B4P

Beyond Former Performance.

A powerful programming language and analytics engine
enabling rapid transformation of big data into powerful insights

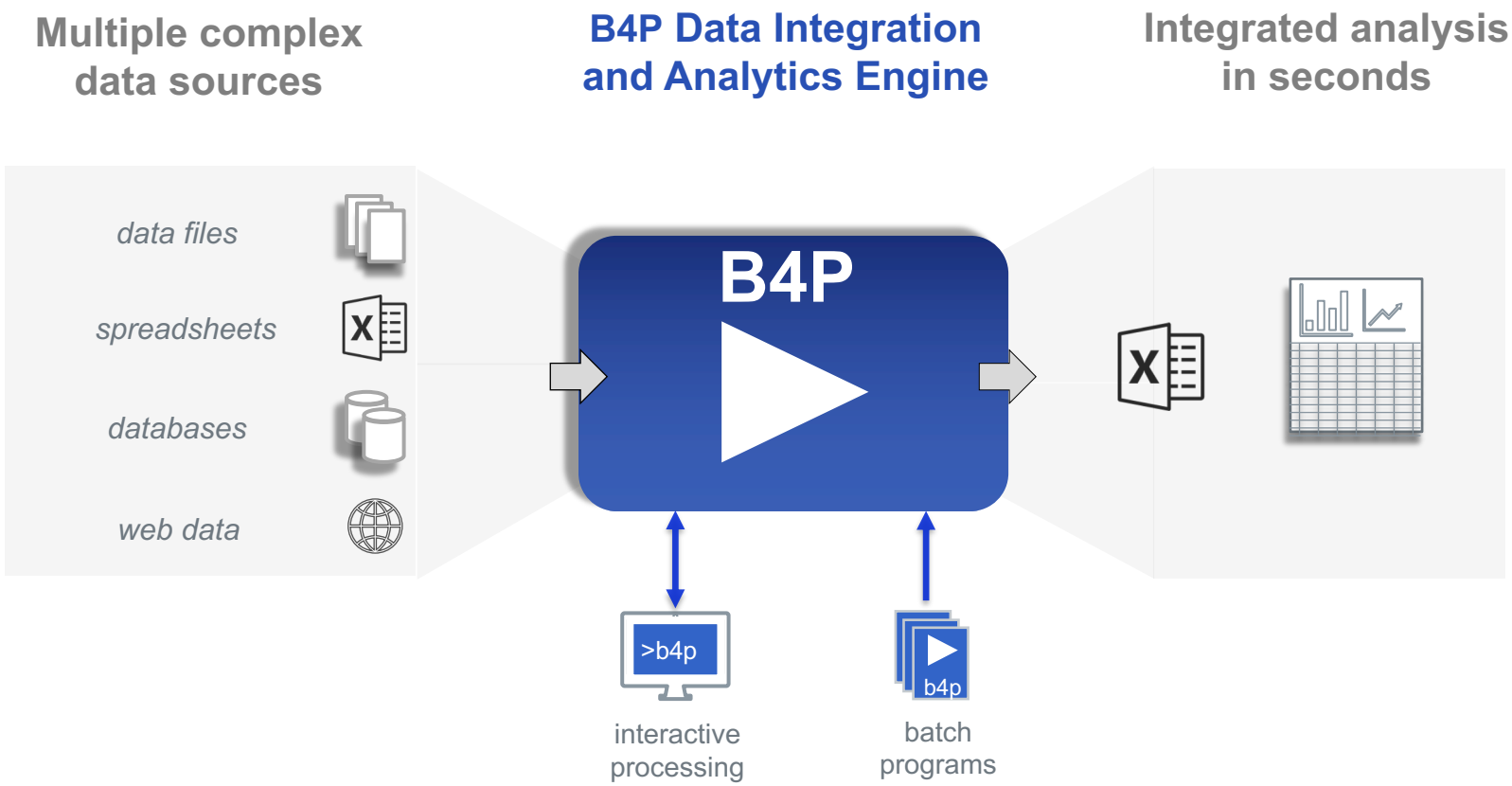
Transforming Big Data into Powerful Insights

Version 8.02 'Edouard Albert Roche' (2021-04-18)

Image source: Amir Hanna, Rush Hour in Dubai / UAE, unsplash.com.

Overview

The B4P Data Integration and Analytics Engine



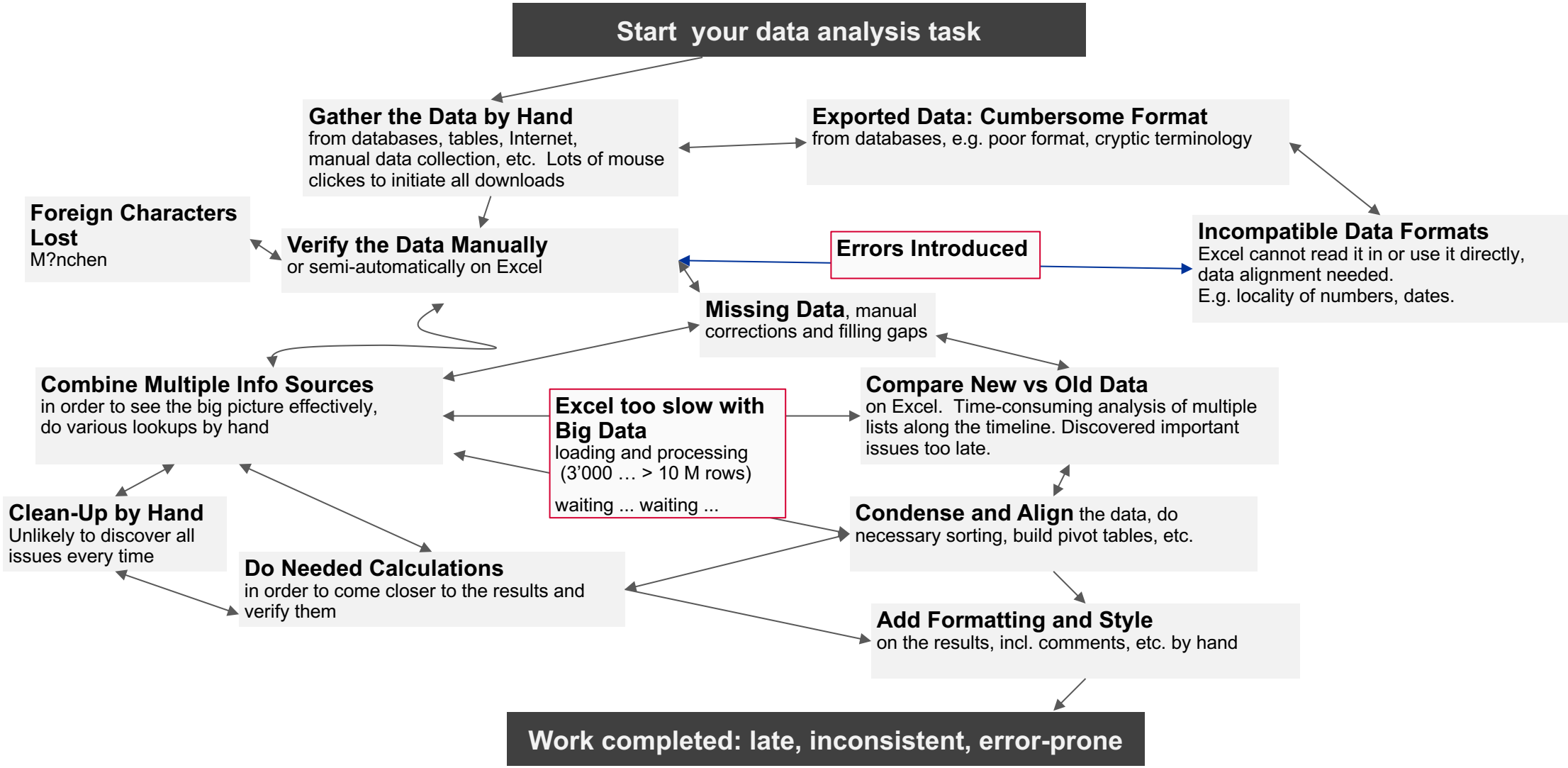
data sources

data files	Excel, XLS, CSV, XML, JSON, HTML, Zip, Text (and others)
databases	Database exports (Salesforce, Oracle, SAP, Filemaker, et al)
web data	Internet sources of structured data (websites, web services)
other data	Statistical (R, SAS, SPSS, Stata), PDF (via Tabula)

Table of Contents

Problem Statement

Manual data integration and analysis is labor-intensive and error-prone



Problem Statement

Conventional methods of analytics automation are complex and unsustainable

Write Excel Macros
(Visual Basic)

- OK for simple tasks, but ...
- ... coding becomes cumbersome if problems are more complex. Vulnerable if data format changes.
- Processing performance drops significantly when working with large data volumes.

Complex, opaque, un-auditable, poorly performing code if tasks are not very small and simple

Write a Computer Program
(C, Java, Python, etc.)

- Runs fast, but takes a lot of time to program, debug and optimize.
- Others may have difficulties to understand what you have written.
- Such programs end up very large, with many functional details coded by hand.
- Good programming know-how, ideally object-oriented programming skills are needed, as well as obtaining a suitable development environment.

Unreadable, unchangeable, opaque code cannot be created nor adapted by business users.

Hire a Consultant
(or two)

- They are happy to solve your problems for cash. Solutions are quite decent, but ...
- ... if you need further enhancements, they will ask for more cash.
- You will depend on them as they expected, and keep convincing your boss to have these expenses approved.

Expensive, external vendor dependency, no long-term sustainability

B4P Solution

Automate your data integration and analysis with a low-code analytics engine

Start Data Analysis Task

Import source data

- From files, databases and the Internet
- Excel
- JSON
- HTML, XML
- CSV, TSV, ...
- Text files and many more

Explore & validate

- Identify and extract data you need
- Check all headers
- Identify data
- Resolve inconsistencies
- Fill information gaps
- Interpolating missing values
- Identify and resolve duplications

Clean data

- Harmonize keywords and values
- Align headers and overall structure
- Align different number formats
- Align deviating terminologies

Merge data

- Combine data from multiple tables
- Eliminate redundant data

Perform lookups

- Enrich the information value in your tables

Align and Process

- Consolidate
- Analyze
- Calculate
- Pivot
- Align
- Sort

Format

- Sort and arrange rows and columns
- Add style, formats and colors

Save and Export

- HTML, XML
- JSON
- CSV, TSV
- Text files and many more

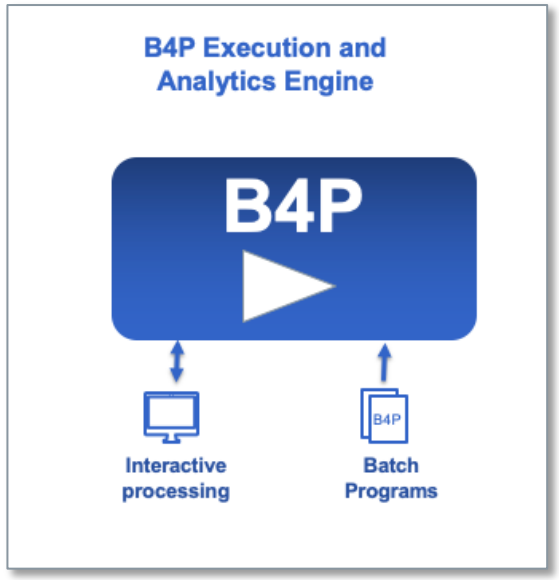
B4P

Integrated analysis completed in seconds

Rapid, High Quality Results

B4P Solution
Based on 14 Years of Experience Solving Problems in a Global Corporation

The Engine



The Language

```
table load excel file          ( football club, Football Membership List.xlsx );
table load                     ( soccer club,   Soccer Membership List.csv ); // Beginners = Novices
table rename column headers    ( football club, { Family Name, City }, { Last Name, Town } );
table process selected rows    ( soccer club,   [Level]==Novice, [Level]=Beginner );
table merge                    ( football club, soccer club, {Last Name,First Name},
                                                             {Level,Town}, append, " or " );
table sort rows                ( soccer club,   { Level, Last Name,  First Name });
table rearrange columns        ( soccer club,   { Level, First Name, Last Name, Town } );

table save excel file          ( soccer club, Soccer Club, New Soccer Club Membership List.xlsx );

echo ("New soccer club has ", table length( soccer club ), " members. Enjoy playing.");
```

- **Fast**: Runs at **full machine performance**
- Supports **many data formats** for inputs and outputs (Excel, HTML, XML, JSON, text files, etc., full UNICODE)
- Processes and delivers **accurate results reliably**
- High performance even with big data – **In seconds, not hours**
- **Styled and formatted output** for Excel and HTML (e.g. Structured tables, colors, multiple Excel sheets per file)

Principle of Low-Code Approach: Few statements suffice

- **Simple syntax**: Easy to read, learn, understand and run
- Key strengths on large **structured data tables** and **hierarchically structured variables**
- **Extensive library** with very powerful functions and features
- **Compact methods** for powerful processing steps minimizes coding loops and using variables

B4P Solution
Supported Data Formats

Inputs

Excel

- **XLSX**, **XSLM**, open formats
- **CSV** comma and tab separated files

Database Exports

- **HTML**, **MHTML** and **XML** formats (depending what the database is producing). Examples: Salesforce, Oracle, SAP
- **JSON** files (JavaScript Object Notation format)
- **CSV** comma / tab / semicolon / ...symbol separated files

Other Inputs

- Files with fixed columns on every row
- Any other form of structured text files
- **ZIP** files (B4P does data decompression)

Character Sets (both input and output)

- UNICODE UTF-8 and UTF-16; Basic and extended multilingual planes
- Legacy formats (like ASCII / Windows West Europe)

Outputs

Unformatted Output for Excel

- **CSV** comma separated files

Formatted Output for Excel (with colors, formatting and style)

- **XLSX** (Excel 2007 onwards, in use today)
- **XLS** (Excel 2003 XML format)

Unformatted and formatted output for Browsers

- **HTML** (incl. colors, formatting and style)
- **XML** (planned)

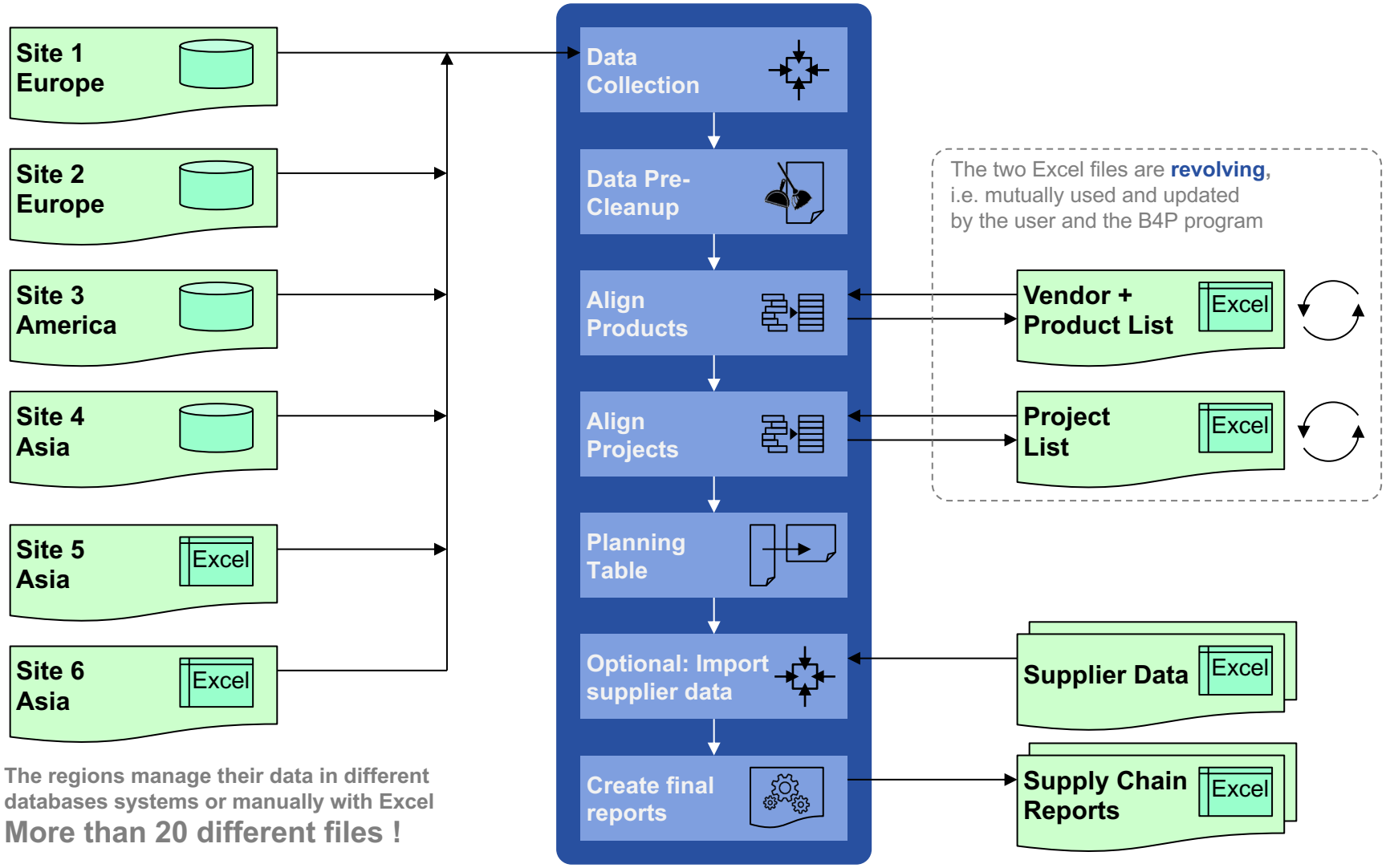
Output for other databases

- **CSV** comma / tab / semicolon / ... symbol separated files
- **JSON** files
- Plain text files
- **ZIP** files (B4P does data compression)

Additional data formats can be supported easily through B4P library extensions

B4P: Real-world Use Case #1

Corporate data integration from branch offices worldwide



The regions manage their data in different databases systems or manually with Excel

More than 20 different files !

1. Load all Demand Files
- The data from the different sites originate from different database exports or manually prepared Excel files

2. First Clean-Ups
- Harmonize data formats to week numbers and years

3. Product Alignment
- The revolving table manages the products to include and allow for using harmonized product names.
- Orientation is by common product identification number.

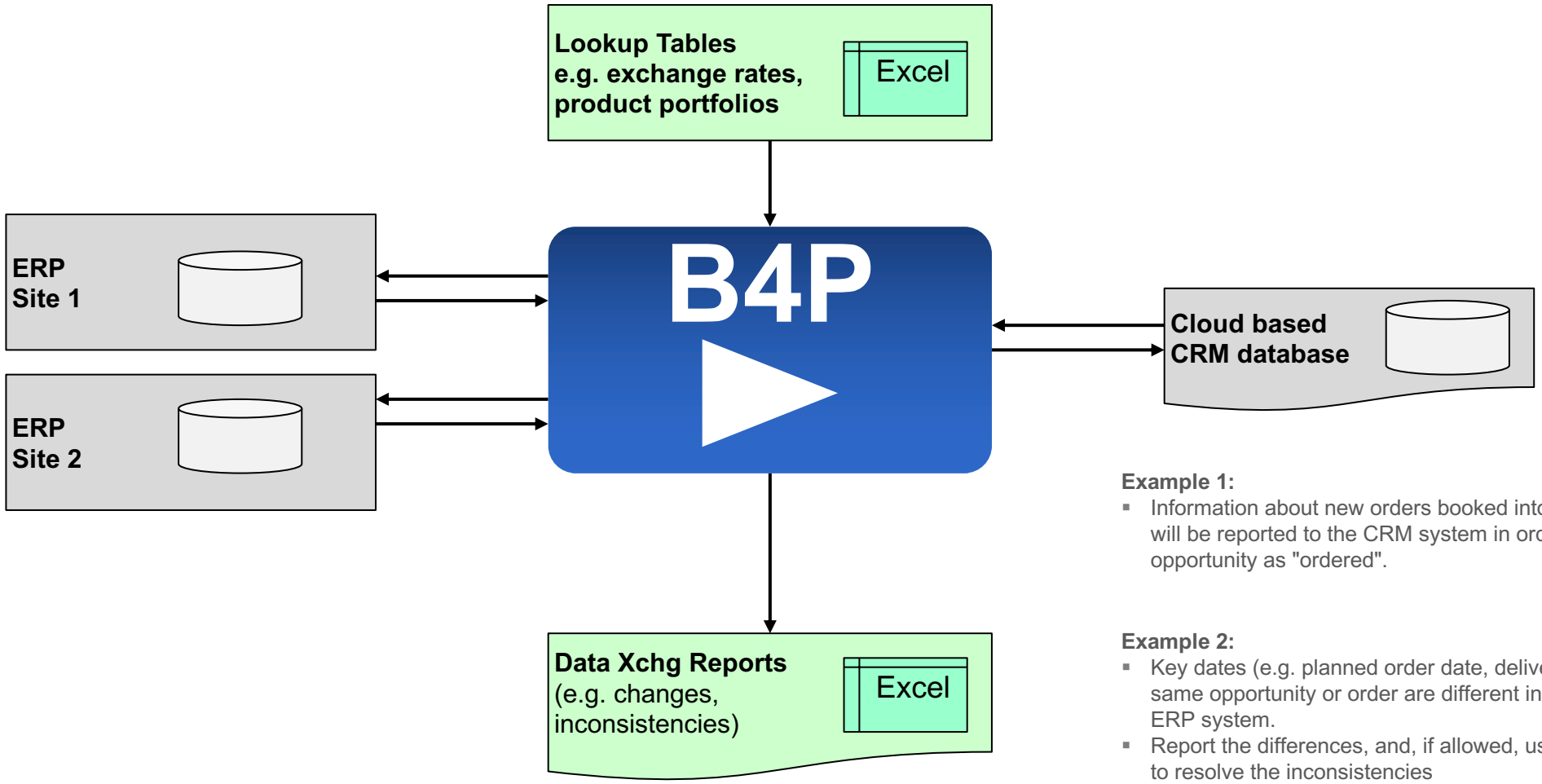
4. Project Name Alignment
- Different project names and/or abbreviations are used by the sites. They will be aligned.

5. Project Name Alignment
- The sequential list of individual demands is transformed to a horizontal planning table with weekly schedule.
- Information consolidation and summing up

6. Import supplier planning data
- If supply info is available, then match demand with their delivery plans

7. Reports
- Detailed internal reports for performance monitoring
- Condensed reports for suppliers

B4P: Real-world Use Case #2
Information interchange between multiple different databases



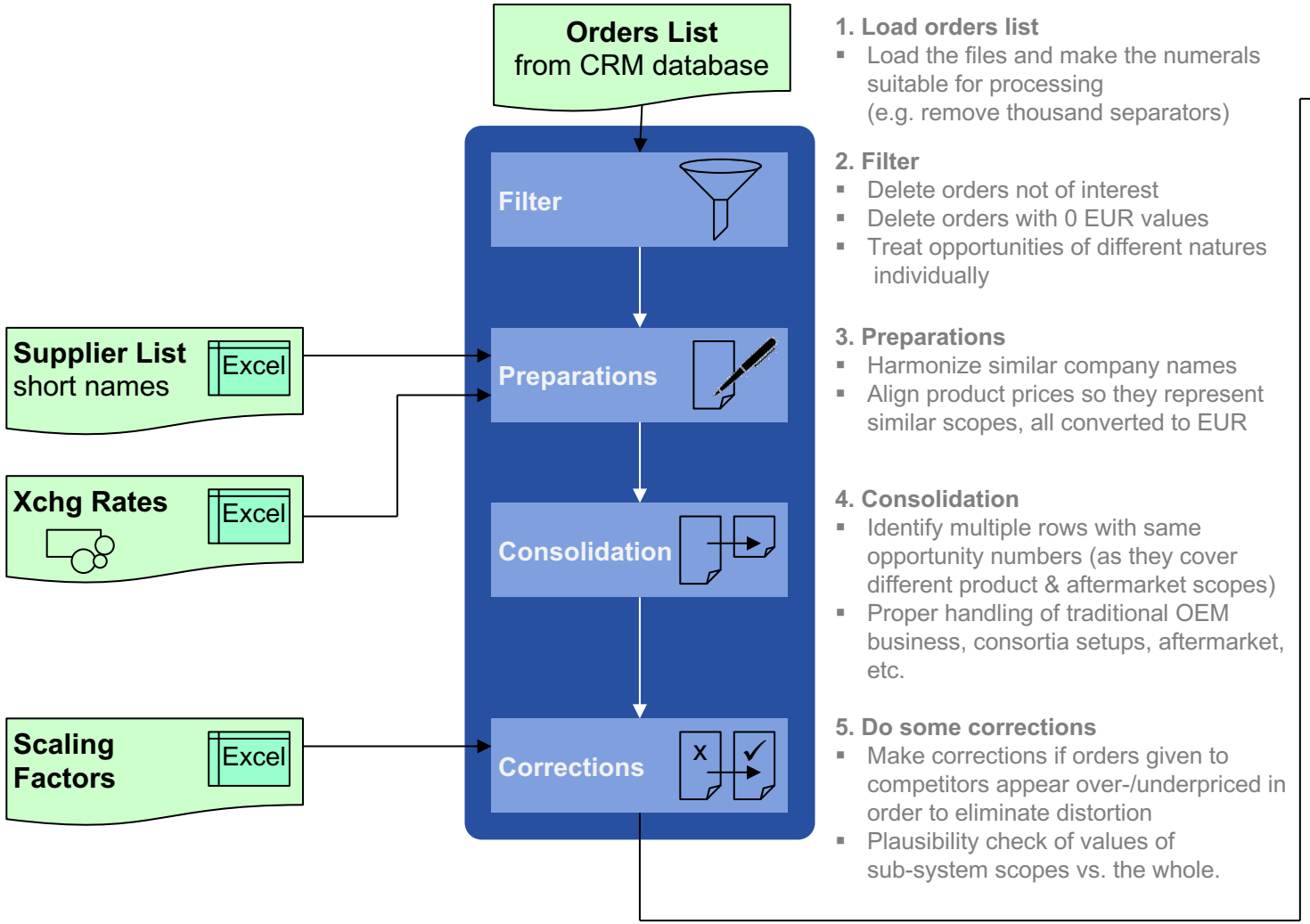
Example 1:

- Information about new orders booked into the ERP system will be reported to the CRM system in order to mark the opportunity as "ordered".

Example 2:

- Key dates (e.g. planned order date, delivery dates) for the same opportunity or order are different in the CRM and ERP system.
- Report the differences, and, if allowed, use automated rules to resolve the inconsistencies

B4P: Real-world Use Case #3
Enriched Business Intelligence from many data sources



1. Load orders list
- Load the files and make the numerals suitable for processing (e.g. remove thousand separators)

2. Filter
- Delete orders not of interest
- Delete orders with 0 EUR values
- Treat opportunities of different natures individually

3. Preparations
- Harmonize similar company names
- Align product prices so they represent similar scopes, all converted to EUR

4. Consolidation
- Identify multiple rows with same opportunity numbers (as they cover different product & aftermarket scopes)
- Proper handling of traditional OEM business, consortia setups, aftermarket, etc.

5. Do some corrections
- Make corrections if orders given to competitors appear over-/underpriced in order to eliminate distortion
- Plausibility check of values of sub-system scopes vs. the whole.

6. Expand
- Expand opportunities with multiple equipment suppliers into multiple rows and list the suppliers separately

7. Derive Equipment Values
- Derive equivalent equipment values
- Plausibility checks + adjustments against upper and lower boundaries.

8. Final cleanup
- Keep and rearrange columns of interest,
- Discard the rest
- Numeric output in commercial format

Use Excel to visualize
- Market development over time
- Market development across regions
- Market share among suppliers
- Identifying market disruptions
- etc.

ERP = Enterprise Resource Management Databases (e.g. SAP, Oracle, BAAN, Abacus)

CRM = Customer Relationship Database (e.g. Salesforce)

B4P: The Language
Key Benefits of a Low-Code Language Approach

- **Simple** procedural language
- **Easy to read and understand** the code, therefore very easy to learn programming.
- **Powerful** language semantics **keeps your program short to solve complex problem**.
- **Immediately** get your code running
- **Clear, natural language**. Give your variables, tables, functions, etc. natural names (spaces are allowed !)
- **Big Data Tables** of any size are one of the main data storage models and B4P is optimized for this.
- **No programming complexity** such as type definitions, declaring all the variables and doing memory management on your own.
- Significantly less need for fine grained programming like formulating loops, using variables, coding detailed algorithms, etc.
- B4P understands data formats such as **Excel, HTML, XML, JSON, CSV**, etc. to retrieve data from Excel, database and the Internet directly
- The execution engine and all library files are **very light-weight and lean**, very robust and start quickly.
- P**ortability** (Windows, Linux, MacOS, etc.), enabling to run the same code on any computer.
- Output Excel files **with style and formatting** like colors, row widths, etc.
- **Rich function library** with **over 800 functions**, including 200 functions for processing tables, and growing.

The B4P Language allows you to express yourself easily in plain English to solve complex problem.
Focus on the *what*, not the *how*.

B4P: The Language
Language Syntax and Semantics

- Overall language block structure similar to C / C++ / Java.
- Full and homogeneous UNICODE support.
- Tables and structured variables are the two main data storage mechanisms.
- Table names, variable names and function names are fully flexible, e.g. multiple words and spaces are allowed.
  - Full Excel support, including formatting and style.
- Variables organized in a dynamic tree, allowing to build up nested arrays and structures.
  - Load / save sophisticated JSON contents to / from the variable structure using 1 statement.
- Code pieces can be passed as function parameters which will be executed multiple time or on a on-demand basis:
  - Example: table process (...), pick if (...)
  - Benefit: Eliminates need to write loops or other details.
- Numerous flexible control flow mechanisms, going beyond the common ones like if, while, for, ...
- Cross platform compatibility: Windows / Linux / MacOS:
  - File names with directory paths are understood and interpreted correctly in other platforms (e.g. Windows vs. Linux).
  - Your program does not need to be modified to run on a different system.
- Powerful parameter set and matrix operations to process big data.

Complete solutions require only 5-20 statements

1. **Use the rich B4P function library** to process your big data. They deliver naked machine performance.
2. **Use deep operations** (vector and matrix operations) to process large amount of data inside tables and parameter sets.
3. Think how you can formulate your code in a very compact manner without compromising comprehensibility.

Program Example #1
Merging Two Tables

Football Membership List

First Name	Family Name	City	Level
Abel	*Amberstone*	*Amsterdam*	*Beginner*
Beata	*Berghill*	*Barcelona*	*Experienced*
Corinne	*Carlson*	*Copenhagen*	*Beginner*
Dietmar	*Davis*	*Dublin*	*Beginner*
Ellen	*Evans*	*Essen*	*Beginner*
Fred	*Fisher*	*Frankfurt*	*Experienced*
Gregory	*Green*	*Gaza City*	*Experienced*
Henry	*Hansson*	*Hamburg*	*Experienced*
Ida	*Ingelberg*	*Ingolstadt*	*Beginner*
John	*Janssen*	*Johannesburg*	*Beginner*
Karl	*Karlsson*	*Kansas City*	*Experienced*

Soccer Membership List

Level	Town	Last Name	First Name
Questionable	Kyoto	Karlsson	Karl
Novice	London	Lee	Linda
Experienced	Morristown	Miller	Mike
Experienced	New York	Nguyen	Nathali
Experienced	Oslo	Oliveiro	Oscar
Novice	Phoenix	Paulsson	Petra
Novice	Quebec City	Quarles	Quincy
Experienced	Riga	Richardson	Richard
Experienced	San Diego	Stewart	Sandra
Experienced	Tahoma	Turner	Tim
Questionable	Ulm	Ufford	Uwe
Novice	Venice	Viking	Victor

Merged Membership List

Level	First Name	Last Name	Town
Beginner	Abel	Amberstone	Amsterdam
Beginner	Corinne	Carlson	Copenhagen
Beginner	Dietmar	Davis	Dublin
Beginner	Ellen	Evans	Essen
Beginner	Ida	Ingelberg	Ingolstadt
Beginner	John	Janssen	Johannesburg
Beginner	Linda	Lee	London
Beginner	Petra	Paulsson	Phoenix
Beginner	Quincy	Quarles	Quebec City
Beginner	Victor	Viking	Venice
Experienced	Beata	Berghill	Barcelona
Experienced	Fred	Fisher	Frankfurt
Experienced	Gregory	Green	Gaza City
Experienced	Henry	Hansson	Hamburg
Experienced	Mike	Miller	Morristown
Experienced	Nathali	Nguyen	New York
Experienced	Oscar	Oliveiro	Oslo
Experienced	Richard	Richardson	Riga
Experienced	Sandra	Stewart	San Diego
Experienced	Tim	Turner	Tahoma
Questionable	Uwe	Ufford	Ulm
Questionable or Experienced	Karl	Karlsson	Kyoto or Kansas City

Task: A new football club should be created by merging two existing sports clubs:

- The tables of the two clubs are arranged differently and use different naming schemes (e.g. qualification levels)
- Some people are members in both clubs and need to be resolved properly.
- Highlight possible inconsistencies (red text color)

Program Example #1
Merging Two Tables

8 statements: load, clean, align semantics, merge, and save

Simple statements
easy to understand

Multiple words for functions, variables, table names, header names, allow for readability and naming flexibility

Excel file loaded with
a single simple statement

No loops. Simple statements
apply for whole tables

**One statement merges
two tables** as specified

Full Excel support

```
table load excel file              ( football club, Football Membership List.xlsx );
table load                         ( soccer club,   Soccer Membership List.csv ); // Beginners = Novices
table rename column headers        ( football club, { Family Name, City }, { Last Name, Town } );

table process selected rows        ( soccer club,   [Level]==Novice, [Level]=Beginner );


table merge                        ( football club,
                                     soccer club,   {Last Name,First Name},{Level,Town},append," or " );
table sort rows                    ( soccer club,   { Level, Last Name,  First Name });
table rearrange columns            ( soccer club,   { Level, First Name, Last Name, Town } );

table save excel file              ( soccer club, Soccer Club, New Soccer Club Membership List.xlsx );
echo ("New soccer club has ", table length( soccer club ), " members. Enjoy playing.");
```

Program Example #1
Merging Two Tables

7 additional statements add coloring, formatting, and style to Excel file

Simple statements
easy to understand

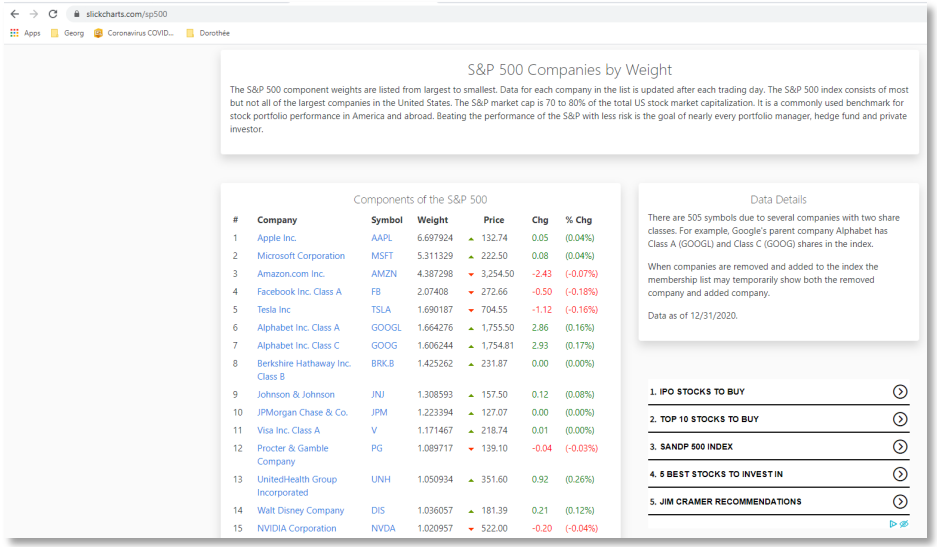
Simple and target inde-pendent approach to format
tables, rows, columns and cells

Full Excel support

Highly comprehensible function names

```
table style table                    ( soccer club, sheet, column width, 20, row height, 20,
                                                freeze rows, 1, autofilter, 0, vertical align, center );
table style rows                     ( soccer club, 0, sheet, boldface, true, fill color, gray 15 );
table style columns                  ( soccer club, Level, sheet, column width, 30 );

table process selected rows          ( soccer club, ([Level] = '*Questionable*'),
      table style cells                       ( soccer club, Level, row(), single, text color, red ) );

translate style attributes for excel( soccer club );
table save excel file                ( soccer club, Soccer Club, New Soccer Club Membership List.xlsx );

echo ("New soccer club has ", table length( soccer club ), " members. Enjoy playing.");
```

Program Example #2
Combining Stock Data: SP 500 and NASDAQ 100



#	Company	Symbol	Price	Chg	% Chg	Weight sp500	Weight nasdaq10
1	3M Company	MMM	175	0.21	0.12%	0.318459	
2	A. O. Smith Corporation	AOS	54.82	0	0.00%	0.023472	
3	ABIOMED Inc.	ABMD	326.6	2.4	0.74%	0.046276	
4	AES Corporation	AES	23.7	0.2	0.85%	0.049417	
5	AMETEK Inc.	AME	120.94	0	0.00%	0.087884	
6	ANSYS Inc.	ANSS	363.8	0	0.00%	0.098729	0.258
7	ASML Holding NV	ASML	486.99	-0.73	-0.15%		0.305
8	AT&T Inc.	T	28.76	0	0.00%	0.647313	
9	AbbVie Inc.	ABBV	107.15	0	0.00%	0.597494	
10	Abbott Laboratories	ABT	109.49	0	0.00%	0.612924	
11	Accenture Plc Class A	ACN	261.21	0	0.00%	0.522713	
12	Activision Blizzard Inc.	ATVI	92.85	0	0.00%	0.226706	0.593
13	Adobe Inc.	ADBE	500.1	-0.02	0.00%	0.757774	1.984
14	Advance Auto Parts Inc.	AAP	157.51	0	0.00%	0.033788	
15	Advanced Micro Devices Inc.	AMD	91.66	-0.05	-0.05%	0.348385	0.912
16	Aflac Incorporated	AFL	44.47	0	0.00%	0.091757	
17	Agilent Technologies Inc.	A	118.49	0	0.00%	0.114563	
18	Air Products and Chemicals Inc.	APD	273.22	0	0.00%	0.190791	

Task: Import the SP 500 and NASDAQ 100 listings and merge them

- Data source1: https://www.slickcharts.com/nasdaq100
- Data source2: https://www.slickcharts.com/sp500
- Some companies are listed in only one of them, others are listed in both.
- Combine the information, show the weighting in the two listings and color the stock price developments

Program Example #2
Combining Stock Data: SP 500 and NASDAQ 100

13 Statements, 1 loop and 1 variable do the complete job

Download 2 files
and align header names

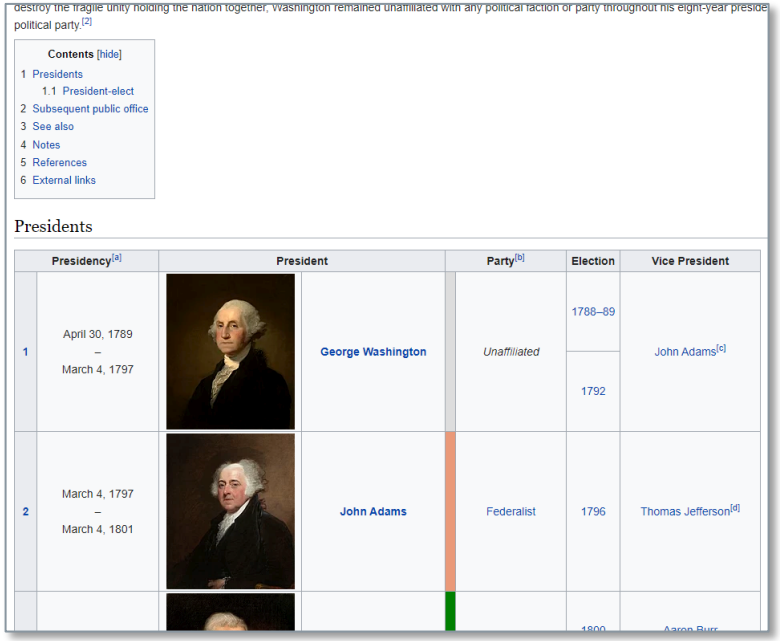
Merge the 2 tables
and assign common name

Clean Up

Save as Excel File

```
//    Download two web pages.
      for all parameters ( {nasdaq100, sp500} , listing[] )
      {
            file download overwrite       ( "https://www.slickcharts.com/" + listing[], listing[] + .html);
            table load                    ( listing[] ,listing[] + .html,  HTML, "Components of the" );
            table rename column headers   ( listing[], "Weight", "Weight " + listing[] );
                                          // Weight info is specific to SP and Nasdaq, so add the listing name
            table save                    ( listing[], listing[] + " as loaded.csv" );
      }

//    Combine the two tables
      table merge extend columns    ( nasdaq100, sp500, Symbol );
      table rename                  ( sp500, stocks );

//    Cleanup
      table correct headers         ( stocks, '*Price*', Price );
      table rearrange columns       ( stocks, { '#', Company, Symbol, Price, Chg, '% Chg' } ); // Weightings follow afterwards

//    Percent value gets converted to regular number, price value is cleaned up
      table process           ( stocks, ['% Chg'] = smart numeral( middle( ['% Chg'], '(', ')' )); [Price] = clean numeral([Price]) );
      table sort rows         ( stocks, Company );
      table process           ( stocks, ['#'] = row() );

//    Save the work
      table save excel file                   ( stocks, "NASDAQ and SP500", Stocks.xlsx );
```

Program Example #3
Web Data: Organizing all Presidents in Wikipedia



destroy the fragile unity holding the nation together. Washington remained unaffiliated with any political faction or party throughout his eight-year preside political party.[2]

Contents [hide]

1 Presidents
1.1 President-elect
2 Subsequent public office
3 See also
4 Notes
5 References
6 External links

Presidents

	Presidency[a]	President	Party[b]	Election	Vice President
1	April 30, 1789 – March 4, 1797	George Washington	*Unaffiliated*	1788–89 1792	John Adams[c]
2	March 4, 1797 – March 4, 1801	John Adams	Federalist	1796	Thomas Jefferson[d]

Presidenc	Presidency (1)	President	Party	Election	Vice President
1	April 30, 1789 – March 4, 1797	George Washington	Unaffiliated	1788–89, 1792	John Adams,
2	March 4, 1797 – March 4, 1801	John Adams	Federalist	1796	Thomas Jefferson
3	March 4, 1801 – March 4, 1809	Thomas Jefferson	Democratic-Republican	1800, 1804	Aaron Burr, George Clinton
4	March 4, 1809 – March 4, 1817	James Madison	Democratic-Republican	1808, 1812	, Vacant after Apr. 20, 1812, Elbridge Gerry, Vacant after Nov. 23, 1814
5	March 4, 1817 – March 4, 1825	James Monroe	Democratic-Republican	1816, 1820	Daniel D. Tompkins,
6	March 4, 1825 – March 4, 1829	John Quincy Adams	Democratic-Republican	1824,	John C. Calhoun,
7	March 4, 1829 – March 4, 1837	Andrew Jackson	Democratic	1828, 1832	, Vacant after Dec. 28, 1832, Martin Van Buren
8	March 4, 1837 – March 4, 1841	Martin Van Buren	Democratic	1836	Richard Mentor Johnson
9	March 4, 1841 – April 4, 1841	William Henry Harrison	Whig	1840	John Tyler
10	April 4, 1841 – March 4, 1845	John Tyler	Whig		Vacant throughout presidency,
11	March 4, 1845 – March 4, 1849	James K. Polk	Democratic	1844	George M. Dallas

Task: Download the list of Presidents and generate Excel table with one president per row

- Data source: https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States
- Some Presidents had multiple terms
- Ignore the portraits
- Some vice presidents had deviating terms
- Remove redundant artefacts, e.g. cross-referencing symbols
- Generate a nice table with **parties colored differently**

Program Example #3
Web Data: Organizing all Presidents in Wikipedia

16 Statements, 0 Loops and 0 Variables to organize all the Presidents

Internet Download
in 1 statement

Cleanups
with minimum effort

Align the data.
Contents done!

Define colors
affiliated to the parties

Color and Style the results

Save as Excel File

```
include ( Style Library ); // This library needs to be included if you want to add style and formatting to the table

      file download overwrite       ( "https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States", presidents.html );
      table load                    ( presidents, presidents.html, HTML, 'id="Presidents"' );

      // Clean up and strip out all footnote references and new lines in the fields
      table delete rows             ( presidents, table length( presidents ) -1 );  // Remove last row with redundant footnote info
      table process all cells       ( presidents, [.] = replace all( literal([.]), { "[a]" .. "[z]", new line}, '' ) );

      // Remove the blank column originally containing portraits and put president name into all rows

      table delete columns          ( presidents, {President, Party } );
      table rename column headers   ( presidents, {"President (1)","Party (1)"}, {President, Party} );
      table fill vertically         ( presidents, President );
      table consolidate             ( presidents, President, { Election, Vice President }, append, ", " );

      table initialize ( party colors,    // Define party colors
            {{ Party Name,                Colors },
             { Democratic,                "#8080FF" }, // Light blue
             { Republican,                "#FF8080" }, // Light red
             { Federalist,                coral },
             { "Democratic- Republican",  excel light green },
             { "Democratic-Republican",   excel light green },
             { Whig,                      yellow },
             { National Union,            ocre },
             { Unaffiliated,              gray 15 } } );

      // Add some colors and styles
      table process                 ( presidents, table style cells( presidents, Party, row(), single,
                                                  fill color, [ party colors : Party Name, [Party], Colors ] ) );
      table style columns           ( presidents, { "Presidency (1)", "President", "Vice President" }, sheet, column width, 30 );
      table style columns           ( presidents, { Party, Election }, sheet, column width, 20, horizontal align, middle );
      table style rows              ( presidents, 0, table, boldface, true );
      table style table             ( presidents, sheet, wrap text, true, autofilter, 0, freeze rows, 1 );


      translate style attributes for excel     ( presidents );
      table save excel file                    ( presidents, All U.S. Presidents, presidents.xlsx );
```

B4P

Beyond Former Performance.

Information *www.b4p.app*

Contact

Europe
Georg zur Bonsen
+41 56 221 82 00
zur-bonsen (at) bluewin .ch

North America
Rafael M Richards
+1 202 469 15 27
rmrich5 (at) gmail .com

Presentation: Updated 2021-01-07