Comparables Analysis Module

Instructions

Confidential, for internal use only. Do not distribute.

Description: This module provides the user the ability to *compare* public companies to a target company.

Note for the developer: Most functionality will be provided by an API that handles the data querying and processing. For completeness we provide the exact API endpoints that we believe are required for each task. All API endpoints require a `api_token` parameter which for testing puroses is set to `t3stt@ken` and is ignored in the examples for simplicity. The full documentation of the API as well as an interface for testing is available at https://pyedgarai-jfimbett.replit.app/openapi

Identification of firms to compare.

Task 1: Search bar for firms

The first requirement of the module is a search bar for firms, using different type of identifiers (CIK, ticker, name, etc...) via which the user selects a company for which the comparables will be retrieved. This implies a connection to the API endpoint of the backend. There are two API endpoints necessary for this task: `/cik_tickers`, and `/cik_names`. Apart from the `api_token` these endpoints do not require any additional parameter.

Example:

```
curl -X 'GET' \
  'https://pyedgarai-jfimbett.replit.app/cik_tickers?api_token=t3stt%40ken' \
  -H 'accept: application/json'
```

Response

```
{
"0000012040": [
    "BDL"
  ],
  "0000012208": [
    "BIO",
    "BIO-B"
  ],
  "0000012239": [
    "DOMH"
  ],
  "0000012659": [
    "HRB"
  ],
  "0000012927": [
    "BA"
  ],
  "0000013156": [
    "GLXZ"
  ],
  "0000013372": [
    "NSARP",
    "NSARO"
  ],
  "0000014177": [
    "BRID"
  ],
  "0000014272": [
    "BMYMP",
    "BMY",
    "CELG-RI"
  ],
  "0000014693": [
    "BF-A",
    "BF-B"
  ],
  "0000014707": [
    "CAL"
  ],
  "0000014846": [
    "BRT"
  ],
  "0000014930": [
    "BC-PB",
    "BC-PC",
    "BC",
    "BC-PA"
  ],
}
```

where the keys are the CIKS (treated as strings with leading zeros) and the values are lists of tickers. Note how one CIK can have multiple tickers, this happens when a company has different classes of shares.

Equivalently for the `/cik_names` endpoint the response is a dictionary with the keys being the CIKs (without the leading zeros) and the values being the names of the companies.

```
curl -X 'GET' \
  'https://pyedgarai-jfimbett.replit.app/cik_names?api_token=t3stt%40ken' \
  -H 'accept: application/json'
```

Response

```
{
  "1750": "AAR CORP",
  "1800": "ABBOTT LABORATORIES",
  "1961": "WORLDS INC.",
  "2098": "ACME UNITED CORP",
  "2178": "ADAMS RESOURCES & ENERGY, INC.",
  "2186": "BK Technologies Corporation",
  ...
}
```

The module can perform a request to both endpoints and with the information queried to provide the user with a search bar that can search by CIK, ticker, or company name. The search bar should be able to autocomplete the search query. We require a single input for the search bar that accepts the three types of identifiers, and the user should be able to select the company from the search bar. In the options, provide the complete name of the company, the ticker, and the CIK.

Task 2: Add a new company for comparable search.

If the user wants to add his own company (that is not part of the database) a new section should open where the user can either import data from a document (this functionality will be provided by another module and it is not required at the moment) or input data for the new firm. The user should be able to input the basic information of the company (name), and select the industry and specific accounting variables to fill. The variables that the user can choose to fill are provided in endpoint `/all_accounts`. An example of the response of this endpoint is provided below:

```
curl -X 'GET' \
  'https://pyedgarai-jfimbett.replit.app/all_accounts?api_token=t3stt%40ken' \
  -H 'accept: application/json'
```

```
{
  "AccountsPayableCurrent": {
    "description": "Carrying value as of the balance sheet date of liabilities
incurred (and for which invoices have typically been received) and payable to
vendors for goods and services received that are used in an entity's business.
Used to reflect the current portion of the liabilities (due within one year or
within the normal operating cycle if longer).",
    "instant": 1,
    "name": "Accounts Payable, Current",
    "taxonomy": "us-gaap",
    "units": "USD"
  },
  "AccountsReceivableAfterAllowanceForCreditLossCurrent": {
    "description": "Amount, after allowance for credit loss, of right to
consideration from customer for product sold and service rendered in normal course
of business, classified as current.",
    "instant": 1,
    "name": "Accounts Receivable, after Allowance for Credit Loss, Current",
    "taxonomy": "us-gaap",
    "units": "USD"
  },
  "AccountsReceivableAllowanceForCreditLossCurrent": {
    "description": "Amount of allowance for credit loss on accounts receivable,
classified as current.",
    "instant": 1,
    "name": "Accounts Receivable, Allowance for Credit Loss, Current",
    "taxonomy": "us-gaap",
    "units": "USD"
  },
  ...
}
```

The user should be able to select the variables that he wants to fill, and search for them by typing in the search bar. The search bar should be able to autocomplete the search query or to provide a list of possible variables that the user can select.

The companies added by the user should be stored inside of the website and should be available for the user to select in the future. However, these companies should not be stored in the database of the API, since we do not have a way to verify the data input by the user.

Identification of comparables model:

A dropdown menu where the user selects a method for choosing the comparables (also chooses how many comparables to show). For the moment the developer can have a dropdown menu with the following ***methods*** available:

- Automatic
- LLM

However for the moment the API only has implemented one method, the Automatic method. The output of all methods have the same format, so for the moment there is no need to send the method to the API.

Comparable selection of variables

When computing comparable firms, the user should be able to select a subset of variables from the following list:

- industry
- size
- profitability
- growth_rate
- capital_structure
- location

Finally there are some optional parameters that the user can select:

- industry_digits: The number of digits to consider when comparing the industry. For example, if the user selects 2, the industry will be compared at the 2-digit SIC level.
- size_interval: Interval in % to consider similar companies, e.g. 100.
- profitability_interval: Interval in % to consider similar companies, e.g. 100.
- growth_rate_interval: Interval in % to consider similar companies, e.g. 100.
- capital_structure_interval: Interval in % to consider similar companies, e.g. 100.
- location: Parameter not yet implemented in the API, but the user should be able to select one of the following:
    - state_level
    - country_level
    - region_level

Identification of variables to be shown in the output:

When showing information about the comparables, the user should be able to select the variables that he wants to see. These variables should be selected from the same list of variables that the user can select when adding a new company.

Example of the API endpoint for the list of comparables:

```
curl -X 'GET' \
  'https://pyedgarai-jfimbett.replit.app/comparables?
cik=320193&variables_to_compare=industry&variables_to_compare=size&variables_to_compa
\
  -H 'accept: application/json'
```

Response:

```
{
  "EarningsPerShareBasic": {
    "0": 1.53,
    "1": 0.47,
    "2": 0.86,
    "3": -0.11
  },
  "GrossProfit": {
    "0": 42271000000,
    "1": 8273000000,
    "2": 1471200000,
    "3": 495706000
  },
  "NetIncomeLoss": {
    "0": 23636000000,
    "1": 1886000000,
    "2": 278800000,
    "3": -35009000
  },
  "accn": {
    "0": "0000320193-24-000069",
    "1": "0000858877-24-000007",
    "2": "0001327567-24-000017",
    "3": "0001628280-24-027955"
  },
  "assets": {
    "0": 337411000000,
    "1": 122998000000,
    "2": 17930800000,
    "3": 3623610000
  },
  "assets_5": {
    "0": 341998000000,
    "1": 97287000000,
    "2": 6261800000,
    "3": 2069189000
  },
  "cik": {
    "0": 320193,
    "1": 858877,
    "2": 1327567,
    "3": 1474432
  },
  "debt_to_equity": {
    "0": 3.5476857967,
    "1": 1.6874235274,
    "2": 3.0133398988,
    "3": 1.6373225675
  },
  "end": {
    "0": "2024-03-30",
    "1": "2024-04-27",
    "2": "2024-04-30",
    "3": "2024-05-05"
  },
  "entityName": {
    "0": "Apple Inc.",
    "1": "CISCO SYSTEMS, INC.",
    "2": "PALO ALTO NETWORKS, INC",
    "3": "Pure Storage, Inc."
  },
  "equity": {
    "0": 74194000000,
    "1": 45768000000,
    "2": 4467800000,
    "3": 1373973000
  },
  "growth_rate": {
    "0": -0.0134123591,
    "1": 0.2642799141,
    "2": 1.8635216711,
    "3": 0.7512223388
  },
    "liabilities": {
      "0": 263217000000,
      "1": 77230000000,
      "2": 13463000000,
      "3": 2249637000
    },
    "loc": {
      "0": "US-CA",
      "1": "US-CA",
      "2": "US-CA",
      "3": "US-CA"
    },
    "name": {
      "0": "Apple Inc.",
      "1": "CISCO SYSTEMS, INC.",
      "2": "PALO ALTO NETWORKS, INC",
      "3": "Pure Storage, Inc."
    },
    "profit": {
      "0": 23636000000,
      "1": 1886000000,
      "2": 278800000,
      "3": -35009000
    },
    "profitability": {
      "0": 0.0700510653,
      "1": 0.0153335827,
      "2": 0.0155486649,
      "3": -0.0096613598
    },
    "sic": {
      "0": 35,
      "1": 35,
      "2": 35,
      "3": 35
    },
    "start": {
      "0": "2023-12-31",
      "1": "2024-01-28",
      "2": "2024-02-01",
      "3": "2024-02-05"
    },
    "state": {
      "0": "CA",
      "1": "CA",
      "2": "CA",
      "3": "CA"
    },
    "tickers": {
      "0": [
        "AAPL"
      ],
      "1": [
        "CSCO"
      ],
      "2": [
        "PANW"
      ],
      "3": [
      "PSTG"
    ]
  },
  "val": {
    "0": 337411000000,
    "1": 122998000000,
    "2": 17930800000,
    "3": 3623610000
  }
}
```

Multiples

Once the list of comparables is shown, the user should also be able to see the following three multiples for the comparable companies:

- Price to Earnings (P/E)
- Price to Book (P/B)
- Enterprise to EBITDA (EV/EBITDA)

these ratios can be obtained in the endpoint `/valuation_metrics`. The endpoint requires a list of tickers and the `api_token` as parameters. The api response consists of the valuation metrics for the list of tickers and the average of each one of the multiples.

```
curl -X 'GET' \
  'https://pyedgarai-jfimbett.replit.app/valuation_metrics?
tickers=AAPL&tickers=MSFT&tickers=GOOGL&api_token=t3stt%40ken' \
  -H 'accept: application/json'
```

output

```
{
  "avg_multiples": {
    "enterpriseToEbitda": 22.572666666666667,
    "priceToBook": 23.3740289,
    "priceToEarnings": 33.50176351750078
  },
  "variables": [
    {
      "currentPrice": 226.8,
      "enterpriseToEbitda": 26.467,
      "enterpriseValue": 3487801278464,
      "eps": 6.16,
      "marketCap": 3448289886208,
      "priceToBook": 51.75719,
      "priceToEarnings": 36.81818181818182,
      "sharesOutstanding": 15204100096,
      "ticker": "AAPL"
    },
    {
      "currentPrice": 416.06,
      "enterpriseToEbitda": 24.066,
      "enterpriseValue": 3114910875648,
      "eps": 11.86,
      "marketCap": 3092590624768,
      "priceToBook": 11.52042,
      "priceToEarnings": 35.080944350758855,
      "sharesOutstanding": 7433039872,
      "ticker": "MSFT"
    },
    {
      "currentPrice": 167.06,
      "enterpriseToEbitda": 17.185,
      "enterpriseValue": 1984502562816,
      "eps": 5.84,
      "marketCap": 2064878272512,
      "priceToBook": 6.8444767,
      "priceToEarnings": 28.606164383561644,
      "sharesOutstanding": 5858999808,
      "ticker": "GOOGL"
    }
  ]
}
```

apply the multiples to the target company

Finally, the user should be able to apply the multiples to the target company. This will be performed in the website and not in the API. The idea is the following, given an average multiple e.g. the average Price to Book of the comparables, an estimate value of the target company can be obtained by multiplying the average multiple by the accounting variable of the target company.

$$\text{Estimated Value} = \text{Accounting Variable} \times \text{Average Multiple}$$

for example

$$\text{Estimated Market Value} = \text{Book Value of the target company} \times \text{Average Price to Book of the comparables}$$

in this step the user should input the accounting variable of the target company and the average multiple.