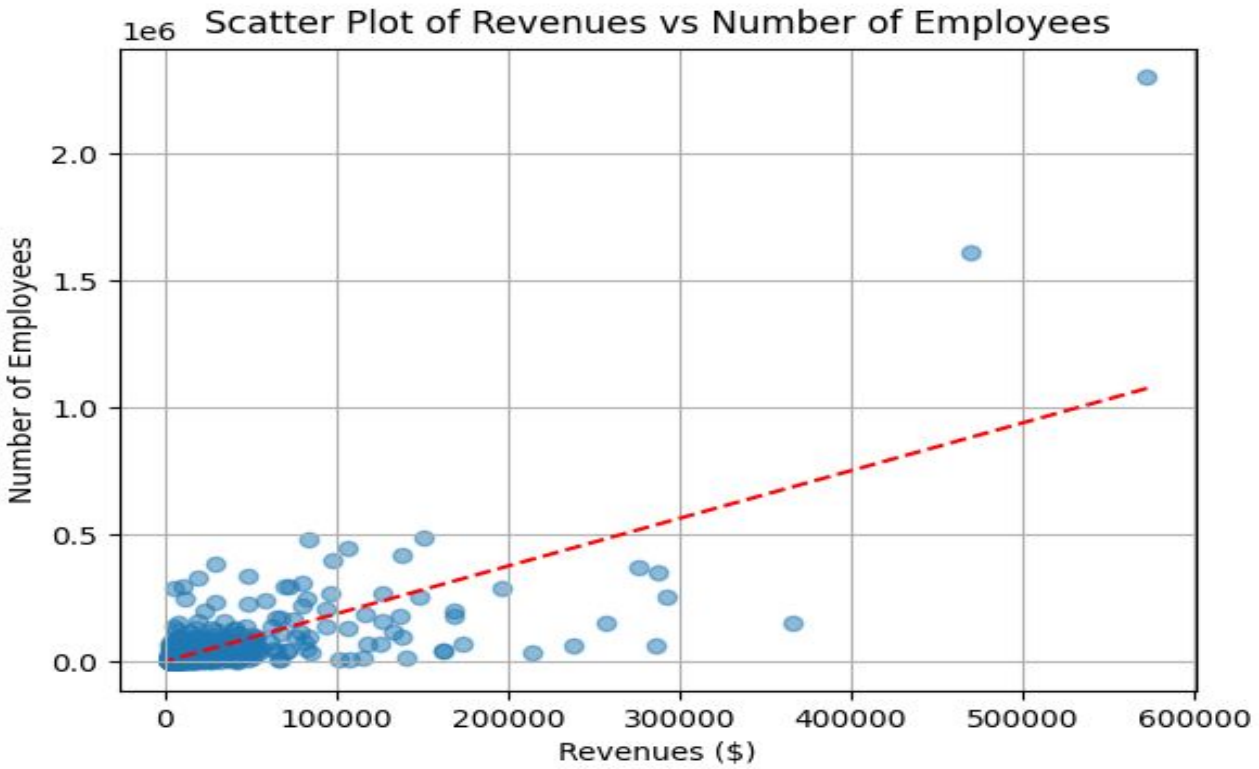
Python script named visualize_data.py plots the profits_percent_change & the market value using a scatter plot. The script reads in the path to processed.csv, which was the file we created using Linux commands. It uses the pandas and matplotlib libraries to visualize this data.

```python
import pandas as pd
import matplotlib.pyplot as plt

file_path = 'C:\\Users\\bdish\\Downloads\\Group10 Nontrivial & Metadata\\Metadata & Nontrivial Info\\data\\processed.csv'

# Load the dataset from the processed CSV file
df = pd.read_csv(file_path, header=0,
                 converters={'profits_percent_change': lambda x: x.rstrip('%'),
                             'market_value': lambda x: x.replace('$', '').replace(',', '').strip()})

# Convert columns to numeric, coerce errors to NaN and drop rows with NaN values
df['profits_percent_change'] = pd.to_numeric(df['profits_percent_change'], errors='coerce')
df['market_value'] = pd.to_numeric(df['market_value'], errors='coerce')

# Create a scatter plot with specified axis limits
plt.figure(figsize=(14, 7))
plt.scatter(df['profits_percent_change'], df['market_value'], color='skyblue')

# Set the maximum value for x-axis at 500 to exclude outliers
plt.xlim(-50, 100)

# Set the maximum value for y-axis at 1 billion to exclude outliers
plt.ylim(0, 100000)

# Set axis labels and plot title
plt.title('Market Value by Profit Percent Change')
plt.xlabel('Profit Percent Change (%)')
plt.ylabel('Market Value ($)')

# Add grid for better readability
plt.grid(True)

# Show the plot
plt.tight_layout()
plt.show()
```

Visualization of Revenue vs. # of Employees for Top 25 Fortune Companies - Plot 2



Visualization of Fortune 1000 Companies based on # of Employees and Revenue ($) - Plot 2



Input files created using linux commands - Plot 2

- **awk -F',"' '{out=""; for(i=2; i<=NF; i++){gsub(/^"|"$/,"",$i); out=out (i>2?",":"") $i} print substr(out,2)}' 'Fortune 1000 Companies by Revenue.csv' > 'new_file.csv'**
  - Initializes awk and sets field separator to `,"` meaning that it will split each line of the input field into fields wherever there is a comma followed by a quote
  - ‘{out=””; for(i=2; i<=NF; i++)` - starts a block of code where ‘out’ is initialized as empty string, then loops from the 2nd to last field
  - `gsub(/^”|”$/,””,$i); - gsub performs global substitution on each field replacing leading and trailing quotes with nothing
  - `out=out (i>2?”,”:””) $i - concatenates current field to `out` variable
  - `print substr(out,2)} - prints out string contained in var `out` starting from second character
- **tr -d '"' < 'new_file.csv' > 'cleaned_file.csv'**
  - Trims remaining quotes from new_file.csv and redirects to cleaned_file.csv
- **cut -d',' -f1,2,12,13,14 cleaned_file.csv > cleaned_file1.csv**
  - Cuts specified fields and redirects output to cleaned_file1.csv
- **head -n 25 cleaned_file1.csv > top_employees_revenue.csv**
  - Extracts the top 25 entries from the cleaned_file1.csv and redirects them to top_employees_revenue.csv

Files created using Linux commands - Plot 2

Top 5 Columns/Rows from new_file.csv

```
revenues ,revenue_percent_change,profits ,profits_percent_change,assets,market_value ,change_in_rank,employees
572,754 ",2.40%,$13,673 ",1.20%,$244,860 ,$409,795 ",-,2,300,000
469,822 ",21.70%,$33,364 ",56.40%,$420,549 ,$1,658,807.30 ",-,1,608,000
365,817 ",33.30%,$94,680 ",64.90%,$351,002 ,$2,849,537.60 ",-,154,000
292,111 ",8.70%,$7,910 ",10.20%,$232,999 ,$132,839.20 ",-,258,000
287,597 ",11.80%,$17,285 ",12.20%,$212,206 ,$479,830.30 ",-,350,000
```

Top 5 Columns/Rows from cleaned_file.csv

```
revenues ,revenue_percent_change,profits ,profits_percent_change,assets,market_value ,change_in_rank,employees
572,754 ,2.40%,$13,673 ,1.20%,$244,860 ,$409,795 ,-,2,300,000
469,822 ,21.70%,$33,364 ,56.40%,$420,549 ,$1,658,807.30 ,-,1,608,000
365,817 ,33.30%,$94,680 ,64.90%,$351,002 ,$2,849,537.60 ,-,154,000
292,111 ,8.70%,$7,910 ,10.20%,$232,999 ,$132,839.20 ,-,258,000
287,597 ,11.80%,$17,285 ,12.20%,$212,206 ,$479,830.30 ,-,350,000
```

```
revenues ,employees
572,754 ,2,300,000
469,822 ,-,1,608
365,817 ,-,154,000
292,111 ,258,000
287,597 ,350,000
```

Top 5 for cleaned_file1.csv

```
revenues ,employees
572754 ,2300000
469822 ,1608000
365817 ,154000
292111 ,258000
287597 ,350000
```

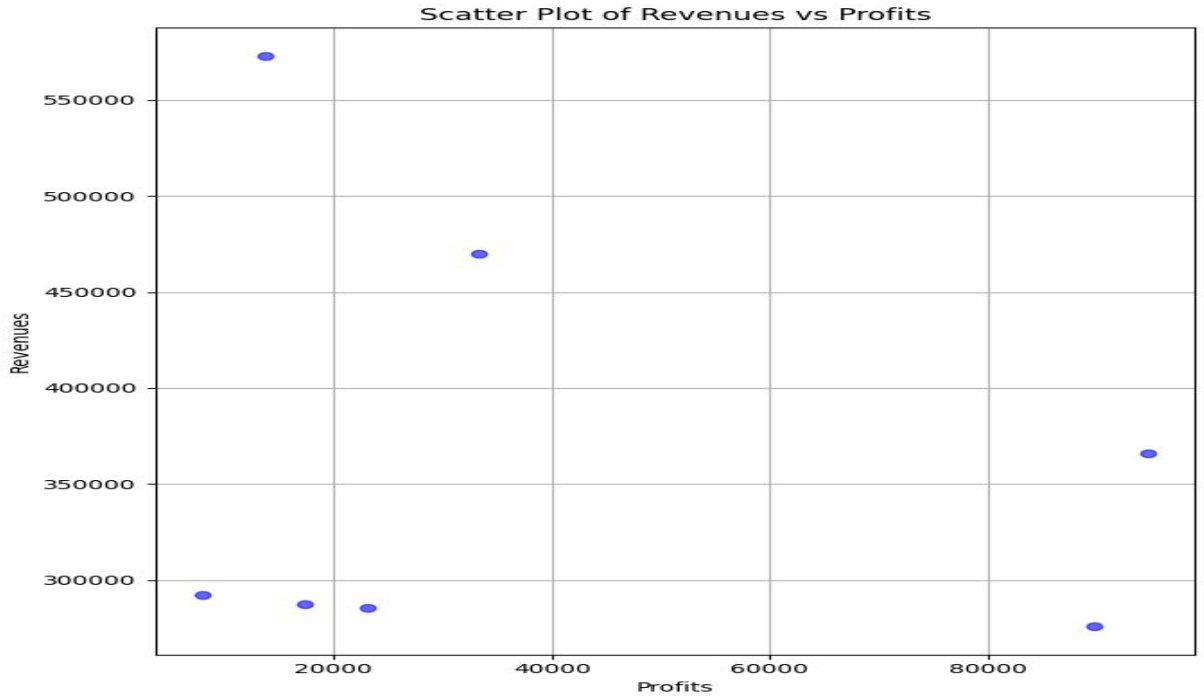
Top 5 for top_employees_revenue.csv

Script for # of Employees and Revenue - Plot 2

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from numpy import polyfit, poly1d

# Load the dataset
df_employees_revenue = pd.read_csv("top_employees_revenue.csv")

df_employees_revenue.columns = df_employees_revenue.columns.str.strip()

# Calculate the coefficients of the linear fit
coefficients = polyfit(df_employees_revenue['revenues'], df_employees_revenue['employees'], 1)

# Create a polynomial object from the coefficients
polynomial = poly1d(coefficients)

# Generate x values for the trend line (from min to max revenues)
x_values = range(int(df_employees_revenue['revenues'].min()), int(df_employees_revenue['revenues'].max()))

# Calculate the y values from the polynomial
y_values = polynomial(x_values)

# Create the scatter plot again
plt.figure(figsize=(10, 6))
plt.scatter(df_employees_revenue['revenues'], df_employees_revenue['employees'], color='green', alpha=0.6)

# Plot the trendline
plt.plot(x_values, y_values, color='red', linestyle='--')

# Set the labels for the axes
plt.xlabel('Revenues')
plt.ylabel('Number of Employees (in millions)')

# Add a title to the plot
plt.title('Scatter Plot of Top 25 Revenues w/ Number of Employees & Trendline')

# Show a grid for better readability
plt.grid(True)

# Display the plot
plt.show()
```

- This script reads in the linux created file “top_employees_revenue.csv”
- Uses pandas, matplotlib, numpy libraries
- Imports polyfit and poly1d to create a linear fit for the data visualization
- This script ultimately visualizes the data for revenue vs the number of employees for each company

Visualization of Revenue and Profit for Top 7 Fortune Companies - Plot 3



Input File Created using Linux commands - Plot 3

- cut -d',' -f3,4,6,7 'Fortune 1000 Companies by Revenue.csv' > revenue_profits.csv
  - This command cuts fields 3,4,6 and 7, the delimiter is a comma, redirects this output to revenue_profits.csv
- head -n 8 revenue_profits.csv > top_revenue_profits.csv
  - Extracts the top 8 columns for revenue and profit column, redirects output to top_revenuePprofits.csv
- sed 's/"//g; s/\$//g' top_revenue_profits.csv > cleaned_top_revenue_profits.csv
  - sed ‘s/”//g - tells sed to search for double quote chars and replace them with nothing, s is substitution command, g is global
  - s/\$//g - tells sed to search for $ and replace with nothing, / before $ is an escape character

Script for Revenue vs Profits - Plot 3

```
import pandas as pd
import matplotlib.pyplot as plt

df_new = pd.read_csv('cleaned_top_revenue_profits.csv')

# Strip any whitespace from the column names
df_new.columns = df_new.columns.str.strip()

# Now we will plot the data again
plt.figure(figsize=(10, 6))
plt.scatter(df_new['profits'], df_new['revenues'], color='blue', alpha=0.6)

# Set the labels for the axes based on the file content
plt.xlabel('Profits')
plt.ylabel('Revenues')

# Add a title to the plot
plt.title('Scatter Plot of Revenues vs Profits')

# Show a grid for better readability
plt.grid(True)

# Display the plot
plt.show()
```

- Reads in linux-created file ‘cleaned_top_revenue_profits.csv’
- Uses pandas and matplotlib libraries to create graph
- This script creates a scatter plot that visualizes the profits vs revenue for the top Fortune companies