Analytics mindset

EDGAR Explorer

Part 1: Inline XBRL

Background

Digital reporting of financial and nonfinancial business information continues to evolve and increase in importance. XBRL, eXtensible Business Reporting Language, is an XML standard for tagging business and financial reports using digital dictionaries, or taxonomies. The US GAAP Financial Reporting Taxonomy is a list of computer-readable tags in XBRL that allow companies to label the vast amounts of financial data that are included in typical long-form financial statements and related footnote disclosures. This tagging provides greater uniformity and helps increase the transparency and accessibility of the information. The SEC requires domestic and foreign companies using US GAAP and foreign private issuers using International Financial Reporting Standards (IFRS) to provide their financial statements in the XBRL format as an exhibit to their periodic and current reports and registration statements, as well as to transition reports needed to be filed because of a change in fiscal year.

"In June 2018, the SEC adopted amendments to require the use, on a phased in basis, of Inline XBRL for operating company financial statement information and fund risk/return summary information. Inline XBRL is a structured data language that allows filers to prepare a single document that is both human-readable and machine-readable, so that filers need only prepare one Inline XBRL document rather than generate an HTML document of their financial statement information or risk/return summary information and then tag a copy of the data to create a separate XBRL exhibit. For data users, Inline XBRL provides an easier way to view, access, and explore the contextual information of the underlying data. For example, users can click on individual tagged data points in the filing to find more information about the data, such as citations and hyperlinks to the relevant accounting guidance, narrative definitions for the values, and reporting period information associated with each value.

Viewing Inline XBRL filings does not require any specialized software, because the Commission has incorporated an Inline XBRL Viewer into the Commission's Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system. Anyone using a recent standard internet browser can view an Inline XBRL filing on EDGAR. The short video describes some of the features and capabilities of the open source Inline XBRL Viewer."[1]

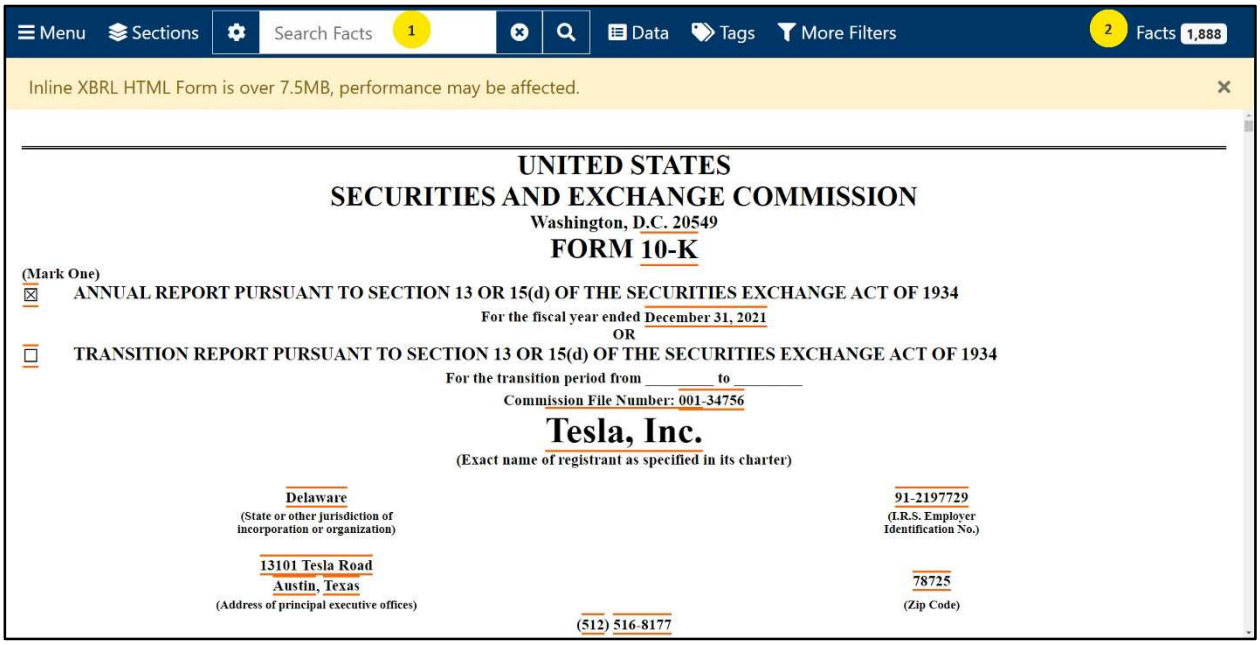
Exploring Tesla's Inline XBRL filing in EDGAR

Inline XBRL filings are searchable documents that allow easy navigation of a financial statement. In this case, we will use the search features of Inline XBRL to search key facts about Tesla's revenue in their 2021 10-K (annual) filing. This case will provide insight into both Tesla's financial statements and their footnotes. You can access the filing here: Inline XBRL Viewer (sec.gov).[2]

[1] U.S. Securities and Exchange Commission website, accessed October 11, 2022; https://www.sec.gov/structureddata/osd-inline-xbrl.html

[2] See https://www.sec.gov/ix?doc=/Archives/edgar/data/1318605/000095017022000796/tsla-20211231.htm accessed October 11, 2022.

The image below is a screen shot of the Inline XBRL filing for Tesla. To navigate the Inline XBRL filing you can use the search bar at the top of the filing (see 1 below) and the search results appear in the facts section (see 2 below). Inline XBRL filings are navigated through by clicking on the links which appear in the facts section. These links take you to the section of the filing that discusses each of the facts.



Required

Explore Tesla's 2021 10-K Inline XBRL filing in EDGAR and perform the following:

- Enter the accounting standard for revenue recognition "ASC 606" into the search bar.
  - Confirm that the number of facts decreases from 1,888 (all the facts in the filing) to 3.
  - Click the Facts section and confirm that you can find the text blocks relating to "Significant Accounting Policies," "Revenue," and "Digital Assets Net."
  - Click on "Revenue" and read the section under "Automotive Sales Revenue" In the second paragraph confirm that Tesla records a sales return reserve for estimated variable consideration related to future product returns.
- Now you are ready to explore Tesla's revenue in more depth. Go back to the search bar and type in "Revenue" and use Inline XBRL to search through the 176 facts about Tesla's Revenue. Answer the following questions. For each question, note how you found this information:
  - How much revenue did Telsa earn in 2021?
  - What are the two main sources of revenue for Tesla?
  - Does Tesla have an account for returns of automobiles?
  - Explain Telsa's Automotive Regulatory Credits and why they count as revenue for the company.
  - Did Tesla accept digital assets (i.e., Bitcoin) from customers? Where do they disclose this?
    - How much Bitcoin does Tesla hold?

- Did Tesla recognize revenue from the sale of digital assets, if so, how much?

- Revisit the revenue recognition discussion about the sales return reserve discussed in the case. In your own words, describe what Tesla is doing to account for the possibility of returns and then answer the two questions below:
  - Why is it necessary to have this account as an estimate?
  - Based on this footnote, do you think that many customers return their cars to Tesla? What wording in the footnote helped you reach your conclusion?
- What else can you find about Telsa's revenue that you did not expect? Make a note of your discoveries to share with the class.

Analytics mindset

EDGAR Explorer

Part 2: Exploring EDGAR Submissions API

Background

Via EDGAR, the SEC also provides submissions history and the XBRL data from financial statements (forms 10-Q, 10-K, 8-K, 20-F, 40-F, 6-K, and their variants) for a company (filer) via a Representational State Transfer (REST or RESTful) application programming interface (API).[3] Using these services, users can download data about a company's filings by interface categories that include:

- Submissions API: the recorded history of all filings for a company by form name, date, and time of filing, etc. This API can be used to gather information on all the filings made by a single company. You will access the submissions API in Part 1 of this case "Exploring EDGAR Submissions API."
- XBRL data:
  - XBRL Company Concepts API: the XBRL disclosures from a single company and concept (a taxonomy and a tag). For example, you could use this API to gather information on Accounts Receivable for a single firm over several years.
  - XBRL Company Facts API: all the company concepts XBRL data, which includes all the XBRL data contained with the financial statements. For example, you could use this API to collect full financial statement information. You will access the submissions API in part 2 of this case "Exploring EDGAR Company Facts API."
  - XBRL Frames API: aggregation of one company fact most recently filed based on a requested calendar period of time (e.g., annual, quarterly). For example, you could use this API to gather information for accounts receivable for the most recent period.

The following information outlines some common features of these APIs:

- All the SEC's APIs can be accessed via a browser or automated using various programming languages or automation software.
- The SEC's APIs do not require any form of authentication or API keys to access the data. Use of the API is, however, subject to the SEC's web site privacy and security policy. In this case, we are required to follow the SEC's programmatic downloads requirement and declare a User-Agent in our requests.
  - A User-Agent header can be declared using the name of your University and University email address.
  - Users can also only make 10 requests per second.
- The SEC's APIs return data in JavaScript Object Notation (JSON) format. According to Wikipedia, "JSON is an open standard file format and data interchange format that uses human-readable text to

[3] The SEC's API documentation is available at: https://www.sec.gov/edgar/sec-api-documentation

store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values)."[4]

  - The SEC's JSON files use a combination of objects (key-value pairs) and arrays (lists of data). In this format, the SEC JSON files will include metadata, such as the company's central index key (CIK) and company name.
  - Depending on the API being accessed, the JSON data structure can accommodate both the standard XBRL taxonomy as well as extended taxonomies that can be specific to a company and/or industry.

- The SEC also provides access to XBRL tagged financial data in several other formats. For example, the SEC provides the Financial Statements and Notes Datasets which contain XBRL data for a specific period (one month since October 2020 and quarterly before then back to Q1 2009) in tab separated value (TSV) format.[5] These files are useful for broad comparisons of entities across the entire market.

The SEC's EDGAR Submissions API is accessed via https://data.sec.gov/submissions/CIK##########.json by replacing the 10 hashes with the entity's unique 10-digit central index key (CIK). For example, using Tesla's CIK "0001318605" the submissions file for Tesla can be found at:

- https://data.sec.gov/submissions/CIK0001318605.json

The SEC's EDGAR website maintains a database of company names and tickers to help users look up entities' CIKs to access their filings. It can be found at:

- https://www.sec.gov/edgar/searchedgar/cik.htm

The submissions file contains both metadata about the entity (including CIK, company name, geographical and industry data) and a large amount of data describing the various disclosures made by the entity. Some of the information provided includes:

- The accension number (`accessionNumber`) which can be used to locate the original full-text filing on EDGAR.
- Information about the day and time of the filing, including the date of the filing (`filingDate`), the filing report date (`reportDate`) and the date and time the filing was accepted by the SEC (`acceptanceDateTime`).
- Information about the type of form filed with the SEC, including the form type (`form`), and when applicable, the items (`items`) included in the form and/or which Act (`act`) the filing is required by (e.g., the 1934 Securities and Exchange Act). Additional descriptive information about the filings including the size of the filing and whether it is an XBRL filing (`isXBRL` and `isInlineXBRL`) are included along with two variables describing the primary document related to the filing (`primaryDocument` and `primaryDocDescription`).
  - Important form types include the 10-K and 10-Q filings which contain financial statements, however, there are many file types that are more commonly seen for most companies including 8-K filings (the current report) and Form 4 filings (insider transactions).

[4] https://en.wikipedia.org/wiki/JSON
[5] https://www.sec.gov/dera/data/financial-statement-and-notes-data-set.html

- The 8-K filing reports on current material events, which is further classified into various types of events using an item number. For example, Item 4.01 relates to changes in the entity's auditor and Item 9.01 relates to the provision of financial statements and exhibits which often accompany an earnings announcement.

- Finally, at the end of the submission form, if the entity has filed more than 1,000 filings, there is information on earlier filings. In our Tesla example, the file references an earlier file as:
  - `"files":[{"name":"CIK0001318605-submissions-001.json","filingCount":265,"filingFrom":"2005-02-17","filingTo":"2012-12-17"}]`
  - Note that as Tesla makes future filings, the submission file on the API will push older filing information into the prior submission's JSON file. This means that the filing to date and count will change in the current file.

Required

- Pick a company and find its CIK using the EDGAR Company Filings CIK Lookup Tool at https://www.sec.gov/edgar/searchedgar/companysearch.
- Using Google Colab (or another Python integrated development environment (IDE)), download the submissions filing for your company. Remember to declare your University Name and University email address in the User-Agent header. An example of the Python syntax to achieve this is provided in the Appendix.
- The following requirements relate to understanding the structure of the submissions data. See the Appendix for hints on possible ways to use Python syntax to achieve these steps:
  - Use Python syntax to:
    - Confirm that the JSON object returned is a dictionary.
    - Identify the keys associated with the top-level dictionary and the data type for each of the values associated with these keys.
    - Identify where the data relating to the type of forms filed by the entity are in the data object (hint: it is a list nested within several dictionaries). Save the list of forms filed as a pandas dataframe.
    - Identify the most recently filed form and the dates and timestamps associated with that form.
  - Using the list of forms filed by the entity (which you have saved as a pandas dataframe), use Python syntax to:
    - Identify the number of unique forms filed by the entity.
    - Calculate the number of times each form was filed based in this dataset and identify the most popular filing for your company.
- The following questions require researching forms filed with the SEC:
  - Based on your analysis above, research the type of forms that have been filed by your example company. The SEC website https://www.sec.gov/forms provides a useful starting point for your research.

- Write a brief description of the forms you are researching and when describing the forms try to identify (i) what information is being disclosed by the company, and (ii) why this information is expected to be important to users of financial information.
  - Note that a filing with /A appended to the end is an amended filing. Amended filings are corrected filings that have been previously filed with the SEC.

Analytics mindset

EDGAR Explorer

Part 3: Exploring EDGAR Company Facts API

The SEC's company facts API is accessed via https://data.sec.gov/api/xbrl/companyfacts/CIK##########.json by replacing the 10 hashes with the entity's unique 10-digit central index key (CIK). For example, using Tesla's CIK "0001318605" again, the company facts file for Tesla can be found at:

- https://data.sec.gov/api/xbrl/companyfacts/CIK0001318605.json

The company facts file contains both metadata about the entity (the CIK and Company Name), and a large amount of data included under the header 'facts'. Nested inside 'facts' is the document and entity information ('dei') and financial statements prepared under US GAAP ('us-gaap'). Financial data is provided within the 'us-gaap' section and includes:

- Information on each account, including the label and description of the account, for example, in our using Tesla's company facts file, the header data on Accounts Receivable can be seen as follows:
  - `"us-gaap":{"AccountsAndNotesReceivableNet":{"label":"Accounts and Financing Receivable, after Allowance for Credit Loss","description":"Amount, after allowance for credit loss, of accounts and financing receivable. Includes, but is not limited to, notes and loan receivable."`
- Following the descriptive data, which appears for every account, financial data is contained under the heading 'units' which contains information on the currency the account is reported in, the amount and information on the period and form the account relates to. In our Tesla example:
  - `"end":"2021-12-31","val":299000000,"accn":"0000950170-22-000796","fy":2021,"fp":"FY","form":"10-K","filed":"2022-02-07","frame":"CY2021Q4I"`
  - Note here that the currency is US Dollars, and that Tesla's accounts receivable as at the 31st of December (`"end":"2021-12-31"`) was $299 million (`"val":299000000`) which relates to the fiscal year 2021 (`"fy":2021,"fp":"FY"`) as reported in the 10-K (`"form":"10-K"`) which was filed with the SEC on February 7th, 2022 (`"filed":"2022-02-07"`).
- Whereas the data does not appear easy to read for us as humans, it is highly structured, which is great for computer processing.
- To prepare this data for analysis will require extracting data from the nested JSON structure. Fortunately, as the data is provided in this structure for all entities reporting to the SEC, the extract, transform, and load task is a great candidate for automation.

Required

- Pick a company and find its CIK using the EDGAR Company Filings CIK Lookup.
- Using Google Colab (or another Python integrated development environment (IDE)), download the company facts json file for your company. Remember to declare your University Name and University

email address in the User-Agent header. An example of the Python syntax to achieve this is provided in the Appendix.

- Extract the financial statement data from the SEC's Company Facts API and save the output to a csv or excel file for future analysis. This is a challenging task, below ae a few hints:
  - The json file is heavily nested, which means that the dictionary structure is one that contains multiple dictionaries (key-value pairs) within dictionaries (i.e., for a higher-level key such as 'facts', the value includes additional key-value pairs). In addition, the data arrays (lists of data) that contain financial statement data are also within dictionaries.
  - This task will require a loop. In python, the loop you can use is a "for loop."
  - As mentioned above, the financial statement data is within the dictionary 'us-gaap' which is within the dictionary 'facts' also recall that each account has a label and a unit, ideally each financial statement variable you extract will be saved as the label of the variable underscore the unit it is measured in (usually USD). For example, the cost of goods sold amounts should be stored in the variable "CostOfGoodsSold_USD." Please note that some of the XBRL labels are quite long.
- Examine the csv file output and be prepared to discuss the XBRL data. Here are some potential starting points to investigate:
  - Note that many of the accounts will not be reported in all years or quarters. Which accounts are the most populated and which accounts are the least populated? Find an example or two of each.
  - Depending on how you program the for loop, in most cases the data returned will not be sorted. Examine the various dates and the variable "frame" how might you consider sorting the XBRL output?
  - Compare your company's XBRL data to the Financial Statement data on Yahoo Finance. What do you notice about the XBRL tags? (note that there are many detailed tags available, see: https://xbrl.us/xbrl-taxonomy/2022-us-gaap/)
  - Does your firm have an amended filing (either a 10-K\A or a 10-Q\A)? If so, how would you deal with this data? Does it cover all the same XBRL as is seen in the corresponding 10-K or 10-Q?
  - What other steps would you undertake to get the data ready for analysis?

Appendix

Using Python on Google Colab

Google Colab provides a free cloud-based Python integrated development environment (IDE) allowing users to write, edit and execute Python scripts within a browser. Google Colab can be accessed here: https://colab.research.google.com/. Users are required to set up an account and/or login using Google Account (i.e., Gmail) credentials.

Google Colab provides a Python IDE that is very similar to Jupyter Notebooks, allowing users to add both code and comment (markdown) sections. The top menu of Google Colab includes "File" and selecting "New Notebook" for file allows you to start a new Python script. You can edit the name of the new notebook by clicking on the name of the file (located on the top bar) and can access any files uploaded or created in script by clicking the folder icon on the left (the folder should include a sub-folder "sample_data" even if you have not uploaded or created any files on Gooble Colab.

Google Colab provides a free cloud-based Python interface, however, for this case, any python IDE you have access to can be used to create Python scripts to access the SEC EDGAR APIs. Other popular environments include using Jupyter Notebooks within the Anaconda package for those seeking a non-cloud-based approach with similar editing characteristics.

Useful Python Syntax

This section provides some of the syntax used in the solution to this case. This section is written with some basic understanding of Python syntax:

- *First steps: Importing required modules*. Python is an open-source programming language and makes use of modules to perform various tasks. For this exploratory exercise we will require the requests, json, and pandas (often imported using the acronym "pd") modules which can be imported using the following syntax at the beginning of the script:

```
import json
import requests
import pandas as pd
```

- *Using requests to download data*. For the SEC's API, the syntax to download data can use the requests library using the get function. The get function requires a URL and as mentioned above the SEC requires User-Agent information. Often the URL and header information can be stored in a variable and used in the get function, for example:

```
url = "https://url of the API"
headers = {'User-Agent' : 'University Name  email@XX.edu'}
response = requests.get(url, headers=headers)
data=response.json()
```

- *Exploring the JSON data object*. Using the syntax above, the JSON object containing the data will be stored in the object `data`, and in Python, this object will be recognized as a dictionary. The complexity in these JSON files is that dictionaries can nest other dictionaries and lists (and lists can also nest dictionaries and lists). This means that extracting the data you are seeking can take several steps. For example:

- *Obtaining dictionary keys*. Python dictionaries provide keys which work as an index to obtain data. Remember that in the EDGAR API setting, the data associated with a key can be either a single value, another dictionary or a list. To explore nested dictionaries, you need to use Python syntax that will allow you to (i) identify the data type and nested data type, and (ii) identify the keys associated with the data (stored as values in a dictionary) for each of the nested levels. For example, you can use the Python function type in combination with print to identify whether the JSON object you downloaded from the SEC is a dictionary:

```
print(type(data))
```

Should return:

```
<class 'dict'>
```

Which suggests that the JSON object is a dictionary. To identify keys in the dictionary, the Python function `keys()` can be used on the object or on nested dictionaries. For example:

```
data.keys()

data['filings'].keys()

data['facts']['us-gaap']['units'].keys()
```

In the first example, you examine the top-level keys, which will include metadata including the CIK of the filing entity. The second two examples will recover the keys associated with nested dictionaries. In the second example, the keys of the nested dictionary `'filings'` will be returned and in the third example the keys associated with the nested dictionary `'units'` which is nested within the `'us-gaap'` dictionary, which is nested within the `'facts'` dictionary will be recovered. Those who are automation-inclined could consider writing a for-loop to recover all data types for keys within the dictionary. For example:

```
for k,v in data.items():

    dt=type(data.items())

    print (k,dt)
```

In both the submissions and company facts JSON files all the top-level keys are associated with nested dictionaries. Automation concepts will be covered in further detail in the related Innovation Mindset Case called “Innovation Mindset: Automating data extraction from EDGAR using XBRL.”

- *Using dictionary keys to find values*. Python keys can be used like an index to look-up values within a Python dictionary. dictionaries provide keys which work as an index to obtain data. As these values can be both single values (e.g., CIK and Company Name) or dictionaries (key-value pairs) or lists of values it can be useful to break the original filing into smaller datasets or objects by saving the values associated with the dictionary keys. For example:

```
data1 = data['cik']

data2 = data['filings']['recent']['form']
```

In the first example, Python will save the CIK of the filing’s entity into an integer variable as `data1`, and the second example will save a list of forms filed by the entity into the variable `data2`.

- *Using pandas dataframes and functions*. The Python module pandas provides useful functions for data analysis and data exporting functions (e.g., exporting to a csv for subsequent analysis in a different program). The first step is to read the desired data into a pandas dataframe. For example:

```
df2 = pd.DataFrame(data2)
```

```
df2 = pd.DataFrame(data['filings']['recent']['form'])
```

Note that the first example follows from the prior example and the second combines the two. In both cases, this example saves the forms filed by the entity into a pandas dataframe. There are many useful pandas functions, including `unique()`, which returns only the unique values from a list. For example:

```
udf2 = pd.unique(df2[0])
```

Another useful function in pandas is counting variables. For example, you can count the frequency of any item in a list using groupby and count. For example:

```
df.groupby(df[0])[0].count()
```

where the syntax `[0]` refers to the first item in the list. Additional guidance on pandas dataframes can be found at: https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html