Compare Companies

Data Science-Project based on comparing two companies financials on their Income statement/Balance sheet

Table of Contents

1. Goals of the Project
2. Getting Data for the Analysis
3. Transforming the Data from the API
4. Jupyter Python Code Example
5. Building a Data Model in Power BI
6. Data Model
7. Implementation
8. Implementation
9. Dynamic Graph Axis
10. Business Logic
11. Business Logic

Goals of the Project

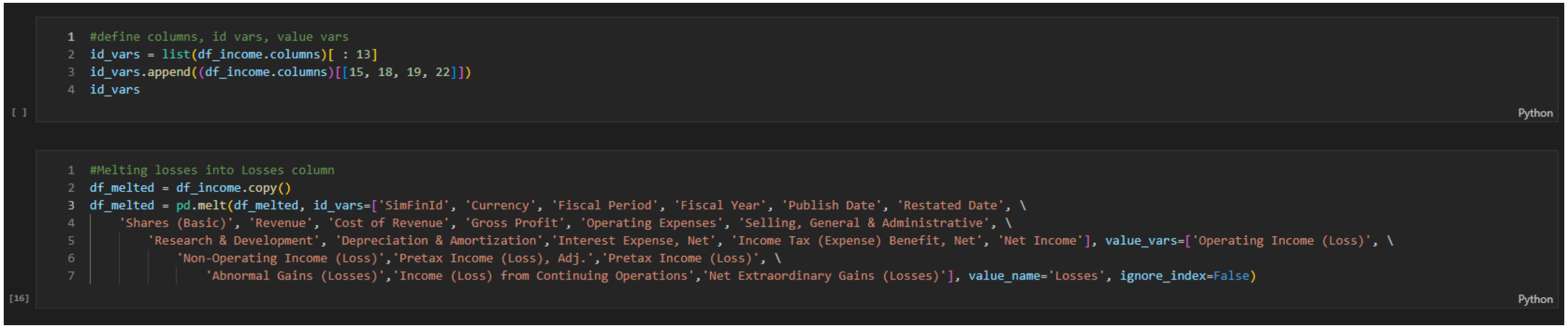
- The ability for the user to compare two companies financials
- Interact, compare data dynamically using a dashboard created in Power BI
- Having relevant data of all S&P 500 Companies of the last 5 Years
- Calculate financial differences for two Date Ranges over a 5 Year fiscal period

Getting Data for the Analysis

- For the data I used the API of the website SimFin
- It offers great detailed data. The API is well documented - free version offers 5 years of relevant financial data of all Companies based in the US, Germany, Canada and China
- The data that was used is from the income statement and balance statement, together with company data for the US market
- The data focuses on the S&P 500 Companies based on the following Wikipedia List
- Data model can be expanded to include all markets

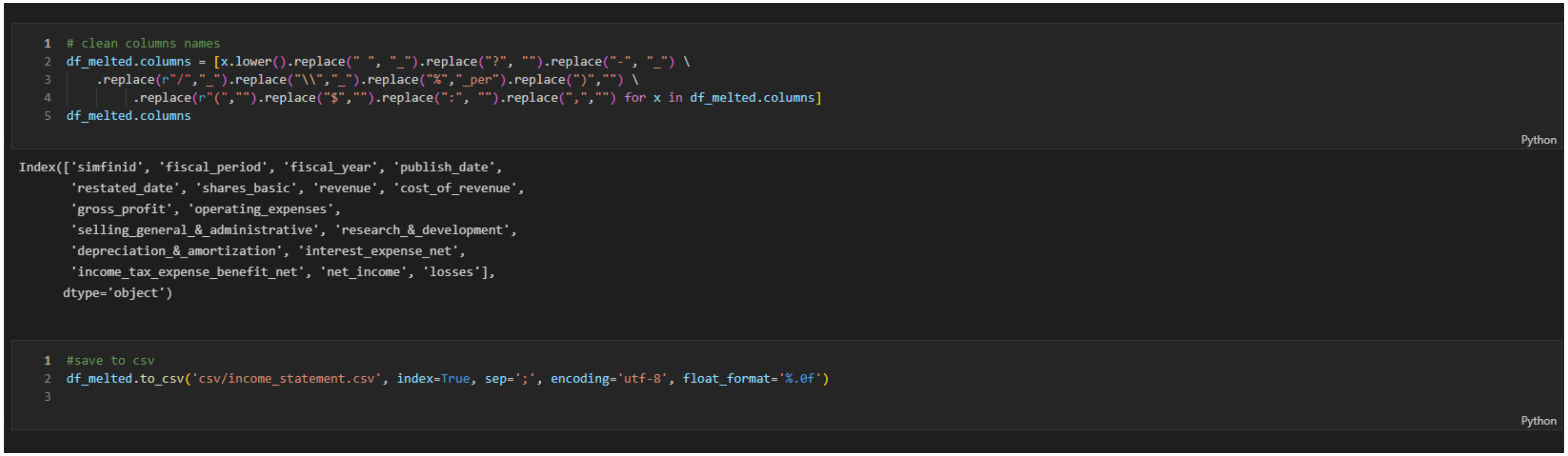
Transforming the Data from the API

- To transform the data I used Python with the Jupyter Notebooks extension
- Libraries used are: The official Simfin API and Pandas Data frame to transform the data
- The scripts are available on my Github
- Used Pandas Data frame to index, melt columns with Losses into one column
- Cleaned column names (lower case, empty spaces, special signs) in case we used the Data frame for a Database
- Saving the Data frame into .csv format so we can use it in Power BI



```python
#define columns, id vars, value vars
id_vars = list(df_income.columns)[ : 13]
id_vars.append((df_income.columns)[[15, 18, 19, 22]])
id_vars
```

```python
#Melting losses into Losses column
df_melted = df_income.copy()
df_melted = pd.melt(df_melted, id_vars=['SimFinId', 'Currency', 'Fiscal Period', 'Fiscal Year', 'Publish Date', 'Restated Date', \
    'Shares (Basic)', 'Revenue', 'Cost of Revenue', 'Gross Profit', 'Operating Expenses', 'Selling, General & Administrative', \
        'Research & Development', 'Depreciation & Amortization','Interest Expense, Net', 'Income Tax (Expense) Benefit, Net', 'Net Income'], value_vars=['Operating Income (Loss)', \
            'Non-Operating Income (Loss)','Pretax Income (Loss), Adj.','Pretax Income (Loss)', \
                'Abnormal Gains (Losses)','Income (Loss) from Continuing Operations','Net Extraordinary Gains (Losses)'], value_name='Losses', ignore_index=False)
```



```python
# clean columns names
df_melted.columns = [x.lower().replace(" ", "_").replace("?", "").replace("-", "_") \
    .replace(r"/","_").replace("\\","_").replace("%","_per").replace(")","") \
        .replace(r"(","").replace("$","").replace(":", "").replace(",","") for x in df_melted.columns]
df_melted.columns
```

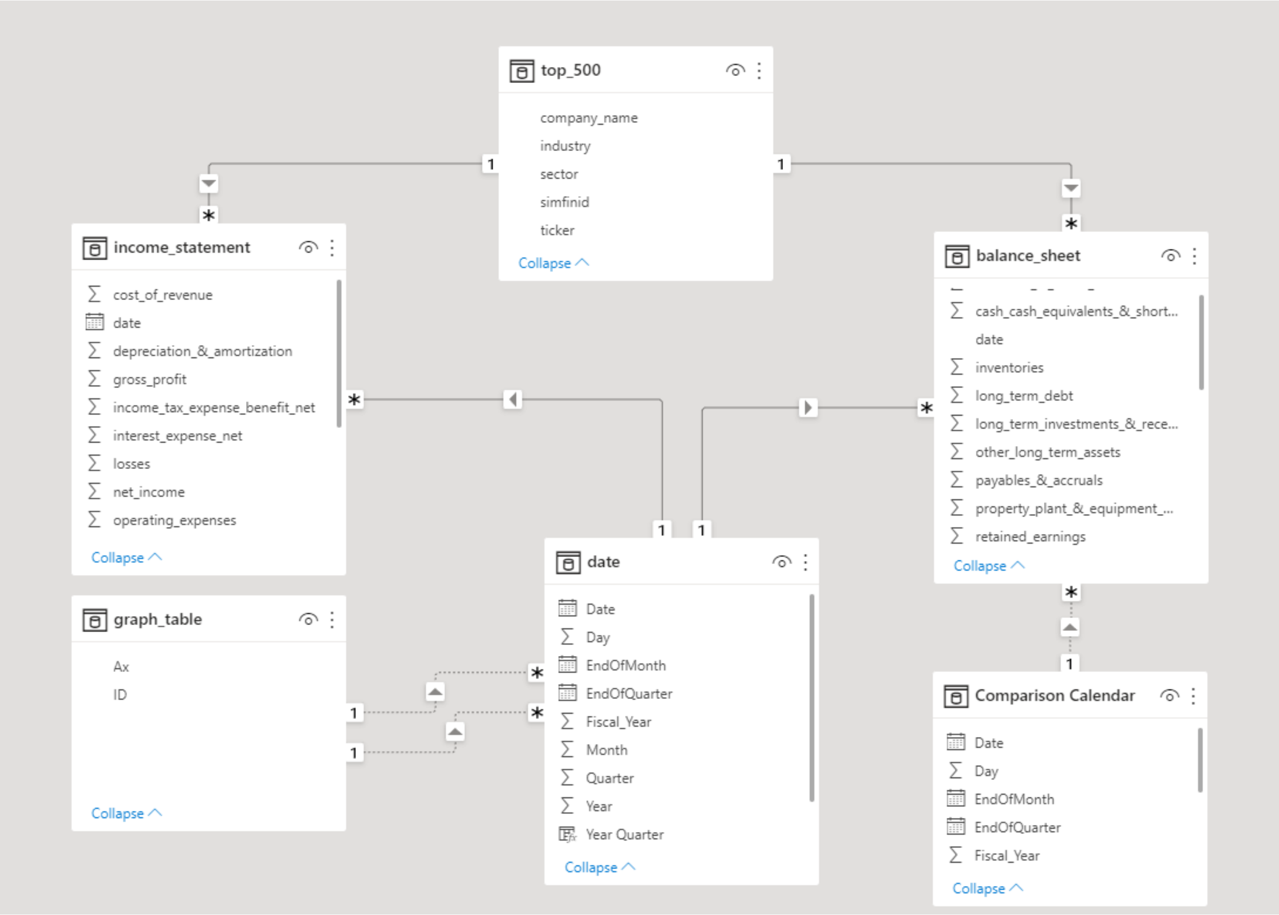
```
Index(['simfinid', 'fiscal_period', 'fiscal_year', 'publish_date',
       'restated_date', 'shares_basic', 'revenue', 'cost_of_revenue',
       'gross_profit', 'operating_expenses',
       'selling_general_&_administrative', 'research_&_development',
       'depreciation_&_amortization', 'interest_expense_net',
       'income_tax_expense_benefit_net', 'net_income', 'losses'],
      dtype='object')
```

```python
#save to csv
df_melted.to_csv('csv/income_statement.csv', index=True, sep=';', encoding='utf-8', float_format='%.0f')

```

Building a Data Model in Power BI

- Two Fact tables (Income statement and Balance sheet)
- The Fact tables are connected with two Dimension tables (Date and Top 500 company)
- The Comparison Calendar table so we can compare two time periods of a selected company
- Two not connected tables are for Company A and Company B, so we could compare with sliders two different company financials
- All measures are separate in the Table Calculations
- The graph table is for the chart slider time period selection

Data Model



top_500

company_name
industry
sector
simfinid
ticker

income_statement

cost_of_revenue
date
depreciation_&_amortization
gross_profit
income_tax_expense_benefit_net
interest_expense_net
losses
net_income
operating_expenses

balance_sheet

cash_cash_equivalents_&_short...
date
inventories
long_term_debt
long_term_investments_&_rece...
other_long_term_assets
payables_&_accruals
property_plant_&_equipment_...
retained_earnings

graph_table

Ax
ID

date

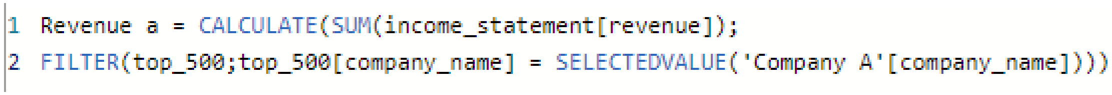
Date
Day
EndOfMonth
EndOfQuarter
Fiscal_Year
Month
Quarter
Year
Year Quarter

Comparison Calendar

Date
Day
EndOfMonth
EndOfQuarter
Fiscal_Year

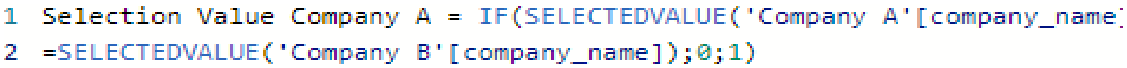
Implementation

- For comparing two different companies we need two different selection values
- The solution is creating two separate Company tables and fill them with values of the main top-500 company name table
- We filter measures with a Filter expression, and the help of SELECTEDVALUE – pointing to the selected value in the slicer



```
Revenue a = CALCULATE(SUM(income_statement[revenue]);
FILTER(top_500;top_500[company_name] = SELECTEDVALUE('Company A'[company_name])))
```

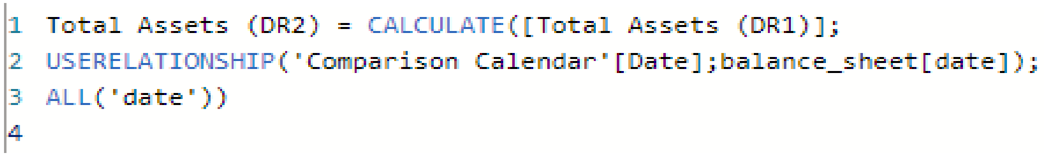
- To prevent the user selecting the same company twice, we use a simple IF statement that does not allow selecting a company if it was already selected



```
Selection Value Company A = IF(SELECTEDVALUE('Company A'[company_name
=SELECTEDVALUE('Company B'[company_name]);0;1)
```

Implementation

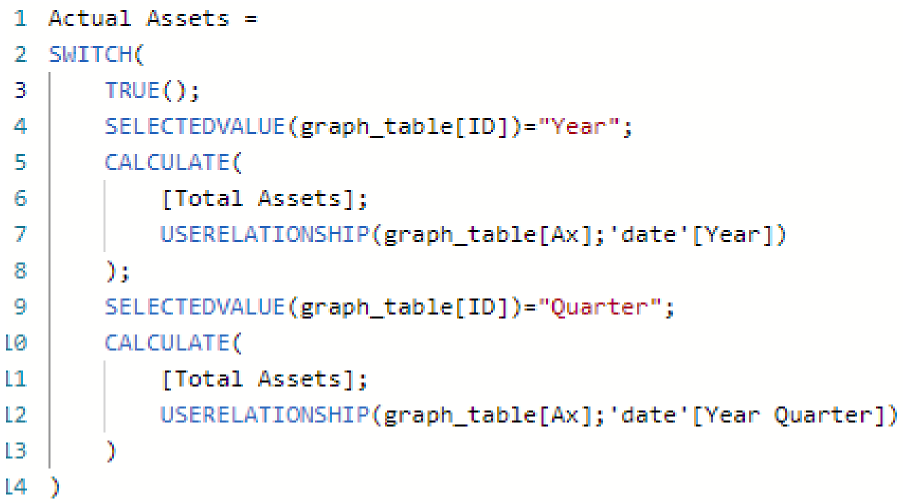
- To calculate the Difference for two Date Ranges we need two separate Date slicers
- Since we already have one date table we only need to create one for comparison. We simply copy the existing Date table
- We connect the comparison calendar to the Fact table (Balance sheet)
- The relationship between the tables can be set to inactive, since we will make our calculations with the help of USERELATIONSHIP



```
Total Assets (DR2) = CALCULATE([Total Assets (DR1)];
USERELATIONSHIP('Comparison Calendar'[Date];balance_sheet[date]);
ALL('date'))

```

Dynamic Graph Axis

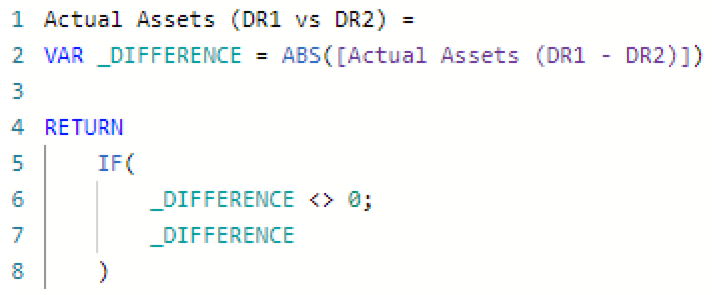
- To enable the user the ability to chose if the Column chart should display financial data in Quarters or Years
- We implement it with the help of the SWITCH function, and USERELATIONSHIP – we use the relationship only when we want to calculate or use a different measure



```
Actual Assets =
SWITCH(
    TRUE();
    SELECTEDVALUE(graph_table[ID])="Year";
    CALCULATE(
        [Total Assets];
        USERELATIONSHIP(graph_table[Ax];'date'[Year])
    );
    SELECTEDVALUE(graph_table[ID])="Quarter";
    CALCULATE(
        [Total Assets];
        USERELATIONSHIP(graph_table[Ax];'date'[Year Quarter])
    )
)
```

Business Logic

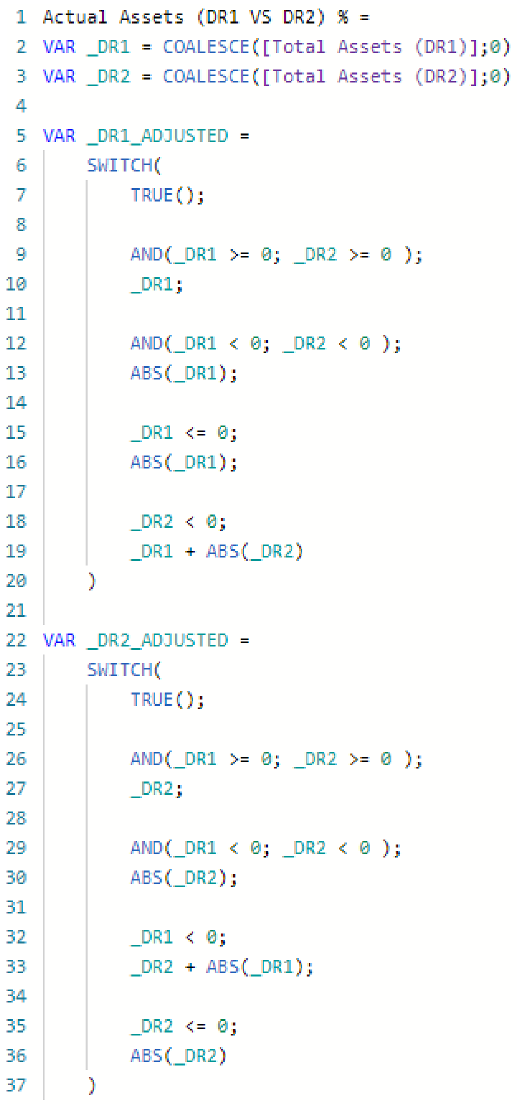
- The Business Logic applied in this Data model is very simple with basic operations such as SUM, DIVIDE etc.
- The most difficult part is ensuring that the logic can handle zeros, blanks and negative values when calculating differences between fiscal periods
- A simple logic to implement for the measure to calculate the difference between Assets return a blank rather than 0. The difference is always positive so we use the function ABS



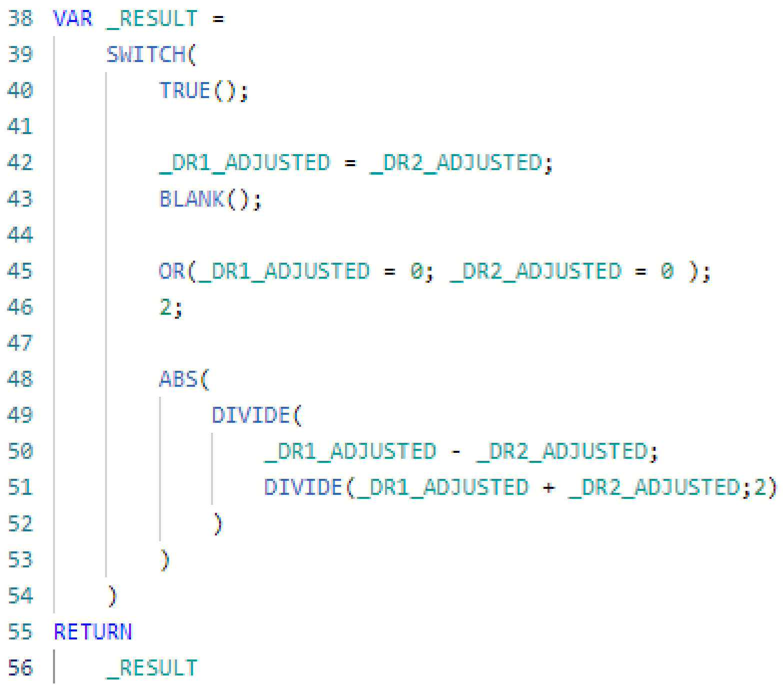
```
Actual Assets (DR1 vs DR2) =
VAR _DIFFERENCE = ABS([Actual Assets (DR1 - DR2)])

RETURN
    IF(
        _DIFFERENCE <> 0;
        _DIFFERENCE
    )
```

Business Logic

- For calculating the % of the actual revenue we used the following functions
- COALESCE() to convert blank values to 0
- We adjust both values to they can only be of value 0 or positive
- The SWITCH(TRUE ()) uses the first branch of the switch sentence, the last branch is the default value
- The last switch statement checks the condition if both values _DR1 and _DR2 (adjusted) are 0. If this condition is not checked we would get an incorrect result
- We format the measure as Percentage



```
Actual Assets (DR1 VS DR2) % =
VAR _DR1 = COALESCE([Total Assets (DR1)];0)
VAR _DR2 = COALESCE([Total Assets (DR2)];0)

VAR _DR1_ADJUSTED =
    SWITCH(
        TRUE();

        AND(_DR1 >= 0; _DR2 >= 0 );
        _DR1;

        AND(_DR1 < 0; _DR2 < 0 );
        ABS(_DR1);

        _DR1 <= 0;
        ABS(_DR1);

        _DR2 < 0;
        _DR1 + ABS(_DR2)
    )

VAR _DR2_ADJUSTED =
    SWITCH(
        TRUE();

        AND(_DR1 >= 0; _DR2 >= 0 );
        _DR2;

        AND(_DR1 < 0; _DR2 < 0 );
        ABS(_DR2);

        _DR1 < 0;
        _DR2 + ABS(_DR1);

        _DR2 <= 0;
        ABS(_DR2)
    )
```



```
VAR _RESULT =
    SWITCH(
        TRUE();

        _DR1_ADJUSTED = _DR2_ADJUSTED;
        BLANK();

        OR(_DR1_ADJUSTED = 0; _DR2_ADJUSTED = 0 );
        2;

        ABS(
            DIVIDE(
                _DR1_ADJUSTED - _DR2_ADJUSTED;
                DIVIDE(_DR1_ADJUSTED + _DR2_ADJUSTED;2)
            )
        )
    )
RETURN
    _RESULT
```