WHYPRED

Data Workshop

Quilla Consulting x WhyPred

Session Map

WHYPRED

1 **Welcome**

About, Outline, Status Quo

2 **Why Python?**

What makes it the tool of choice of data analysis

3 **Environment Setup**

Preparations and introduction to Google Colab

4 **Python Fundamentals**

Variables, Data Structures and Control Flows

BIO

Michael Wang

WHYPRED

- 14 years' experience across tech, investing, and teaching
- Worked as an analyst in the financial industry for almost 10 years, then made the career switch to being a data scientist
- Has taught python, data science, investments, AI at **University of Sydney**, **CFA Society Australia** and **Chartered Institute of Professional Certifications**. Currently teach at **Kaplan Business School**
- Founded **WhyPred** in 2021; consulting in investment analytics, AI, data science and learning and development

Why Python for Financial Data Analysis?

WHYPRED

- There are many courses that teach python but the majority are taught by developers and/or data scientists with backgrounds ***rooted in tech/IT***.
- While this does give a pure, holistic approach to learning python, it may not address the need of many professionals who's occupation is not that of a developer or data scientist but still ***work a lot with data***.
- Their needs are rooted in being able to ***apply python*** to their specific use cases
- This course aims to equip you with the ***right parts of python*** to help you to ***work practically with data in your domain*** but at the same time ingrain some of the ***relevant best practices from tech***.

Why Python for Financial Data Analysis?

Domain Knowledge

Technical Expertise

Status Quo

WHYPRED

- Excel has been the cornerstone of analysis and data storage for good reason:
  - Ease of use
  - Versatile
  - Transferrable across different organizations
  - Visual interface
- But the needs of organizations and analysts have evolved and Excel may not be the answer for every problem any more
  - The amount of data we're working with nowadays have increased
  - The types of calculations needed have become more complex
  - Lack of automation and reproducibility
  - Lack of controls and governance
  - Doesn't handle unstructured data well
  - Limited analytical capabilities

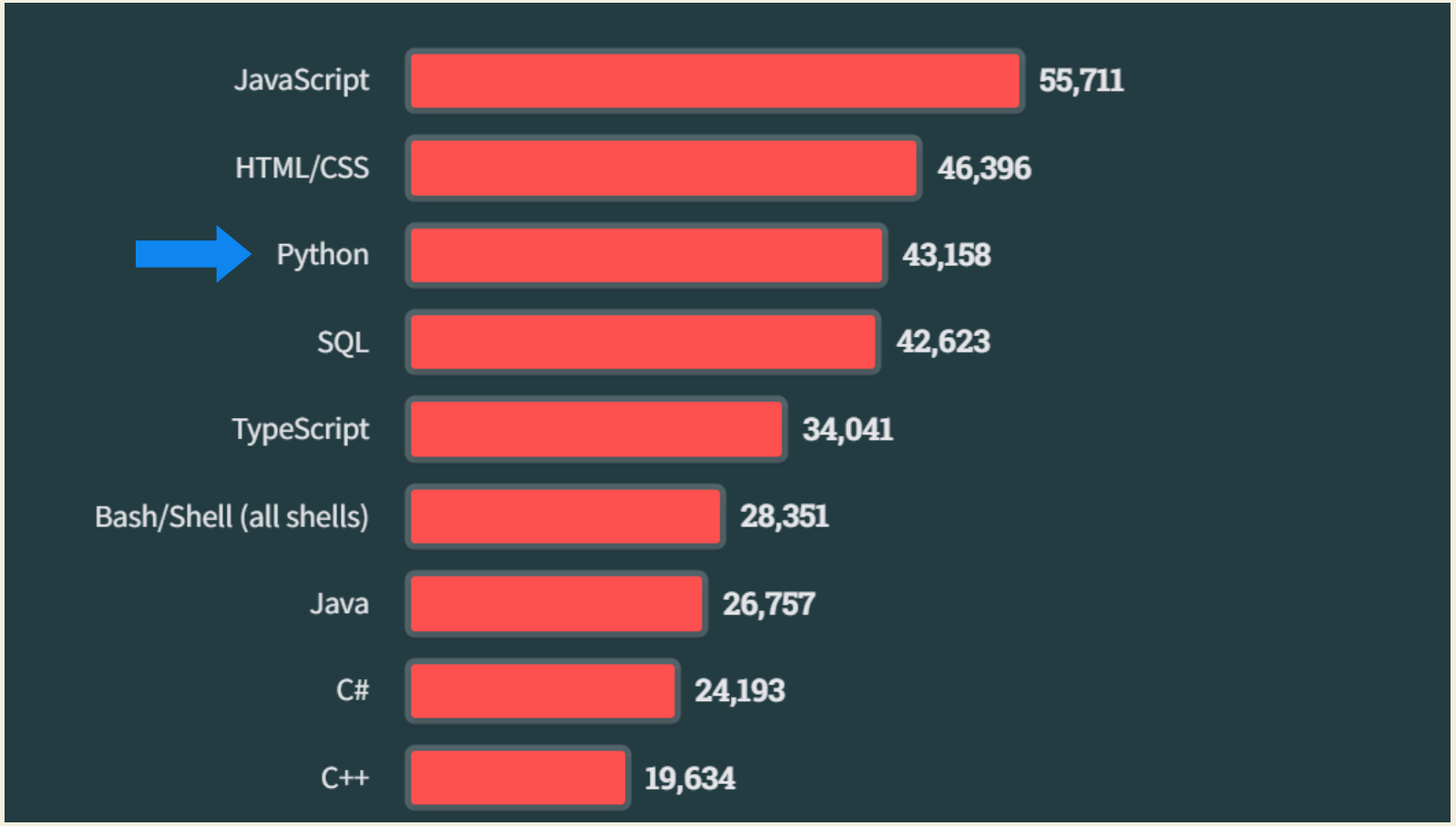
Is there a better way?

WHYPRED

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It has become the de facto language of AI, Data Science and Data Analysis

Stack Overflow Developer Survey 2023

WHYPRED



https://survey.stackoverflow.co/2023/

Features

WHYPRED

- **Automate Reporting**

  Using libraries python can streamline and automate repetitive processes like reporting and save time and reduce errors

- **Complex Calculations**

  Python enables efficient application of complex calculations on large datasets, making it ideal for financial modeling, scientific simulations, and statistical analysis.

- **Data Cleaning / Preprocessing**

  Python simplifies data cleaning tasks such as handling missing values, removing duplicates, reshaping data, and merging datasets. Its string manipulation capabilities also allow for processing of text data

- **Data Science / ML**

  Python offers rich and well-featured libraries for data science, machine learning and AI making it often the language of choice for these tasks.

Setting Up

Notebooks vs IDE

WHYPRED

There a few ways to work with Python, the easiest is with notebooks such as ***Jupyter Notebooks*** and ***Google Colab***.

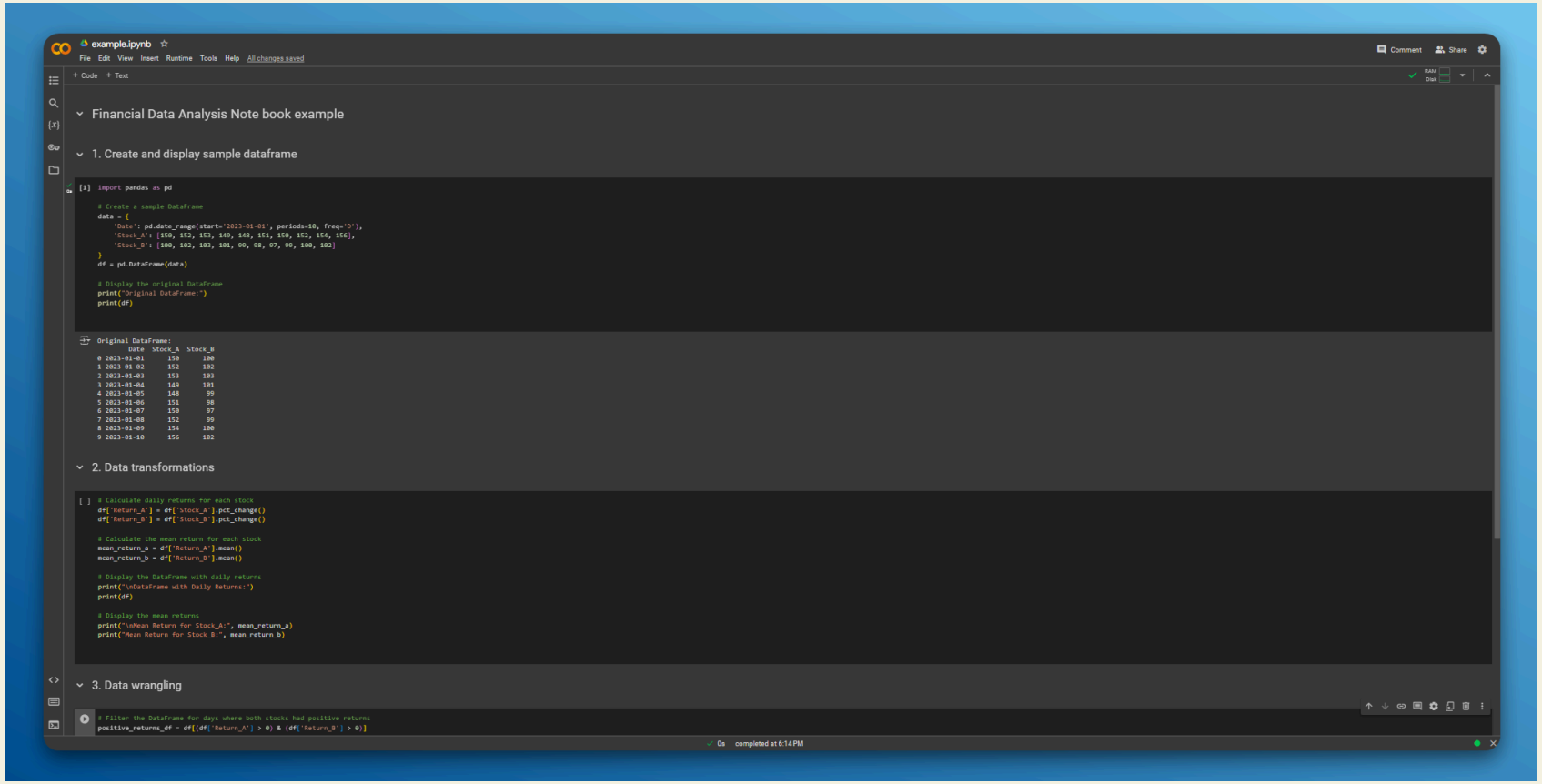
Notebooks are beginner friendly and integrate well with analytic workflows

It is recommended that as you get more proficient with python that the transition to IDEs are made.

The choice of IDE vs Notebook depends on which one you have access to in your environment and preference

WHYPRED

Notebook (Google Colab)



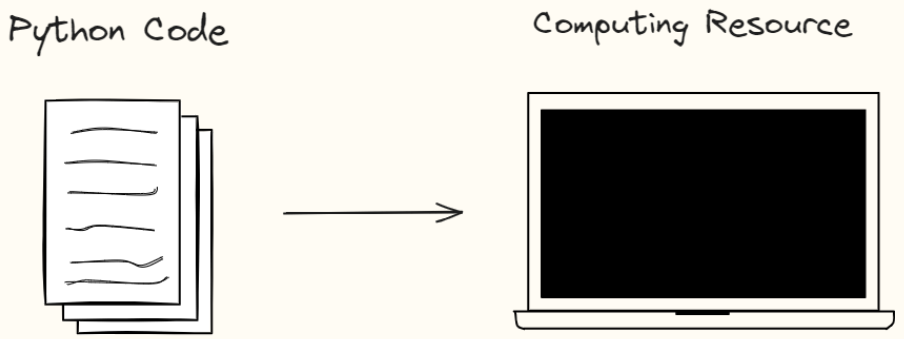
Google Colab Notebooks

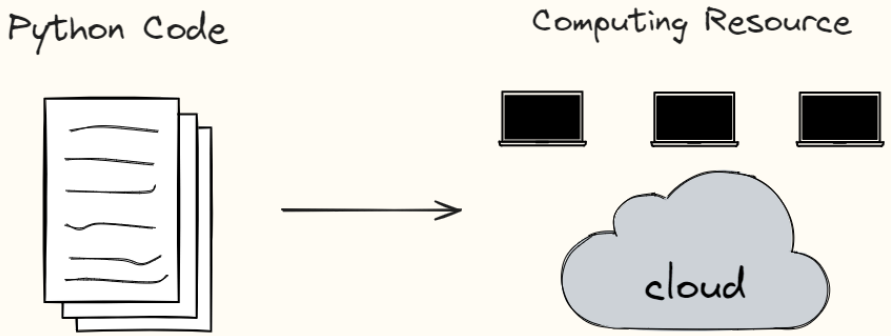
WHYPRED

- Google Colaboratory (Colab), is a cloud-based platform that allows users to write and execute Python code directly in their web browser.
- It is widely used for data analysis, machine learning, and scientific computing.
- Let's you work with python code like a regular notebooks, with code chunks, text, visualizations and charts

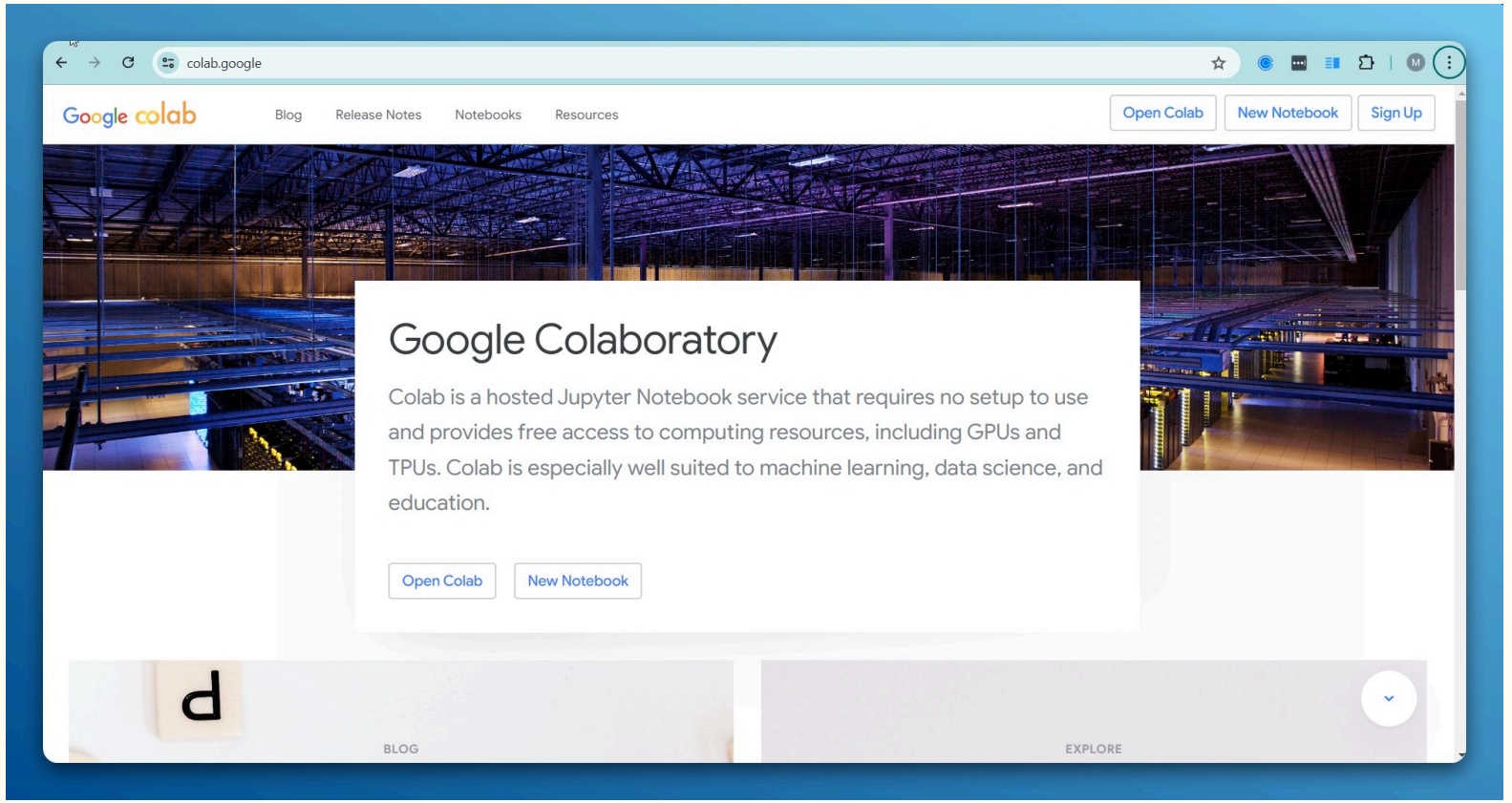
Google Colab Notebooks

WHYPRED





Getting Started



colab.google

Getting Started

WHYPRED

1.

You will need a Google Account, if you have one please make sure you're signed in if not create a Google account ***here***

2.

Navigate to colab.google

3.

Click "New Notebook" in top left corner. This will open a new notebook

4.

Let's create out first program and have a look around

Things to Know

WHYPRED

- ## Commenting

  We can leave comments in our code by using the the hash prefix "#", these act as references for you and other analysts when you have to collaborate on analysis e.g

  # this function helps us to calculate the rolling

  # average sales

- ## Indenting

  Python is sensitive to whitespaces. This is because indenting/spaces are used to determine the hierarchy of the code so extra spaces may cause the code not to run properly e.g

  print("Hello World")

  print("Hello Again")

- ## Text Values

  Text values can be created in a number of different ways, using single quotes, double quotes, or triple single quotes or triple double quotes eg

  'this is text' is the same as ***"this is text"***

  or we can have

  '''this is

  text'''

- ## Case Sensitive

  print("hello world") is not the same as ***Print("hello world")***

- ## Base 0 index

  Python starts counting at 0 not 1!

WHYPRED

Fundamentals

WHYPRED

```
############################################################
# Storing Variables

# Float
stock_price = 120.25

# Integer
stock_shares_held = 100

# String
stock_ticker = "NVDA"

# Boolean
stock_listed = True


print("Price:", stock_price)
print("Shares Held:", stock_shares_held)
print("Ticker:", stock_ticker )
print("Listed Stock:", stock_listed)


## Output

## Price: 120.25
## Shares Held: 100
## Ticker: NVDA
## Listed Stock: True

############################################################
```

Variables

- Variables are the fundamental units of storage in Python.
- They allow you to store data that can be referenced and manipulated throughout your code.
- Common types of variables encountered in data analysis include floats, integers, strings and booleans

Data Structures

WHYPRED

Data structures in Python are ways to organize and store data efficiently. They allow you to manage, access, and manipulate data in various formats

1. ***Lists:*** Ordered, mutable sequences that can contain elements of different types.
2. ***Tuples:*** Ordered, immutable sequences.
3. ***Dictionaries:*** Unordered collections of key-value pairs.
4. ***Sets:*** Unordered collections of unique elements.
5. ***Arrays:*** Efficient for storing large amounts of homogeneous data.

Operators

WHYPRED

Operators are symbols or keywords that perform operations on one or more inputs. They allow you to manipulate data and perform computations and apply logic. Operators can be categorized into several types

1. ***Arithmetic:*** Perform mathematical calculations (e.g., +, -, *, /, **)
2. ***Comparison:*** Compare values and return True or False results (e.g., ==, !=, <, >)
3. ***Logical:*** Combine True or False expressions, such as and, or, not
4. ***Assignment:*** Assign values to variables (e.g., =, +=, -=).

Control Flow

WHYPRED

The order in which individual statements are carried out in the code. It determines how a program proceeds and makes decisions based on certain conditions. Control flow structures allow you to do things like execute code conditionally, repeat parts of the code or skip to other parts of the program. Common structures in control flows include:

1. ***Loops:*** ***R***epeat operations (e.g. for loop, while loop)
2. ***Conditional Statements:*** Execute code based on whether conditions are met (e.g. if, elif, else)
3. ***Exception Handling:*** Manage errors and exceptional situations

Notebook

WHYPRED

1-python_analysis_fundamentals.ipynb

repo: ***here***

Notebook

WHYPRED

Functions are reusable chunks code of code that are defined using the ***def*** keyword

e.g. imagine we want to get the average monthly sales of business over 3 months, in python the code might look something like this:

```
total_sales = 100 + 200 + 300

average_sales = total_sales / 3
```

But we have to repeat the code for each quarter, solution:

```
def avg_sales(sales_m1, sales_m2, sales_m3):
    result = (sales_m1 + sales_m2 + sales_m3) / 3
    return(result)
```

WHYPRED

Data Formats

Data Storage Formats

WHYPRED

There are different ways that data can be stored

The most common is of course the 'csv' file or the spreadsheet

Other file formats include

- XLSX
- JSON
- YAML
- Parquet

Comma Separated Values (CSV)

WHYPRED

- It is a simple file format used to store tabular data, such as a spreadsheet or database.
- Each line in a CSV file corresponds to a row in the table, and each field in that row is separated by a comma
- the contents of a csv file might look like this but it can be opened in spreadsheets
- *Date,Transaction,Amount*

  2023-01-01,Application,1000

  2023-01-02,Redemption,-200

  2023-01-03,Application,500

Comma Separated Values (CSV)

WHYPRED

Features

- **Simplicity**: CSV files are straightforward to create and read. They can be opened with any text editor or spreadsheet software like Microsoft Excel or Google Sheets.
- **Compatibility**: CSV is a widely accepted format and can be used across different platforms and software applications. This makes it easy to share financial data between different systems.
- **Human-Readable**: Since CSV files are plain text, they can be easily read and understood by humans. This is particularly useful for quick inspections and debugging.

Comma Separated Values (CSV)

WHYPRED

Drawbacks

- **Lack of Data Types**: CSV files do not store data types. All data is treated as text, which can lead to issues when importing into systems that require specific data types (e.g., dates, numbers).
- **No Support for Complex Data**: CSV files are not suitable for storing complex data structures like nested records or hierarchical data.
- **Limited Metadata**: CSV files do not contain metadata (data about data), which means additional context about the data (e.g., units of measurement, data source) must be documented separately.
- **Inefficiency with Big Data**: CSV are not are ideal for the storage and querying of large datasets.

JavaScript Object Notation (JSON)

WHYPRED

- JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write,
- it is easy for machines to parse and generate. I
- it is recognisable by the use of curly brackets. JSON is built on two structures:
    1. **A collection of key/value pairs**
    2. **An ordered list of values**

JavaScript Object Notation (JSON)

WHYPRED

Does this sound familiar?

it should because it is exactly like python's dictionary data structure! JSON is one of the most commonly used formats in programming and for sending data over the internet.

*{ **"fund"**: "Jane Doe Capital",*

* **"horizon"**: 7,*

* **"is_open"**: true,*

* **"fum"**: 4000000,*

* **"address"**: { **"street":** "123 Main St", **"city"**: "Anytown", **"state"**: "CA" },*

* **"phone_numbers"**: ["123-456-7890", "987-654-3210"]*

* }*

JavaScript Object Notation (JSON)

WHYPRED

Features

Structured Data Representation: JSON's ability to represent hierarchical data

Interoperability: JSON is language-independent including making it a versatile format for exchanging data between different systems and applications.

Ease of Use: JSON is easy to read and write, which simplifies the process of data entry and review.

APIs and Web Services: Many financial services and APIs use JSON as their primary data format for communication.

Yet Another Markup Language (YAML)

WHYPRED

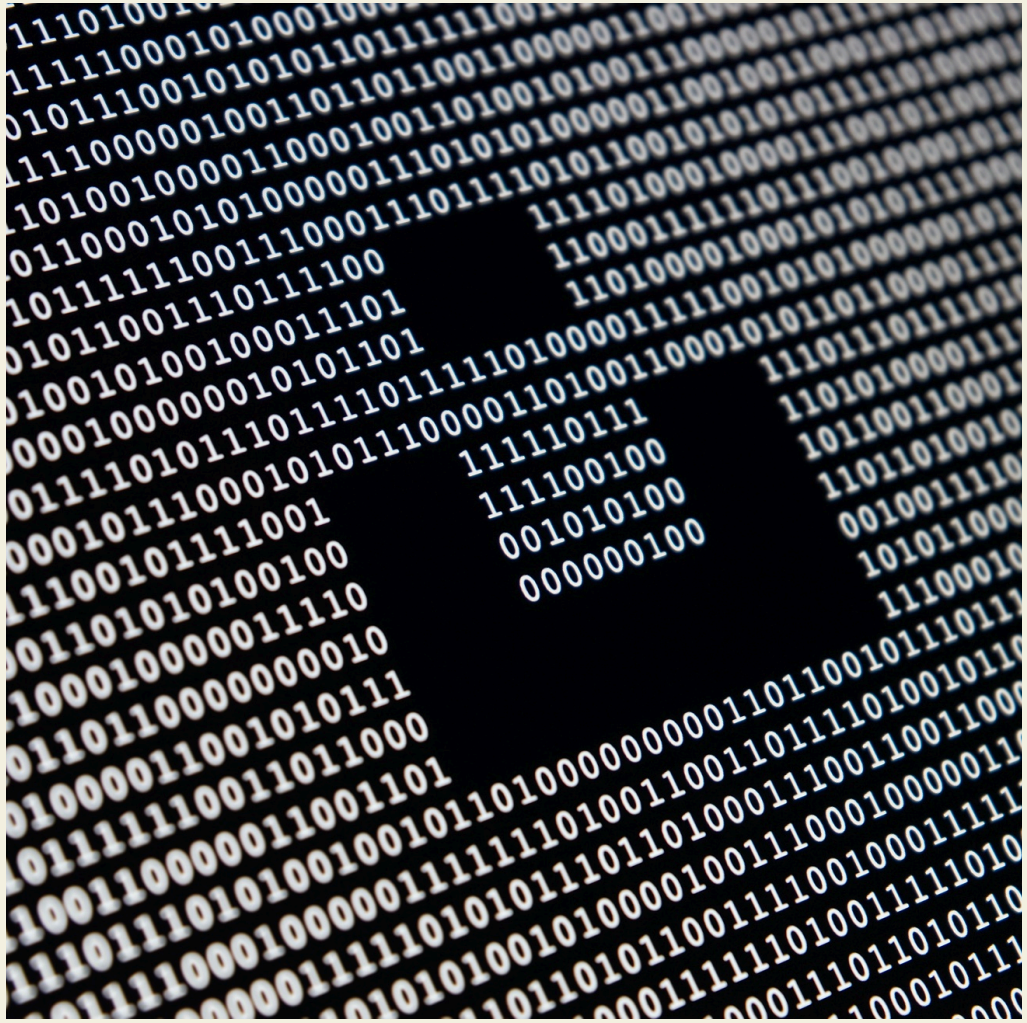
- YAML is a human-readable data serialization standard that is commonly used for configuration files in software applications, defining API and web service specifications, storing and organizing metadata.
- Their human-readable format, flexibility, and widespread adoption across various domains make YAML a versatile language.
- It is commonly used in software configuration files and IaC protocols
- ***baseCurrency: USD***

  exchangeRates:
  - currency: EUR
  rate: 0.92
  description: Euro

  - currency: GBP
  rate: 0.81
  description: British Pound

Parquet

WHYPRED

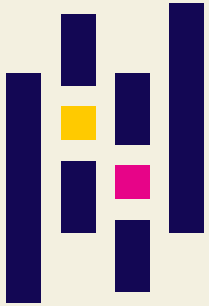


- Parquet is a columnar storage file format optimized for use with big data processing frameworks.
- It is designed to bring efficiency compared to traditional row-based file formats like CSV and JSON.
- What this means is that in parquet the file is read column by column instead of row by row, this saves a lot of time especially you know what columns you want to work with
- The parquet file format is not human readable and is stored in binary format which mean a lot of compression.
- Common pattern is to use Spark to read parquet as it is natively supported

Libraries

WHYPRED

A library is a collection of modules and that provide pre-written code to help analysts perform common tasks. Further reducing the need to write code from scratch. We use one before to help us before with creating a dataframe. We can install libraries using the ***'pip install <library name>'*** command

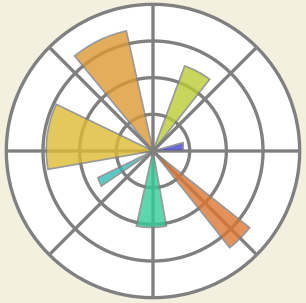


Pandas

Pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames and Series, which make it easy to work with structured data, such as financial time series data.

Numpy

NumPy is a library for numerical computing. It provides support for arrays, matrices, and many mathematical functions to help with financial computations.



Matplotlib

Matplotlib is a plotting library used for creating static, animated, and interactive visualizations in Python. It is widely used for plotting financial data.

Scikit-Learn

Scikit-Learn is a powerful and widely-used machine learning library in Python. It provides simple and efficient tools for financial data mining, machine learning.

Financial Data Analysis

WHYPRED

Modify	Making changes to the data aka **"wrangling"**	Deduping, filling missing value, changing data types.	Filtering Rows / Selecting Columns	Melt / Spread
Enrich	Creating new data using existing data aka **"feature engineering"**	Aggregation	Sampling	Calculations
Collate	Bringing multiple data sources together aka **"combining data"**	Joins	Union / Concatenate	Working with difference sources and formats

Financial Data Analysis

WHYPRED

Modify

Some common functions to wrangle/clean dataframes include:

- Slicing data - ***df.loc()*** and ***df.iloc()***
- Selecting columns with column names - ***df[["colname1", "colname2"]]***
- Filling missing values - ***df.fillna()***
- Filtering data using values - ***df[df["colname" > 100]]***
- Transforming wide data to long data - ***df.melt()***

Financial Data Analysis

WHYPRED

Modify

Long vs Wide Data

- Important concept for investment analysis
- The storage or presentation format of data is not always the best format for financial analysis
- A lot of financial data is presented in "wide" format
- data analysis and plotting tends to work better in long format
- Tidy data paper: link

Financial Data Analysis

Modify

Long vs Wide Data

```
##################################################
# Long vs Wide data
# Wide Format:

+------------+---------+---------+---------+
|    Date    | Stock_A | Stock_B | Stock_C |
+------------+---------+---------+---------+
| 2024-01-01 |   100   |   200   |   300   |
| 2024-01-02 |   101   |   202   |   303   |
| 2024-01-03 |   102   |   204   |   306   |
+------------+---------+---------+---------+

##################################################
```

df.melt()

```
##################################################
# Long vs Wide data
# Long Format:
+------------+---------+-------+
|    Date    |  Stock  | Price |
+------------+---------+-------+
| 2024-01-01 | Stock_A |  100  |
| 2024-01-02 | Stock_A |  101  |
| 2024-01-03 | Stock_A |  102  |
| 2024-01-01 | Stock_B |  200  |
| 2024-01-02 | Stock_B |  202  |
| 2024-01-03 | Stock_B |  204  |
| 2024-01-01 | Stock_C |  300  |
| 2024-01-02 | Stock_C |  303  |
| 2024-01-03 | Stock_C |  306  |
+------------+---------+-------+

##################################################
```

Financial Data Analysis

WHYPRED

Enrich

Some common functions to aggregate and enrich dataframes include:

- Summary statistics - ***mean()***, ***sum()***, ***std()***, ***median()***
- applying transformations to groups- ***df.groupby() and tranform()***
- rolling calculations for time series - ***rolling()***
- Applying a function row-wise of column-wise- ***apply()***
- using a custom calculation - ***lambda***

Financial Data Analysis

WHYPRED

Collate

Some common functions to collate data :

- different formats - ***pd.read_csv()***, ***pd.read_excel()***, ***pd.read_json()***
- Join multiple dataframes together on a column- ***pd.merge()***, ***df.join()***
- Concatentating row/column dataframes - ***pd.concat()***, ***df.append()***

WHYPRED

Modern Data Stack

Big Data

- **Increasing Data Volume**

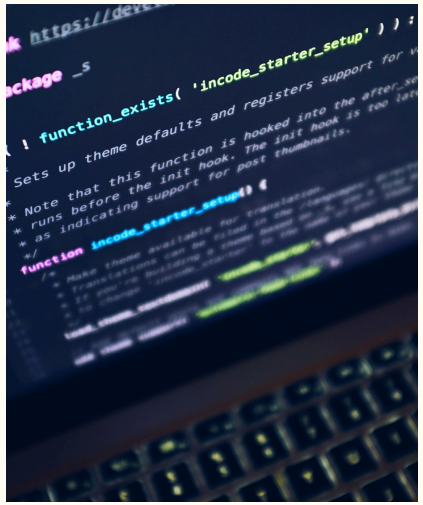
  All businesses are data businesses, having to deal with more and more data is usually a good problem to have.

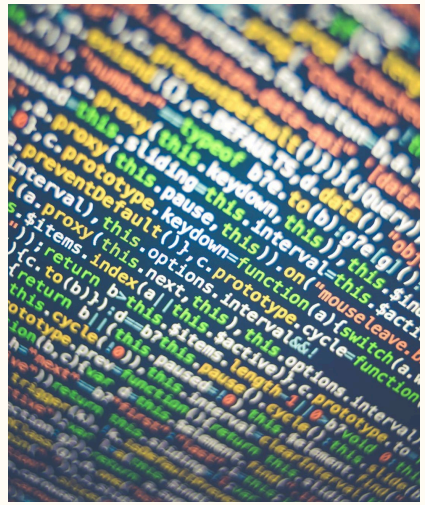
- **Financial Data**

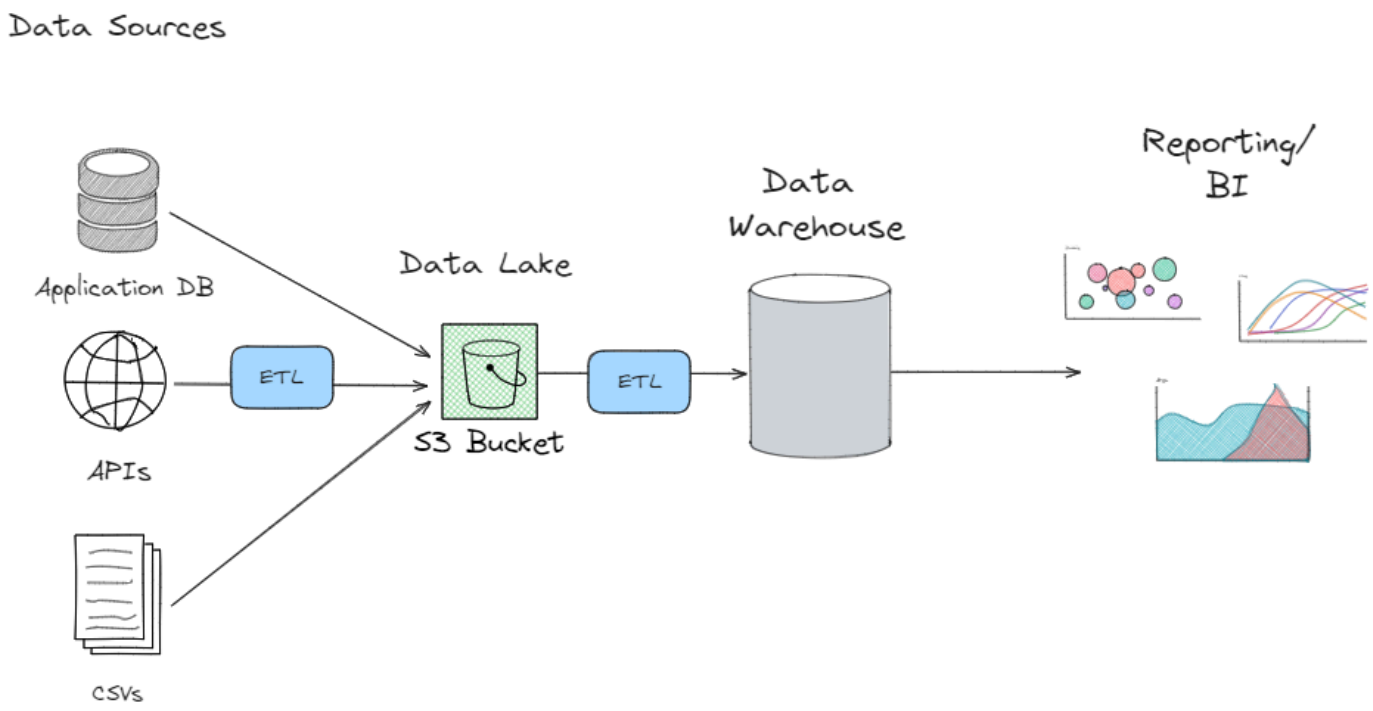
  Businesses in investment industry have traditionally consumed a lot of data, often consumed in bite sized chunks. But the paradigm is changing. Increasing size and velocity of data means that the old paradigm to managing data is becoming inadequate.

- **Modern Data Stacks**

  within tech especially SaaS, there is a generally accepted template to data management and workflows that scales.







Modern Data Stack

- **App DB vs Data Warehouse**

  Comes down to OLAP vs OLTP

- **Online Analytical Processing (OLAP)**

  OLTP are used in data warehouses, data warehouses are denormalized and optimized for read queries, complex calculations and transformation.

- **Online Transactional Processing (OLTP)**

  Used to capture transactions, simple write queries, latency is key. Ensuring integrity and consistency.

- **OLTP usage**

  OLTP are often application databases used to capture things like new user signups, customer transactions in near real time.

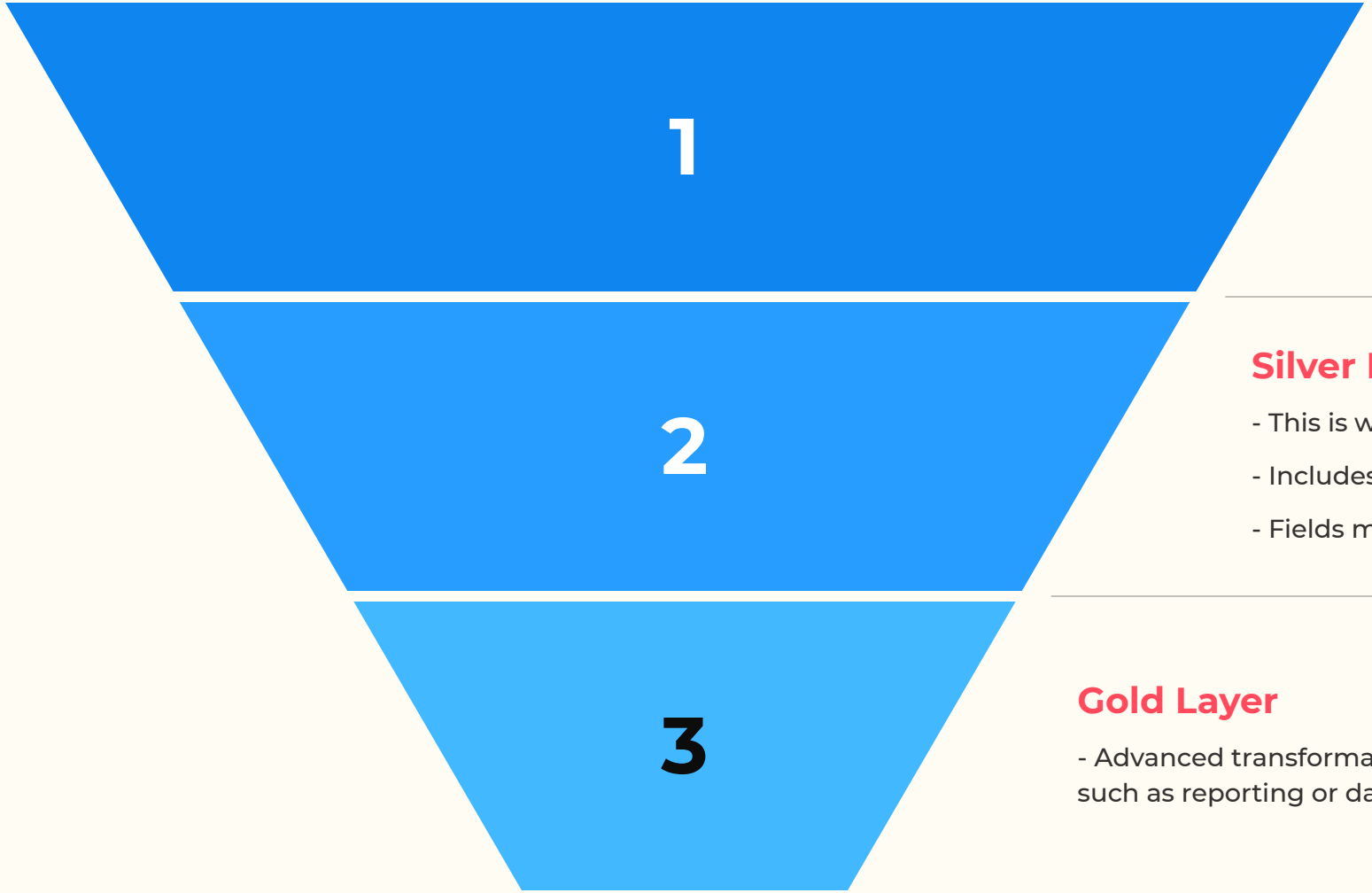
- **OLAP usage**

  OLAP are ideal for capturing historic data, ie snapshots of OLTP systems

  OLAP are also used for reporting, supporting business intelligence as well as data science use cases.

DBs

Data Warehouse Structure



Bronze Layer

- Captures raw data from
- Some simple data quality checks

Silver Layer

- This is where most of the transformations occurs
- Includes tasks like deduplication, handling missing values etc
- Fields may be added or removed here

Gold Layer

- Advanced transformations that make the data ready for the end use case such as reporting or data visualization

Data Warehouse Tools

When to Implement MDS

- When there is a clear ***Data Strategy***
- When to have a data strategy? Every business should have a data strategy!
- But realistically when data becomes a key part of the ***Organizational Strategy***
- Implementation of the MDS can be a incremental
- The key goals are ***scale, persistence, quality.***

Usage Patterns

Status Quo

On the Way to Scale

- Expanding use cases for data assets
- Increasing amount of data storage, persistence
- Workflow management becoming a priority
- Performance is a consideration

Quick Wins

Optimizing local data analysis performance

- Use ***Polars*** instead of ***Pandas***
- Written in rust, support parallelism
- Columnar instead of row wise
- Much faster than Pandas
- ***Spark*** like syntax

Quick Wins

Optimizing storage, pipeline

- Use a data lake (***AWS S3, Azure Blob***) where possible
- Storage, read, write is cheap,
- Well integrated services in cloud ecosystem like Lambda
- No need to structure data until it's needed
- Cloud based, roles and access can be managed

Quick Wins

Storage Format

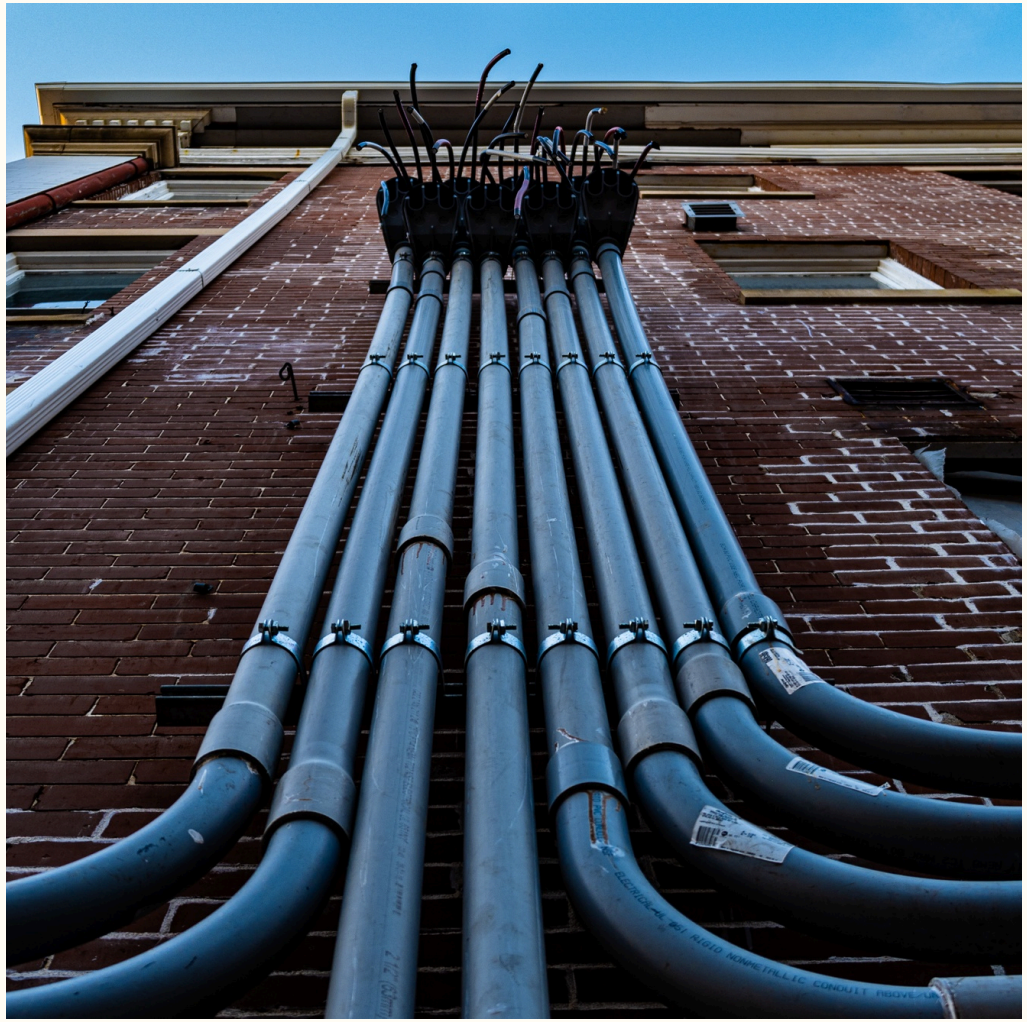
- Use ***.parquet*** files (especially larges ones) for data storage where possible
- Can be easily read into a data frame using the Pandas or Polars library if needed
- Fast reads and querying
- Has good compression, can potentially read 'out of memory' files
- use partitioning to further speed up query performances

Quick Wins

Scripting Performance

- Move scripting where possible into the cloud
- Use ***AWS Lambda*** serverless python functions
- Very generous free tier
- Can run scripts in parallel and also schedule or set triggers. It might circumvent the need for an orchestration tool in the short term
- 15 min time out and 512MB memory limit

Quick Wins



Local Orchestration

- Consider using ***Kestra*** for local orchestration
- Python support
- Free and open source
- Declarative Language with YAML templates

WHYPRED

Questions?

Disclaimer

WHYPRED

The information contained in this slide pack has been prepared by WhyPred Pty Ltd ("WhyPred") for informational purposes only. The data, analysis, and opinions expressed herein are based on sources believed to be reliable and provided in good faith, but no representation or warranty, express or implied, is made as to its accuracy, completeness, or correctness.

In addition, sample data has been used in this presentation and should not be relied upon for accuracy. This presentation does not constitute investment advice, nor is it an offer or solicitation to buy, hold, or sell any securities or financial instruments. Any projections, forecasts, or estimates herein are indicative only and subject to change without notice. Past performance is not indicative of future results.

Investors should seek their own independent financial, legal, and tax advice before making any investment decision. WhyPred and its affiliates, directors, employees, or agents accept no liability whatsoever for any loss or damage arising from any use of this document or its contents. By accessing and using this slide pack, you agree to the terms of this disclaimer.