Introduction to the income statement

ANALYZING FINANCIAL STATEMENTS IN PYTHON

Rohan Chatterjee
Risk Modeler

datacamp

The income statement

- Shows company's **revenue** and **expenses** over a period of time
- Expenses:
  - Operating expenses
  - Non-operating expenses

Income statement of ABC In Thousands of US Dollars	
Total Revenue	13400
Operating Expenses	
Cost of goods sold	(1253)
Marketing expenses	(4520)
Depreciation	(230)
Total Operating Expenses	(6003)
Interest expenses	(2123)
Income before taxes	5274
Income taxes (15%)	(791.1)
Net income	4482.9

Gross margin

- Ratio of total revenue after taking out cost of goods sold to total revenue
- Proportion of revenue left after subtracting cost incurred to earn the revenue

Formula:

$$\frac{\text{Total Revenue - Cost of Goods Sold}}{\text{Total Revenue}}$$

Operating margin

- Ratio of total revenue after taking out operating expenses to total revenue
- Proportion of revenue left after operating expenses

Formula:

$$\frac{\text{Total Revenue - Operating Expenses}}{\text{Total Revenue}}$$

Family of ratios

Efficiency ratio

- Measures company's ability to generate income via its resources
- For example:
  - Gross margin
  - Operating margin

Let's practice!

ANALYZING FINANCIAL STATEMENTS IN PYTHON

Ratios from the income statement and balance sheet

ANALYZING FINANCIAL STATEMENTS IN PYTHON

Rohan Chatterjee
Risk modeler

datacamp

Asset turnover ratio

- Ratio of revenue to assets
- Measures how efficiently a company is using its assets to generate revenue

Formula:

$$\frac{\text{Total Revenue}}{\text{Total Assets}}$$

Computing asset turnover ratio using pandas

```
merged_dat = pd.merge(income_statement, balance_sheet, on = ["Year", "company"])
```

- `merged_dat` can now be used to compute the ratio and add it as another column in the DataFrame

```
merged_dat["asset_turnover"] = merged_dat["Total Revenue"] / merged_dat["Total Assets"]
```

User-defined functions to compute ratios

- Creating the columns below can become repetitive

```
balance_sheet["current_ratio"] = balance_sheet["Total Current Assets"] / balance_sheet["Total Current Liabilities"]

balance_sheet["debt_to_equity"] = balance_sheet["Total Liab"] / balance_sheet["Total Stockholder Equity"]
```

- To avoid having to type this code again and again, we can package this into a user-defined function

User-defined functions to compute ratios

```
def compute_ratio(df, numerator, denominator, ratio_name):
      df[ratio_name] = df[numerator]/df[denominator]
      return df
```

Compute the current ratio and debt-to equity ratio from the DataFrame `balance_sheet` using `compute_ratio`:

```
balance_sheet = compute_ratio(balance_sheet, "Total Current Assets",
                              "Total Current Liabilities", "current_ratio")
balance_sheet = compute_ratio(balance_sheet, "Total Liab",
                              "Total Stockholder Equity", "debt_to_equity")
```

User-defined functions to compute ratios

- Define the numerators, denominators and the ratio names in separate lists:

```
list_of_numerators = ["Total Current Assets", "Total Liab"]
list_of_denominators = ["Total Current Liabilities",
                        "Total Stockholder Equity"]
list_of_ratio_names = ["current_ratio", "debt_to_equity"]
```

- Loop over the lists and call the function `compute_ratio` :

```
for numerator, denominator, ratio_name in zip(list_of_numerators,
                                              list_of_denominators,
                                              list_of_ratio_names):


    balance_sheet = compute_ratio(balance_sheet,
            numerator,
            denominator,
            ratio_name)
```

Let's practice!

ANALYZING FINANCIAL STATEMENTS IN PYTHON

Visualizing ratios for within-company analysis

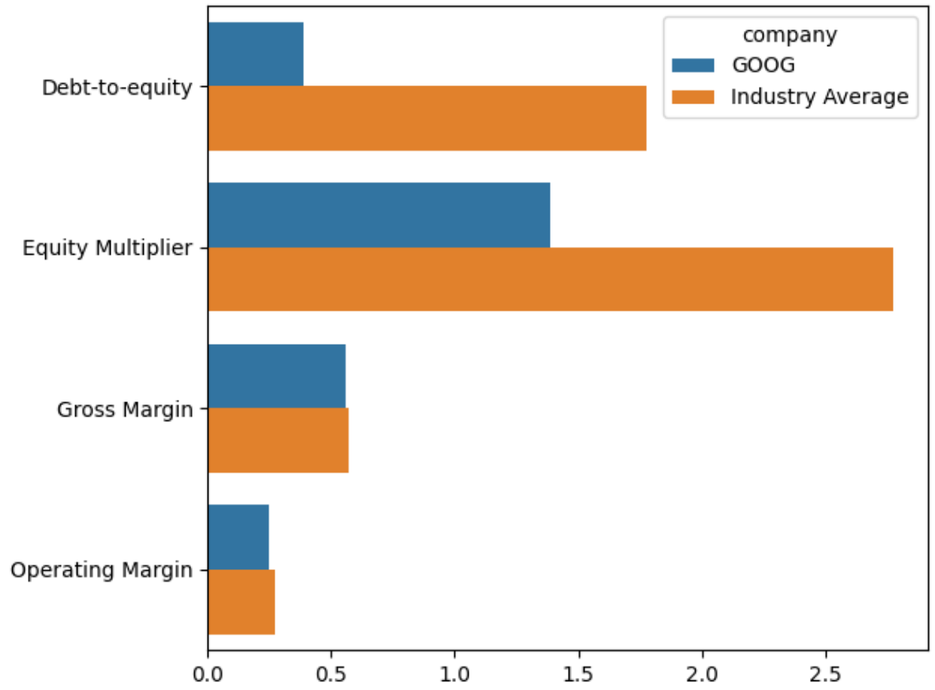
ANALYZING FINANCIAL STATEMENTS IN PYTHON

Rohan Chatterjee
Risk modeler

datacamp

Visualizing financial ratios

- **Bar plots** are helpful for
  - visualizing financial ratios for a company on average, and
  - assessing performance relative to the industry average



Preparing data for plotting

- Using `pivot_table` to compute the average of the ratios by company:

```
avg_company_ratio = plot_dat.pivot_table(index=["comp_type",
                                                "company"],
                         values=["Gross Margin", "Operating Margin",
                                 "Debt-to-equity", "Equity Multiplier"],
                                aggfunc="mean").reset_index()
```

- `print(avg_company_ratio.head())`

comp_type	company	Debt-to-equity	Equity Multiplier	Gross Margin	Operating Margin
tech	AAPL	4.306	5.306	0.403	0.272
tech	AMZN	2.461	3.461	0.407	0.054
tech	GOOG	0.386	1.386	0.556	0.248
tech	META	0.262	1.262	0.816	0.408

Preparing data for plotting

- Use `pivot_table` to compute the average ratio by industry:

```
avg_industry_ratio = plot_dat.pivot_table(index="comp_type",
                        values=["Gross Margin", "Operating Margin",
                                "Debt-to-equity",
                                "Equity Multiplier"],
                                    aggfunc="mean").reset_index()
```

- `print(avg_industry_ratio.head())`

comp_type	Debt-to-equity	Equity Multiplier	Gross Margin	Operating Margin
fmcg	2.998	4.050	0.514	0.207
real_est	5.692	7.353	0.535	0.300
tech	1.777	2.777	0.572	0.274

Preparing data for plotting

- Plotting data in `seaborn` requires the data to be in a "long" format. Use `pd.melt` to melt the DataFrames `avg_industry_ratio` and `avg_company_ratio` into long format:

```
molten_plot_company = pd.melt(avg_company_ratio, id_vars=["comp_type",
                                                          "company"])
molten_plot_industry = pd.melt(avg_industry_ratio,
                               id_vars=["comp_type"])
```

- `print(molten_plot_company.head())`

comp_type	company	variable	value
tech	AAPL	Debt-to-equity	4.306
tech	AMZN	Debt-to-equity	2.461
tech	GOOG	Debt-to-equity	0.386
tech	META	Debt-to-equity	0.262
tech	MSFT	Debt-to-equity	1.472

- `print(molten_plot_industry.head())`

comp_type	variable	value
fmcg	Debt-to-equity	2.998
real_est	Debt-to-equity	5.692
tech	Debt-to-equity	1.777
fmcg	Equity Multiplier	4.050

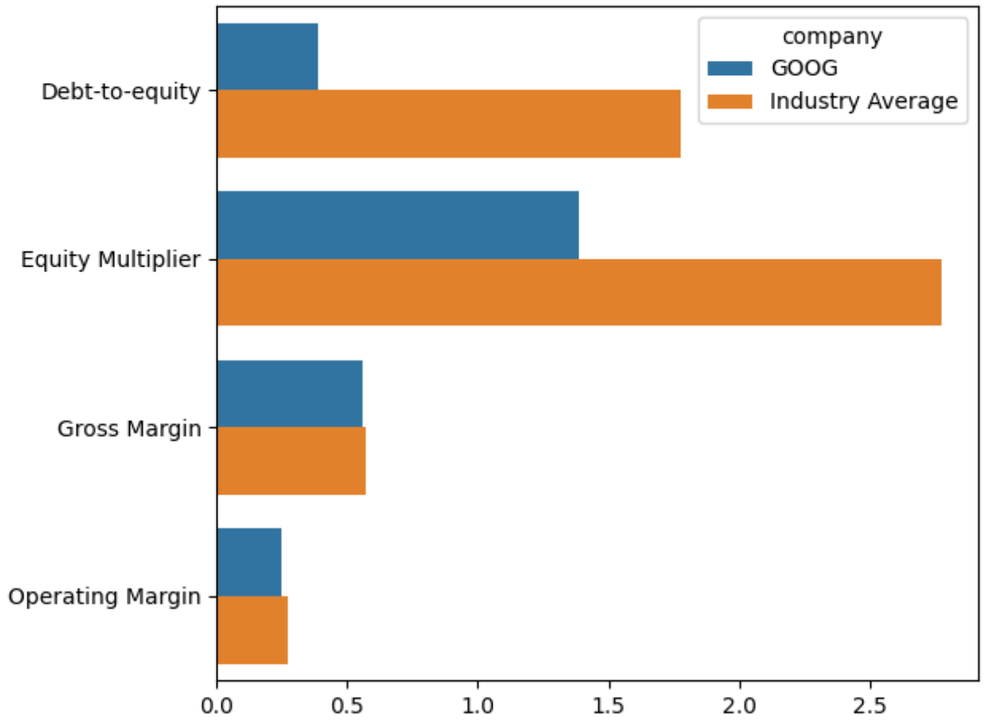
Preparing data for plotting

- Seaborn requires all the data we want to plot in one DataFrame
- We use `pd.concat` to concatenate `molten_plot_company` and `molten_plot_industry`
- `molten_plot_industry` does not have the `company` DataFrame because it contains the average of the ratios per industry overall
- `pd.concat` requires both DataFrames should to have the same columns, so we add the column `company` to `molten_plot_industry`

```
molten_plot_industry["company"] = "Industry Average"
molten_plot = pd.concat([molten_plot_company, molten_plot_industry])
```

Make the bar graph

```
sns.barplot(data=molten_plot, y="variable", x="value", hue="company", ci=None)
plt.xlabel(""), plt.ylabel("")
plt.show()
```



Let's practice!

ANALYZING FINANCIAL STATEMENTS IN PYTHON