Automating SEC Financial Data Analysis with Python

Tomas Milo, CPA

tomas@sigmaoptimized.com

Sigma Optimized

Introducing Tomas Milo, CPA

Sigma Optimized

- PhD candidate in Management at McGill University
- KPMG-trained CPA (Toronto office)
  - Worked in their Audit, Deal Advisory, and Data Science departments
- Started Sigma Optimized to leverage technology (not just AI!) to refine business processes
  - Automate workflows for Silicon-Valley funded startups, CPA firms, and medical clinics
  - Corporate trainings on all things data and programming
- Member of the United Nations Framework Classification (UNFC) Taskforce
  - Presented work at the United Nations in Geneva at the United Nations Economic Commission for Europe (UNECE)'s 2024 convention

My experience coding in Python

Sigma Optimized

- Primary coding language for my academic research
- Used Python to train custom neural network to make sense of financial disclosures
  - Presented to the IFRS' technical XBRL team
- Created XBRLInsights – a leading open-source financial database with text-to-SQL query capabilities *(currently undergoing a redesign)*
  - My GitHub repo if interested in other open source code
- Self taught!

Before we begin…let's talk about the elephant in the room

Sigma Optimized

- ChatGPT (and other LLMs) have *DRASTICALLY* changed the way programmers interact with coding assignments
  - "The new electricity" – Andrew Ng, PhD (Stanford)
- Learning (and working with) Python has never been easier
  - Where before things would take hours to code, now it can be done within minutes
- However, learning the fundamentals is relevant. Why? This will allow you to *ask* the relevant questions

Teaching pedagogy

Sigma Optimized

- Two simultaneous truths:
  - There are *countless* well-curated, structured (and free!) courses online that teach you how to code in Python
  - It is impossible to cover the entirety of Python programming in 90 minutes:
    - Data manipulation
    - Web scraping
    - API interactions
    - Data visualization
    - Database management
    - Text processing
    - And so, so, so much more
- So where is the value?
  - We have ~90 minutes to learn what is possible with Python given a specific focus, with the idea that these skills can be imported to most of your functions

Today’s agenda:

Sigma Optimized

Introduction to Python → XBRL overview → Downloading SEC datasets → Processing and analyzing XBRL data

Introduction to Python

Sigma Optimized

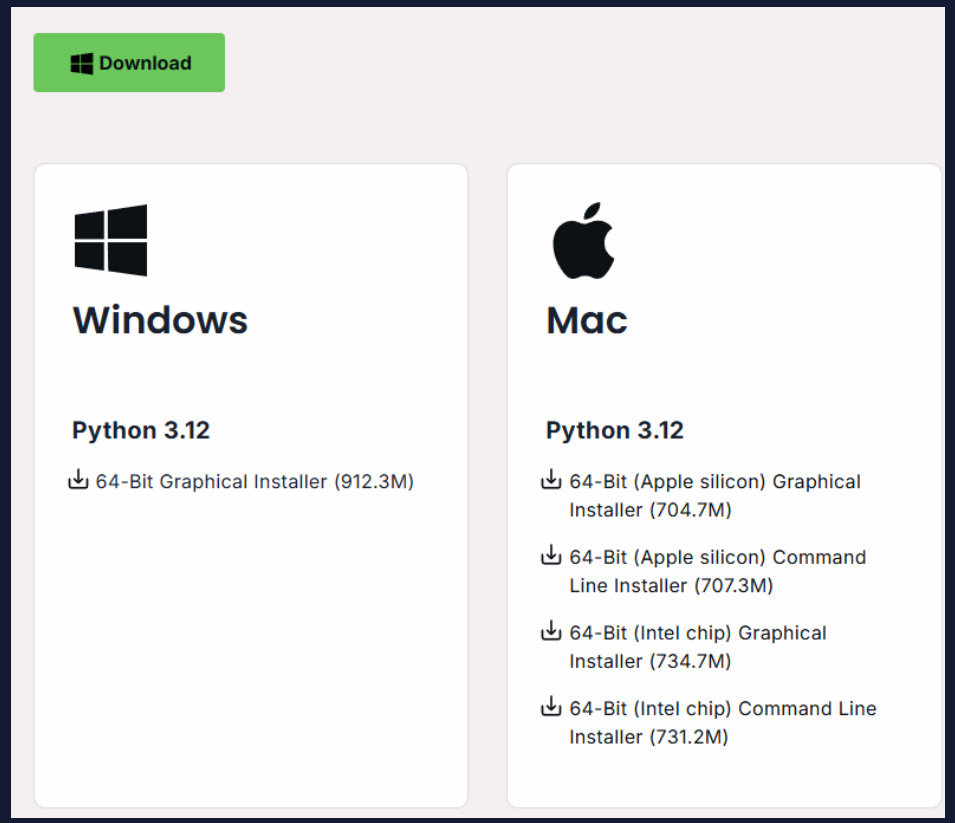
Setting up Python

- We will actually be downloading Python…but not from python.org
  - While that works, there are better ways to get us started
- Instead, we will download Python (as well as other useful tools) from Anaconda (anaconda.com)
  - “Anaconda is the complete data science platform that comes with Python, Jupyter, and popular data libraries pre-installed”
- Why is this a better choice than the simple Python download? Anaconda is designed for data analysis
- Makes it easy to manage packages and environment (more on these later)

Install Anaconda and run "Hello World"

Sigma Optimized

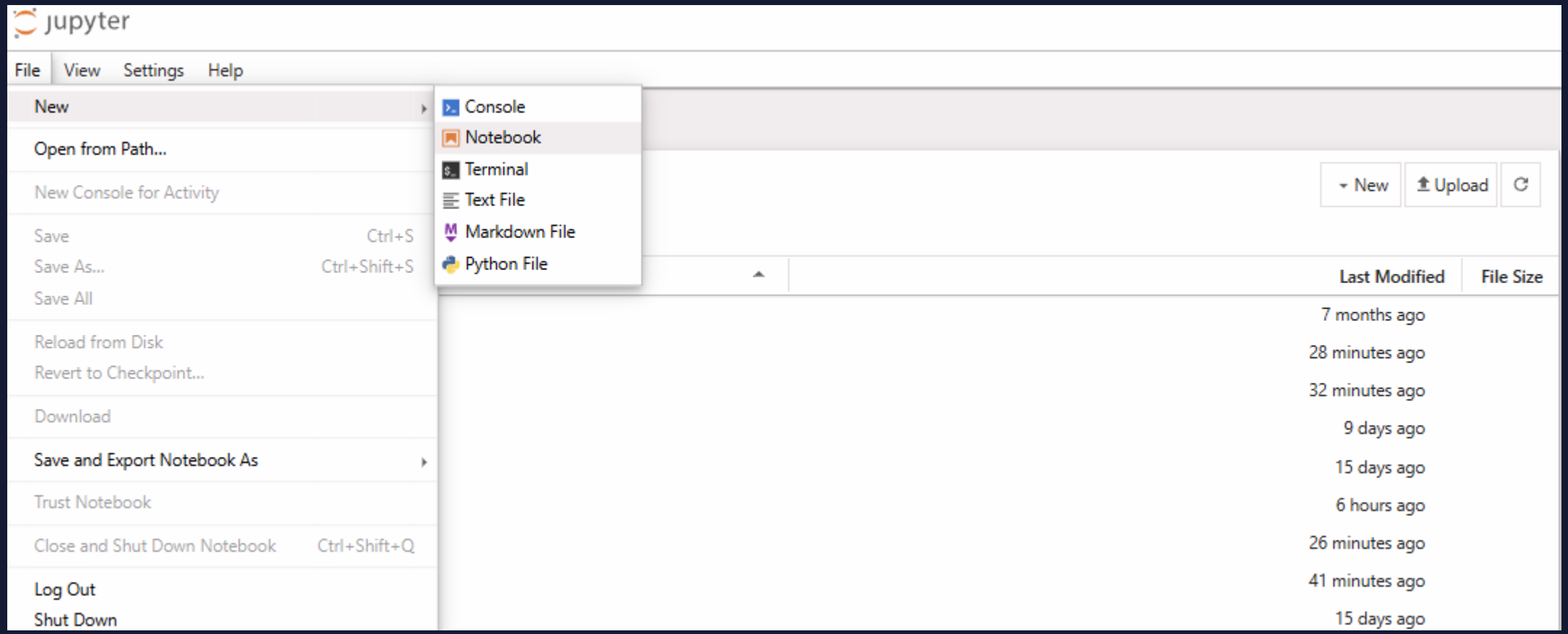
- Step 1: Download Anaconda
  - Go to: Download Now | Anaconda
  - Download Anaconda Individual Edition (Windows or Mac)
- Step 2: Install Anaconda
  - Run the installer
  - **Important!** Choose default options
- Step 3: Launch Jupyter Notebook



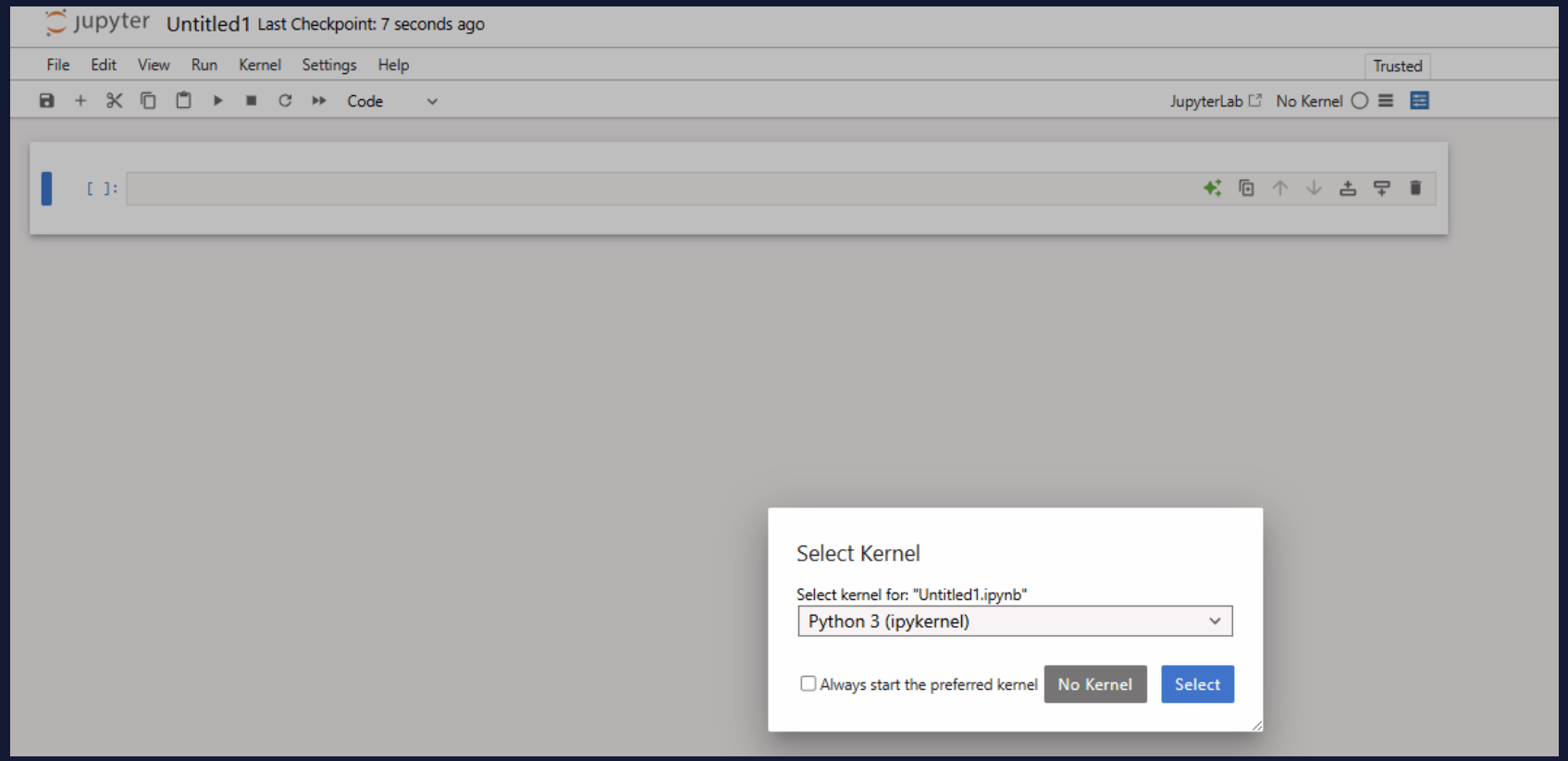
Successful install

Sigma Optimized

➢If you see something similar to the below, you are on the right track:



Creating our first Python script



What are we looking at?

Sigma Optimized

- What is a .ipynb File?
  - It's a Jupyter Notebook file
- Combines code (like a terminal) and text
  - The text makes it easier to explain the code
- Why some coders like using Notebooks to code? It runs live Python code directly in your browser
  - It keeps your results, notes, and code all together
  - *Especially if you work in a data science team!*
- Think of it as a smart financial workbook — code, data, and analysis in one place
- Access the Notebook: Google Collab – Intro to Python

First technical note: Packages

Sigma Optimized

- Packages are pre-built tools that save you time
- Think of packages as Excel functions in Python
  - When you install Excel, it already comes with lots of functions already built in…
    - SUM()
    - PRODUCT()
    - LEFT()
- When you download base Python, it comes with very few packages installed
- Anaconda pre-installs many popular packages, so you avoid manual setup

Packages are the real power of Python

Sigma Optimized

- Packages are what make Python powerful
  - We are now starting to think like programmers
- Instead of writing complex functions from scratch, we use pre-built packages
- Just like Excel has technically complex functions like VLOOKUP(), Python packages have ready-to-use tools
  - Think about it, you don't know what is 'behind the scenes' of VLOOKUP()…you just worry about calling the function and using it properly
- For example, there are many finance, data analysis, and automation packages that save us time because we're using the code that someone else already created

Which packages come pre-installed with Anaconda?

- Pandas – Data analysis and tables
  - Most famously known for ‘DataFrames’ which are like Excel tables
- NumPy – Numerical calculations and arrays
- Matplotlib – Create charts and graphs
- Seaborn – Advanced data visualizations
- SciPy – Scientific and statistical tools
- OpenPyXL – Read and write Excel files
- SQLAlchemy – Connect Python to databases

Installing our first package

- As mentioned, while Anaconda comes with many packages pre-installed, some are not
- Example: yfinance for downloading stock prices and financial data
- How to install? Very easy
  - Simply type "!pip install yfinance”

```
!pip install yfinance

Collecting yfinance
  Downloading yfinance-0.2.54-py2.py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: pandas>=1.3.0 in c:\users\tomas\anaconda3\lib\site-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in c:\users\tomas\anaconda3\lib\site-packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in c:\users\tomas\anaconda3\lib\site-packages (from yfinance) (2.32.3)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\tomas\appdata\roaming\python\python312\site-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in c:\users\tomas\anaconda3\lib\site-packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\tomas\anaconda3\lib\site-packages (from yfinance) (2.4.2)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.9.tar.gz (3.0 MB)
     ---------------------------------------- 0.0/3.0 MB ? eta -:--:--
     ---------------- ---------------------- 1.3/3.0 MB 8.4 MB/s eta 0:00:01
     ---------------------------------------- 3.0/3.0 MB 8.1 MB/s eta 0:00:00
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\tomas\anaconda3\lib\site-packages (from yfinance) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\tomas\anaconda3\lib\site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\tomas\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in c:\users\tomas\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\tomas\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\tomas\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\tomas\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\tomas\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (2025.1.31)
Requirement already satisfied: six>=1.5 in c:\users\tomas\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.16.0)
Downloading yfinance-0.2.54-py2.py3-none-any.whl (108 kB)
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml): started
  Building wheel for peewee (pyproject.toml): finished with status 'done'
  Created wheel for peewee: filename=peewee-3.17.9-py3-none-any.whl size=139096 sha256=68ebffc24bffdec43674b0b8604d1191088f2fe5c2d8904de90f0b270286865a
  Stored in directory: c:\users\tomas\appdata\local\pip\cache\wheels\43\ef\2d\2c51d496bf084945ffdf838b4cc8767b8ba1cc20eb41588831
Successfully built peewee
Installing collected packages: peewee, multitasking, yfinance
Successfully installed multitasking-0.0.11 peewee-3.17.9 yfinance-0.2.54
```

(brief) Introduction to structured financial data

Sigma Optimized

What is structured financial data?

Sigma Optimized

- Structured financial data is organized in a standardized format that computers can easily process.
  - Examples include tables in Excel, XML, and XBRL documents.
- Why it matters? Allows us to automate workflows, reduces errors, and ensures consistency in financial reporting (though this is more advanced).
- Structured data makes it easier to analyze and compare financial statements across companies.
- Regulatory bodies (e.g., SEC) require structured formats like XBRL for reporting.

How did we get here?

Sigma Optimized

- The 1980s saw the first spreadsheets
- XML was introduced in the 1990s to structure data consistently, making it easier to exchange information between systems.
- XBRL, built on XML, was developed specifically for financial reporting. Governments and regulators (like the SEC)
  - Why? XBRL helps standardize the analysis of financial statements.

What is XML?

Sigma Optimized

- XML stands for: Extensible Markup Language
- XML is used to **structure data** in a machine-readable format.
- Example: <Revenue>1000000</Revenue> represents revenue in a structured way.
  - XML is the foundation for many data standards, including XBRL
- See example to the right →

```
  <?xml version="1.0" encoding="UTF-8"?>
- <EmployeeData>
  - <employee id="34594">
        <firstName>Heather</firstName>
        <lastName>Banks</lastName>
        <hireDate>1/19/1998</hireDate>
        <deptCode>BB001</deptCode>
        <salary>72000</salary>
    </employee>
  - <employee id="34593">
        <firstName>Tina</firstName>
        <lastName>Young</lastName>
        <hireDate>4/1/2010</hireDate>
        <deptCode>BB001</deptCode>
        <salary>65000</salary>
    </employee>
  </EmployeeData>
```

Why should you care about XML data?

Sigma Optimized

1. Automation-ready: XML makes it *very* easy to integrate with tools like Python and Power Query
2. Consistent reporting: Standardized XML helps processing data across different clients and systems.
3. Future-proof: Regulators like the SEC and global bodies use XML-based formats (e.g., XBRL) for compliance reporting.
   1. This is especially important with the recent developments of AI

Introduction to XBRL

Sigma Optimized

- XBRL stands for eXtensible Business Reporting Language.
- It is a standardized framework for sharing financial data globally.
- Built on XML, XBRL adds a layer of meaning to financial data with tags that describe the context.
  - Example: <us-gaap:NetIncome> tags data specifically as "Net Income" according to US GAAP.

XBRL tags

- As we will see, XBRL documents are made up of XBRL tags
  - Think of tags as labeled cells in an Excel → they identify financial data points (e.g., revenue, assets) in a structured way.
- Each tag provides meaning, context, and units

```
<ifrs-gp:AssetsHeldSale contextRef="Current_AsOf" unitRef="U-Euros"
  decimals="0">100000</ifrs-gp:AssetsHeldSale>
<ifrs-gp:ConstructionProgressCurrent contextRef="Current_AsOf"
  unitRef="U-Euros" decimals="0">100000</ifrs-
  gp:ConstructionProgressCurrent>
<ifrs-gp:Inventories contextRef="Current_AsOf" unitRef="U-Euros"
  decimals="0">100000</ifrs-gp:Inventories>
<ifrs-gp:OtherFinancialAssetsCurrent contextRef="Current_AsOf"
  unitRef="U-Euros" decimals="0">100000</ifrs-
  gp:OtherFinancialAssetsCurrent>
<ifrs-gp:HedgingInstrumentsCurrentAsset contextRef="Current_AsOf"
  unitRef="U-Euros" decimals="0">100000</ifrs-
  gp:HedgingInstrumentsCurrentAsset>
<ifrs-gp:CurrentTaxReceivables contextRef="Current_AsOf" unitRef="U-
  Euros" decimals="0">100000</ifrs-gp:CurrentTaxReceivables>
<ifrs-gp:TradeOtherReceivablesNetCurrent contextRef="Current_AsOf"
  unitRef="U-Euros" decimals="0">100000</ifrs-
  gp:TradeOtherReceivablesNetCurrent>
<ifrs-gp:PrepaymentsCurrent contextRef="Current_AsOf" unitRef="U-Euros"
  decimals="0">100000</ifrs-gp:PrepaymentsCurrent>
<ifrs-gp:CashCashEquivalents contextRef="Current_AsOf" unitRef="U-
  Euros" decimals="0">100000</ifrs-gp:CashCashEquivalents>
<ifrs-gp:OtherAssetsCurrent contextRef="Current_AsOf" unitRef="U-Euros"
  decimals="0">100000</ifrs-gp:OtherAssetsCurrent>
<ifrs-gp:AssetsCurrentTotal contextRef="Current_AsOf" unitRef="U-Euros"
  decimals="0">1000000</ifrs-gp:AssetsCurrentTotal>
```

The role of regulators in XBRL adoption

Sigma Optimized

- Regulators like the SEC (U.S.) and ESMA (Europe) mandate the use of XBRL for public company filings.
  - Regulators around the world are making automation a priority
- Cant emphasize this enough – structured data empowers individual investors and other stakeholders to easily analyze data

How far back does XBRL data go?

Sigma Optimized

- SEC began its first phase for domestic firms in 2009, ending in 2011.
- In December 2017, IFRS-reporting foreign firms began to file their reports in XBRL
- In June 2018, SEC introduced Inline XBRL (iXBRL)
  - This *combines* 'traditional' HTML-based 10-Ks with XBRL
  - Let's look at the XBRL Viewer

Downloading SEC datasets

Sigma Optimized

Collecting the data

Sigma Optimized

- Now that we have a good foundation of what XBRL is, we can start to make sense of the available data
- The SEC makes it very easy to access the relevant data
- Simply go to: SEC.gov | Financial Statement Data Sets
- Download any of the ZIP files
- As mentioned before, the data goes back to 2009 (although most filers started in 2011)

Which data is reported on XBRL filings

Sigma Optimized

- As we will see, there are four interconnected tables:
- SUB table: Represents each unique submission. Think of this as metadata for filings: who filed, when, and identifiers linking the submission to other tables.
- NUM table: Provides the actual numeric data points from financial statements. Each row corresponds to a specific numeric fact (e.g., total assets, revenue).
- TAG table: Defines what each numeric tag means. It describes each tag, indicating whether it's a standard or custom taxonomy item.
- PRE table: Explains the presentation structure. It shows how filers visually present data—ordering and labeling items on financial statements.

How do the different tables come together?

Sigma Optimized

Dataset	Columns referencing other datasets	Referenced dataset	Referenced columns
NUM	adsh	SUB	adsh
	tag, version	TAG	tag, version
PRE	adsh	SUB	adsh
	tag, version	TAG	tag, version
	adsh, tag, version	NUM	adsh, tag, version

For those interested, you can read the 'readme' file to learn more about

Which sections of the 10-K are tagged

Sigma Optimized

- Cover page: Company information (name, address, fiscal year-end, trading symbols).
- Financial statements: Balance sheet, income statement, cash flow statement, shareholders' equity.
- Notes to financial statements: Detailed footnotes explaining financial statements.
- Auditor information: Auditor name, location, and PCAOB ID.
- This leads to the following important sections that are not tagged
  - MD&A, risk factors and business description

Processing and analyzing XBRL data

Sigma Optimized

Case study: Apple’s 2024 income statement

Sigma Optimized

- Load Apple's 2024 filing data from SEC
- Identify income statement line items
- Extract numeric values for line items
- Create Apple's 2024 income statement *entirely* from XBRL data
  - We can compare the output to the official report
- Let’s get coding: Google Collab – Apple XBRL Filings

Q&A

Sigma Optimized