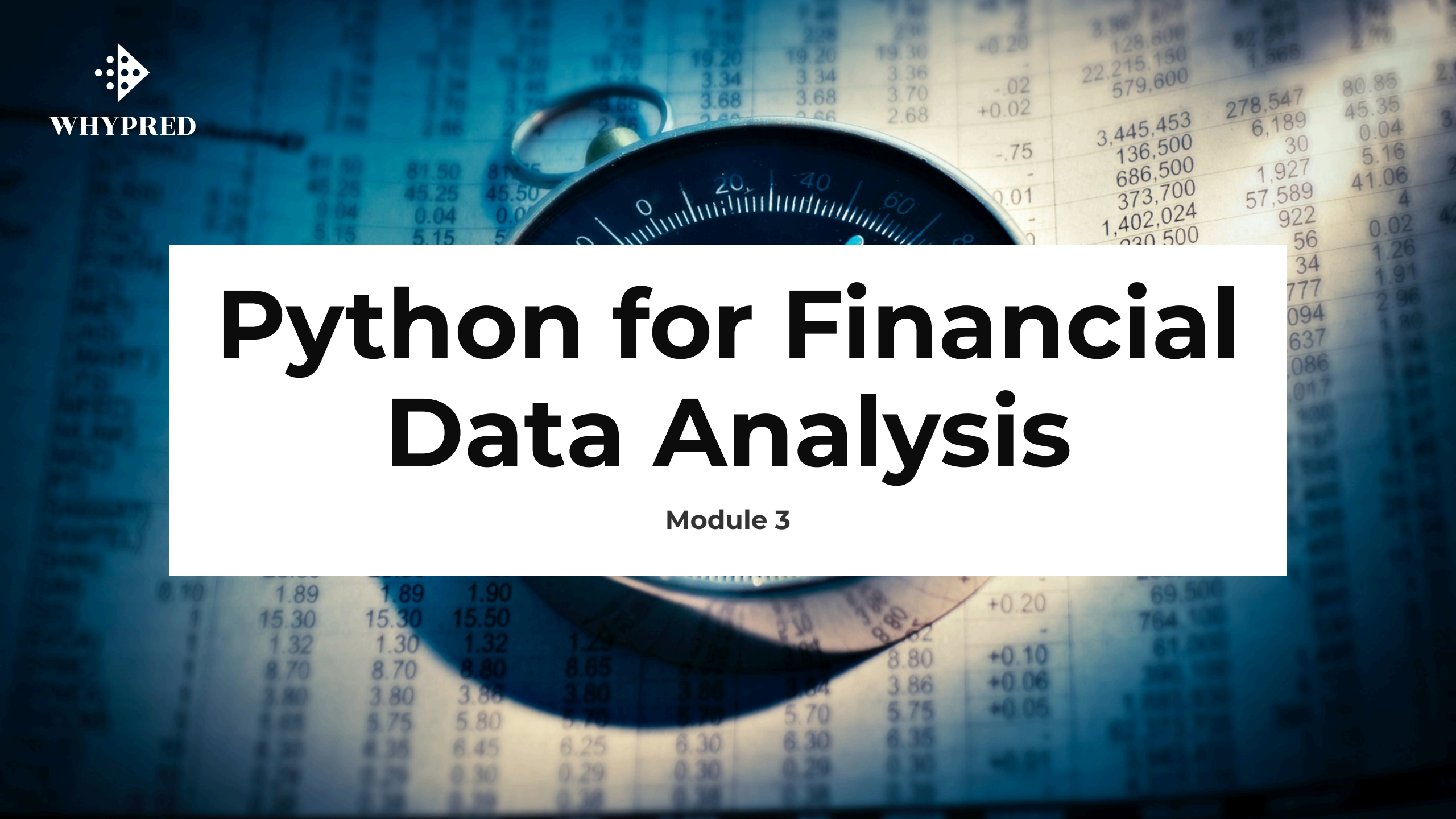

WHYPRED
Python for Financial Data Analysis
Module 3

Session Map

WHYPRED

1 **Recap**

Data storage formats, libraries, dataframes

2 **Time Series Data**

Time is money

3 **Long vs Wide data**

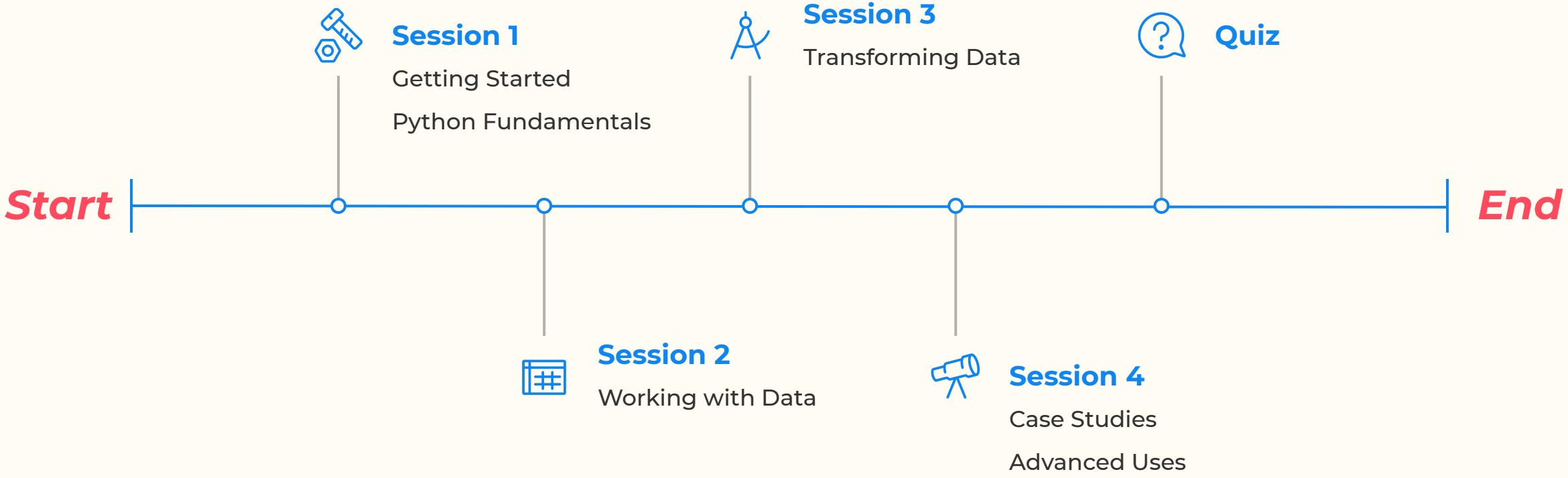
Why it makes a difference

4 **Combing and Transforming Data**

Grouping and custom logic for feature creation, Joins, concatenate, combining different data formats and structures

Course Outline

WHYPRED



Working with Time Series

Time Series

One common key attribute of financial data is that they are time series data

- They have a time and/or date stamp
- The order of the data is important and needs to be retained
- Often each observation is of the same entity but at different points in time e.g. stock prices
- There are special transformations that can be applied to time series data

Working with Dates/Times

WHYPRED

we use the ***datetime*** library in python :

- Today's date / current timestamp- ***datetime.date.today()***
- Creating date object - ***datetime.date()***
- extracting dates from string- ***datetime.datetime.strptime()***
- converting datetime object back to string - ***date_obj.strftime()***
- we can also use pandas to turn a column into a datetime- ***pd.to_datetime(df['date_str'])***
- You can also set datetimes as your dataframe indices, but this is not recommended

Long vs Wide

Long vs Wide Data

WHYPRED

Long vs Wide Data

- Important concept for financial 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**
- Untidy data -> ***column headers are values, not variable names***

Long vs Wide Data

WHYPRED

```
##################################################
# 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  |
+------------+---------+-------+

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

Combing and Transforming Data

Combining Data

WHYPRED

Some common functions to collate data :

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

Transforming Data

WHYPRED

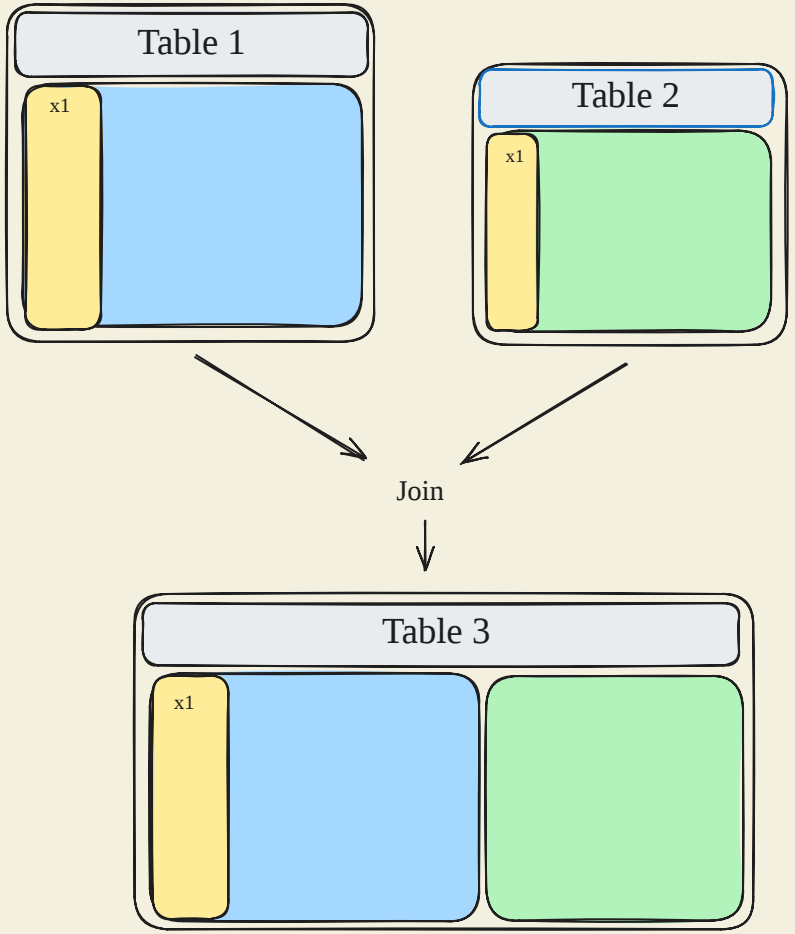
We worked with some basic aggregations of data previously such as ***mean()***, ***sum()***, ***std()***, ***median()***

- but there are times when more complex transformations are need such as applying transformations to groups- ***df.groupby() and tranform()***
- rolling calculations for time series - ***rolling()***
- Applying a function row-wise or column-wise- ***apply()***
- using a custom calculation - ***lambda***

Joins

WHYPRED

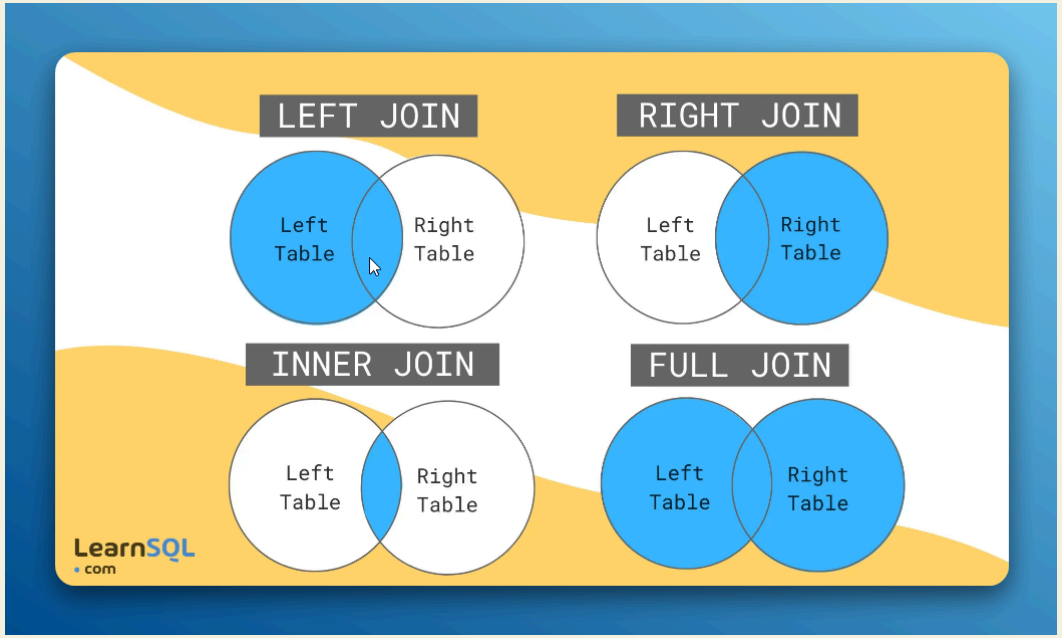
-



Joins

WHYPRED

-

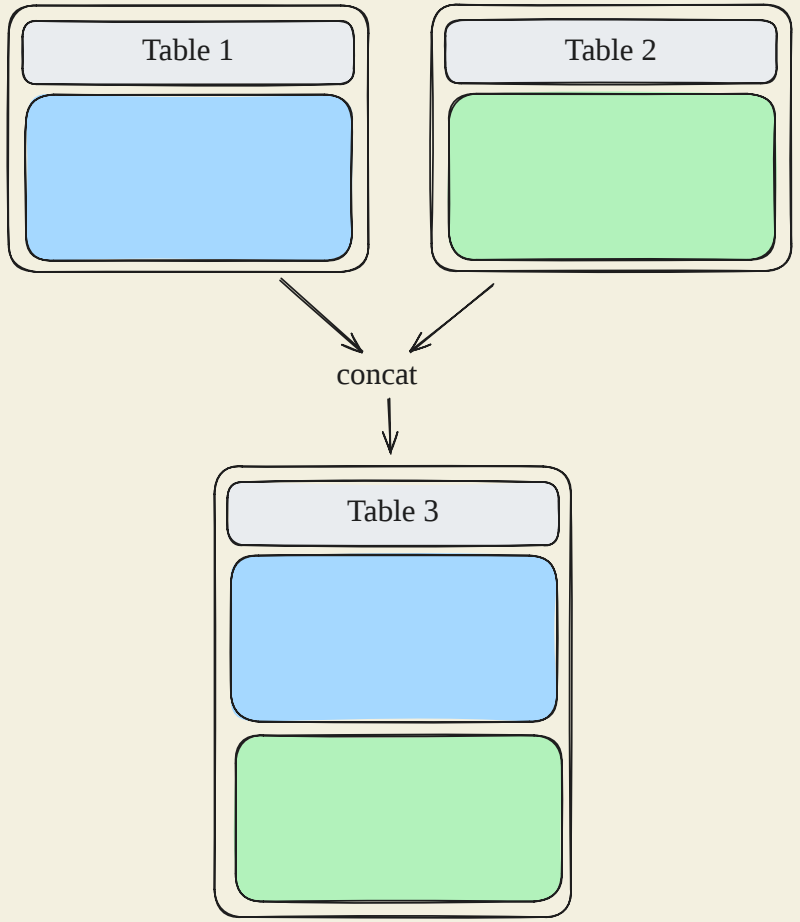


source : https://learnsql.com/blog/learn-and-practice-sql-joins/

Concatenate

WHYPRED

-



Notebook Exercise

3-1_transforming_data.ipynb

- All notebooks and slides are available ***here***
- Remember Google Colab is a shared cloud service, everyone is looking at the same notebook!
- To prevent accidental changes the notebooks are read only, at the beginning of each exercise make sure you create a copy so that you can edit your own copy!

Notebook Exercise

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)
```

That's it for module 3!

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.