

B4P

Beyond Former Performance.

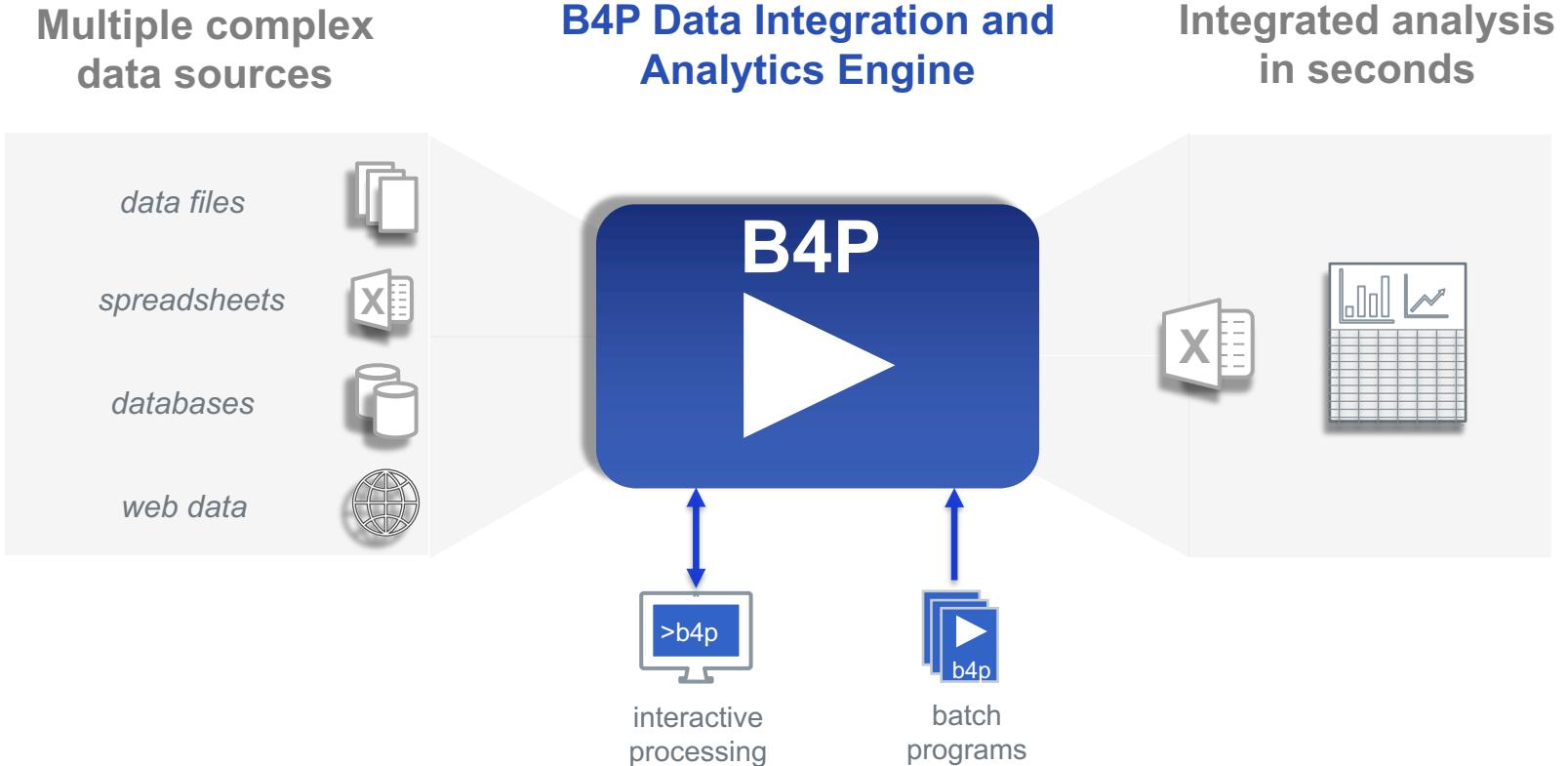
*A powerful programming language and analytics engine
enabling rapid transformation of big data into powerful insights*

Transforming Big Data into Powerful Insights



B4P Data Integration and Analytics Engine

Overview



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

1

Problem Statement

2

B4P Data Integration and Analytics Engine

3

B4P Language

4

B4P Program Examples

5

B4P Real-world Use Cases

Problem Statement

Current methods of analytics automation are complex, expensive, and opaque

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.

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

Write a Computer Program

(C, Java, Python, SAS 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.

Resulting code cannot be understood, shared, managed, nor adapted by business users.

Hire a Consultant

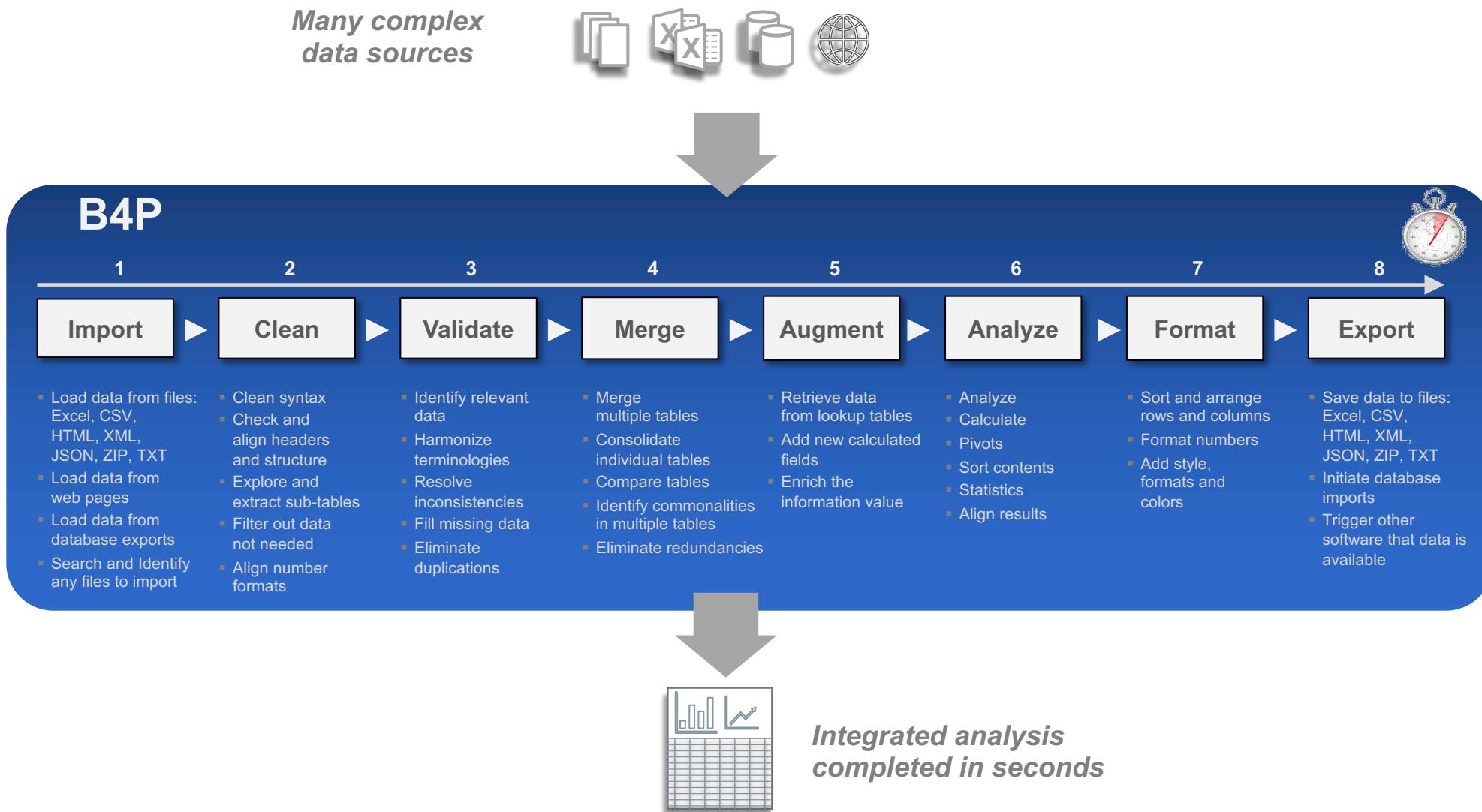
(or two)

- 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 end up depending on them, and you need to repeatedly convince your boss that the updates are worth the money.

Expensive, external vendor dependency, no long-term sustainability

B4P Low-Code Integration and Analytics Engine

Solution Overview

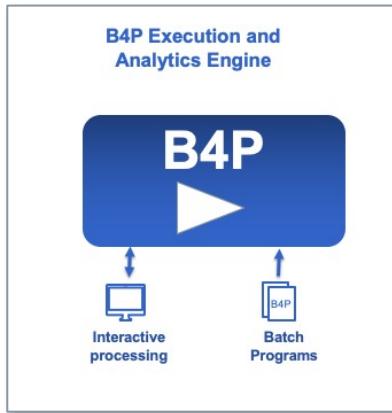


B4P Low-Code Integration and Analytics Engine

Main Components

The B4P Engine

The B4P Engine is designed for extreme performance managing Big Data – processing tens of millions of rows of data in seconds on commodity personal computers.



- **Extremely Fast:** Compiled and runs 'on the metal' to the peak performance of the very latest 8-core and 12-core processors from Intel and Apple (M1).
- **Extremely Light weight** (< 3 MB installation footprint).
- **Extremely Secure.** No connection to any 'cloud service'. Runs 100% on standalone personal computer fully isolated within the corporate network, thus allowing safe use on highly confidential corporate and financial information.
- **Extreme Reliability.** No Dependencies. Once installed there is no means for software to 'break' as it has no dependencies.
- **Many data formats supported** (Excel, HTML, XML, JSON, text files, etc., full UNICODE)

The B4P Language

The B4P Language is a Low-Code, Domain-specific Language designed specifically for tabular data, and has over 800 functions built in.

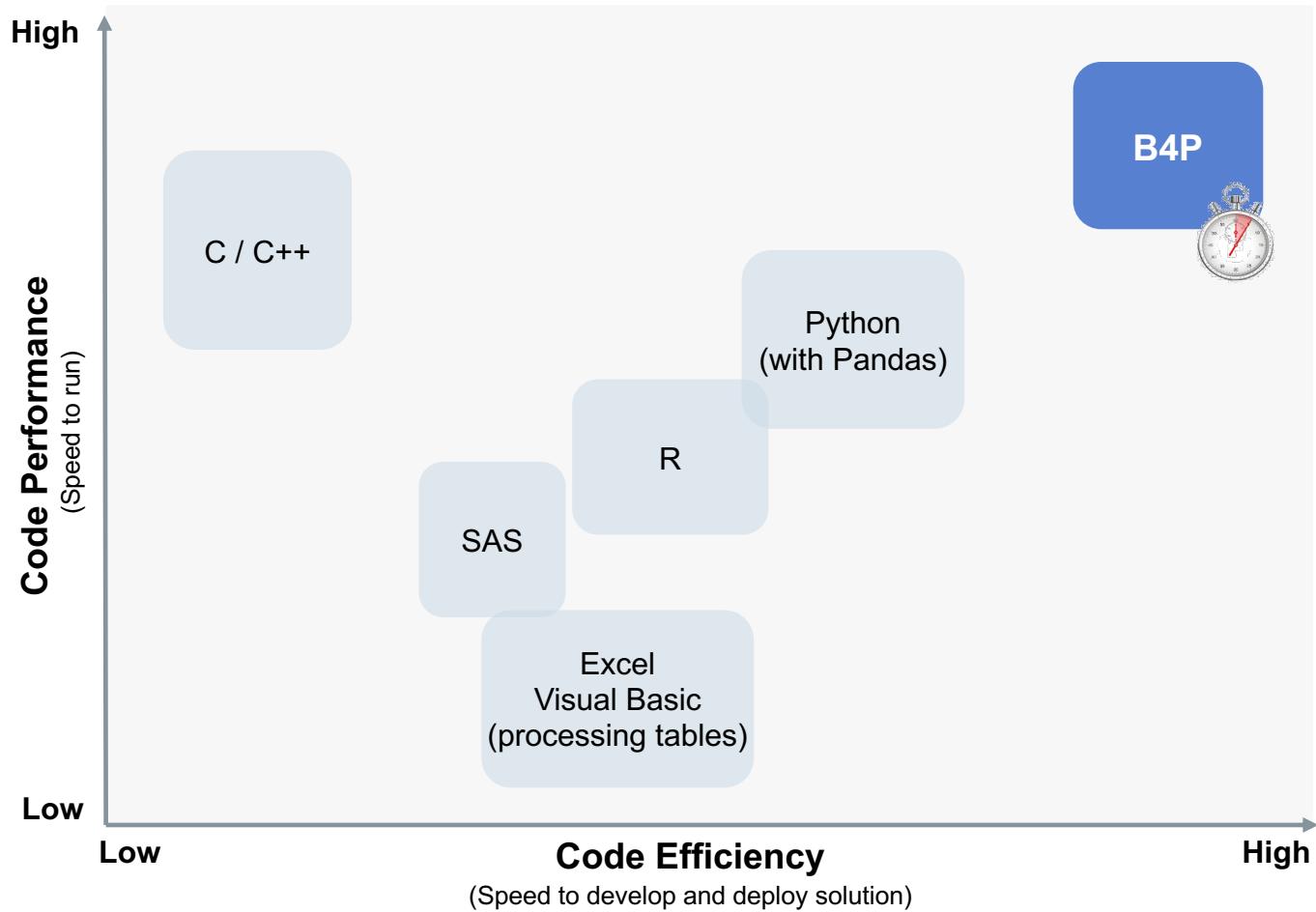
```
table load excel file      ( football club, Football Membership List.xlsx );
table load                  ( soccer club, Soccer Membership List.csv );
table rename column headers( football club, { Family Name, City }, { Last Name, Town } );
table process selected rows( soccer club, [Level]==Novice,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

Principle of Low-Code Approach: Few statements suffice

- **Simple syntax:** Easy to read, learn, understand and run
- **Extensive library:** Over 800 powerful functions built-in, with easy extensibility for new functions.
- **Compact methods** for powerful processing steps eliminates need for complex code, loops, or other administrative overhead.

B4P Language

Provides Highest Code Performance and Efficiency



B4P is highest in both Code Performance and Code Efficiency, providing complete solutions with less than 10-20 lines of code

The B4P Language is a Low-Code, Domain-specific Language for tabular data, and has over 800 functions built in.

High performance Big Data Processing

- Code Performance: Processing very large tables and variables is B4P's core functionality
- Code Efficiency: Single statements replace need for 10-50 lines of code in other languages

Low Code: Delivers solution with minimal coding

- Maximum functionality with fewest lines of code

Flexible Variable Structure

- Assign variable and table names as you wish, including spaces and special characters, directly and indirectly
- Dynamically build up variable structures and reshape them when needed.

Optimized for Simplicity of Coding

- Function library and semantics allow for flexible and powerful operations without loops and variables
- Example: **table process (...)**

Programs are Portable across all Platforms

- B4P programs are fully portable, sharable, and executable across all operating systems (Windows, Linux, MacOS) and all computer architecture (Intel x32, x64; ARM M1), assuring maximum re-use across the enterprise.

B4P Language

Key Features

Simplicity

Easy to read und understand

- Close to natural language.
- Clear syntax easy to read.
- Compact and powerful semantics.

Big Tables are the DNA

- Language semantics are built on processing tables easily.
- No external libraries needed.

Low Code - Very Compact

- Achieve the most with few lines of code in step-by-step approach.
- No hassle with declaring variables, memory management, etc.

High Power Density

- High coding density translates into maximum machine performance.

Performance

Rich Function Library

- More than 800 functions available and growing.
- Many functions process very large tables, sets, matrices and other structures.
- Broad spectrum of other general purpose and file system functions.
- Advanced flow control functions going far beyond *for*, *while* and *if*.

Code in Function Parameters

- B4P supports code passed as parameters to functions which are then executed when needed.
- Makes operations possible without using variables and loops.

Colorful & Formatted Output

- B4P supports a rich library to create Excel and HTML files with rich formatting, autofilters, etc.

Flexibility

Freedom of Naming

- Full naming flexibility (all characters, incl. space) for variables, tables and headers.
- Create variable names from other data, e.g. table header names.
- Multiple word function names

Flexible Variable Structure

- Create and work with simple variables, parameter sets, structures and arrays.
- Supports complex variable tree structures in a simple way.

Libraries

- Import libraries or create your own libraries to optimize your programming efficiency.

Portability

Cross Platform Portability

- B4P programs run under Windows, LINUX and MacOS (incl. M1) without adaptations.
- No need to change path names to run on other OS environments.

File and Data Formats

- Supports CSV, HTML, Excel (with formats), JSON, etc. transparently.

UNICODE

- B4P is fully UNICODE compatible, and accepts all UTF character formats on top of legacy formats.

Standard I/O

- GUI intentionally not supported to preserve cross-platform portability.
- Same console I/O feature set across platforms, incl. text colors.
- Embed B4P in batch programs

The B4P Language allows one solve complex problems with clear, simple, minimal code
Focus on the *what*, not the *how*.

B4P Examples

B4P Example #1

Merging Two Soccer Clubs

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



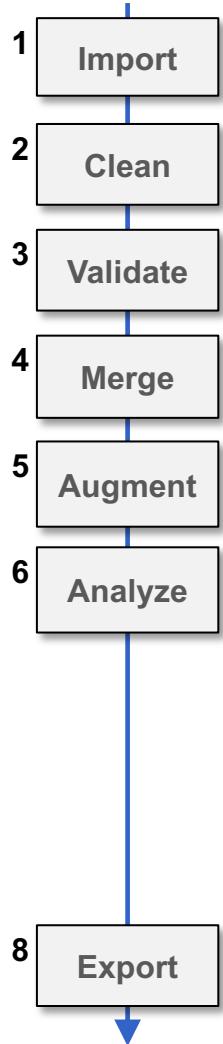
Task: A new football club should be created by merging two existing sports clubs:

- 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)

B4P Example #1

Merging Two Soccer Clubs

Solution: 8 Statements.



```
table load excel file      ( football club, Football Membership List.xlsx );
table load                  ( soccer club,   Soccer Membership List.csv );

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.xlsx );
```

Simple statements

Easy to read multi-word names

- Functions, variables, tables, etc.

Rich and flexible function library

- No or just small number of loops and variables needed for coding

Full Excel support

- Loading and saving
- Full data transparency

Portability ensured

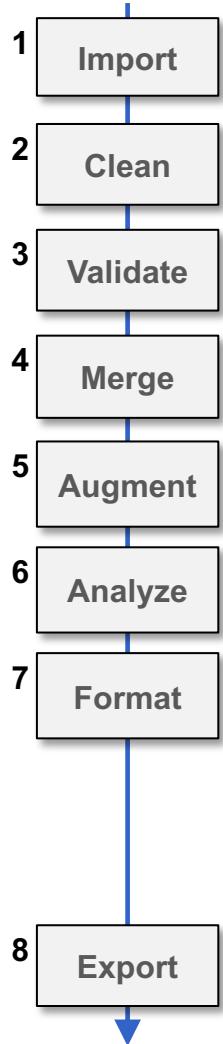
- Statements are independent from platform and output format

B4P Example #1

Merging Two Soccer Clubs

Solution: 8 Statements.

Formatting: 5 Statements.



```
table load excel file      ( football club, Football Membership List.xlsx );
table load                  ( soccer club,   Soccer Membership List.csv );

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 style table              ( soccer club, sheet, freeze rows, 1, autofilter, 0 );
table style auto width         ( soccer club );
table style rows               ( soccer club, 0, sheet, boldface, true, fill color, gray 15 );

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

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

Highly powerful formatting functions

- Small number of statements suffice

B4P Example #2

Combine Online Stock Data: SP 500 and NASDAQ 100

The screenshot shows two separate web browser windows. The top window displays the 'Components of the S&P 500' with columns for #, Company, Symbol, Weight, Price, Chg, and % Chg. The bottom window displays the 'Nasdaq 100 Companies' with similar columns. Both lists include company names, symbols, current price, change, and percentage change.



#	Company	Symbol	Price	Chg	% Chg	Weight sp500	Weight nasdaq100
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

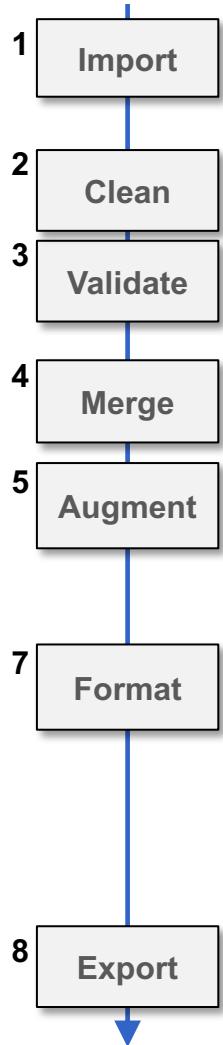
- Data source1: <https://www.slickcharts.com/nasdaq100>
- Data source2: <https://www.slickcharts.com/sp500>
- Some companies are listed in only one of them, others are listed in both.
- Combine the information, show the weighting in the two listings and color the stock price developments

B4P Example #2

Combining Stock Data: SP 500 and NASDAQ 100

Solution: 12 Statements.

Formatting: 6 Statements.



```
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 clean
        ( listing[], trim spaces );
    table process
        ( listing[], [% Chg']=smart numeral( middle( [% Chg'], '(,' ) )); [Price]=clean numeral([Price]) );
    table rename column headers ( listing[], "Weight", "Weight " + listing[] ); // Weights are specific to Nasdaq and S&P
}

table merge extend columns ( nasdaq100, sp500, Symbol );
table rename
    ( sp500, stocks );

table sort rows
    ( stocks, Company );
table process
    ( stocks, ['#'] = row() ); // Number the items

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

table style auto width
table style theme
table process
    table style cells
        ( stocks );
    table style columns
        ( stocks, Zebra Vertical Lines, pattern, 2, table, "gridlines, false" );
    table style table
        ( stocks, // Negative numbers: red; positive numbers: navy blue
        ( stocks, { 'Chg', '% Chg' }, { 2: row() }, single, text color, select if ( [Chg]>0, navy, red ) ) );
        ( stocks, '% Chg', sheet, number format, "0.00%" ); // Value to show as percent.
        ( stocks, sheet, freeze rows, 1, autofilter, 0);

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

B4P Example #3

Analyzing all Presidents in Wikipedia

destroy the fragile unity holding the nation together, Washington remained unaligned with any political faction or party throughout his eight-year presidency.

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]



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

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

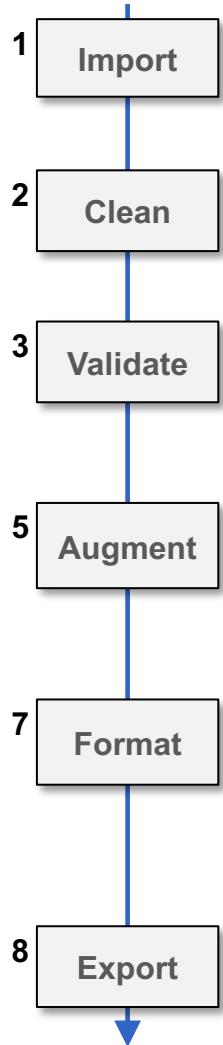
- Data source: https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States
- Some Presidents had multiple terms
- Ignore the portraits
- Some vice presidents had deviating terms
- Remove redundant artefacts, e.g. cross-referencing symbols
- Generate a nice table with **parties colored differently**

B4P Example #3

Analyzing all Presidents in Wikipedia

Solution: 9 Statements.

Party-specific coloring: 6 Statements.



```
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"' );

// Strip all footnote references and new lines in the fields, and the last table row with footnotes inside
table delete rows          ( presidents, table length( presidents ) -1 );
table process all cells     ( presidents, [.] = replace all( literal([.]), { '[?]', new line}, '' ) );

// Remove the blank column originally containing portraits and put president name into all rows
table delete columns        ( presidents, {Portrait, Party} );
table rename column headers  ( presidents, {"Presidency (1)", "Party (1)"}, {Period, Party} );
table fill vertically       ( presidents, President );
table consolidate           ( presidents, President, { Election, Vice President }, append, ", " );

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

// 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 );

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

B4P

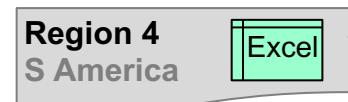
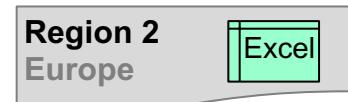
Real-World

Use Cases

B4P Real-world Use Case #1

Integrating Corporate data from branch offices worldwide

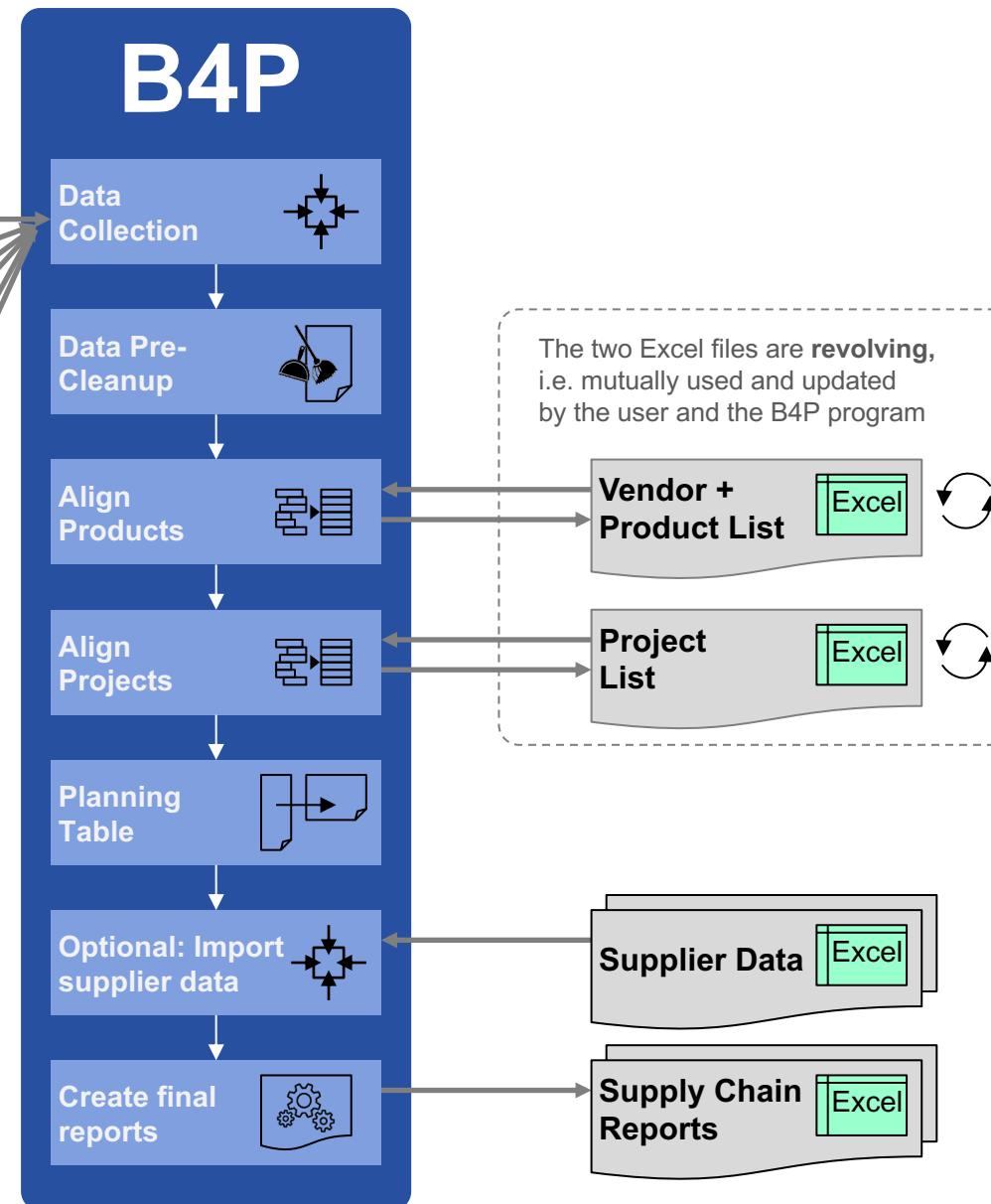
Each region manages their own data in different database systems or manually with Excel



●
●
●



More than 20 different files!



1. Load Data from all Sources

- The data from different sites originate from 20 different database exports or manually prepared Excel files

2. Clean-Up and Harmonize

- Harmonize data formats to week numbers and years

3. Align Product Information

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

4. Align Project Information

- 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

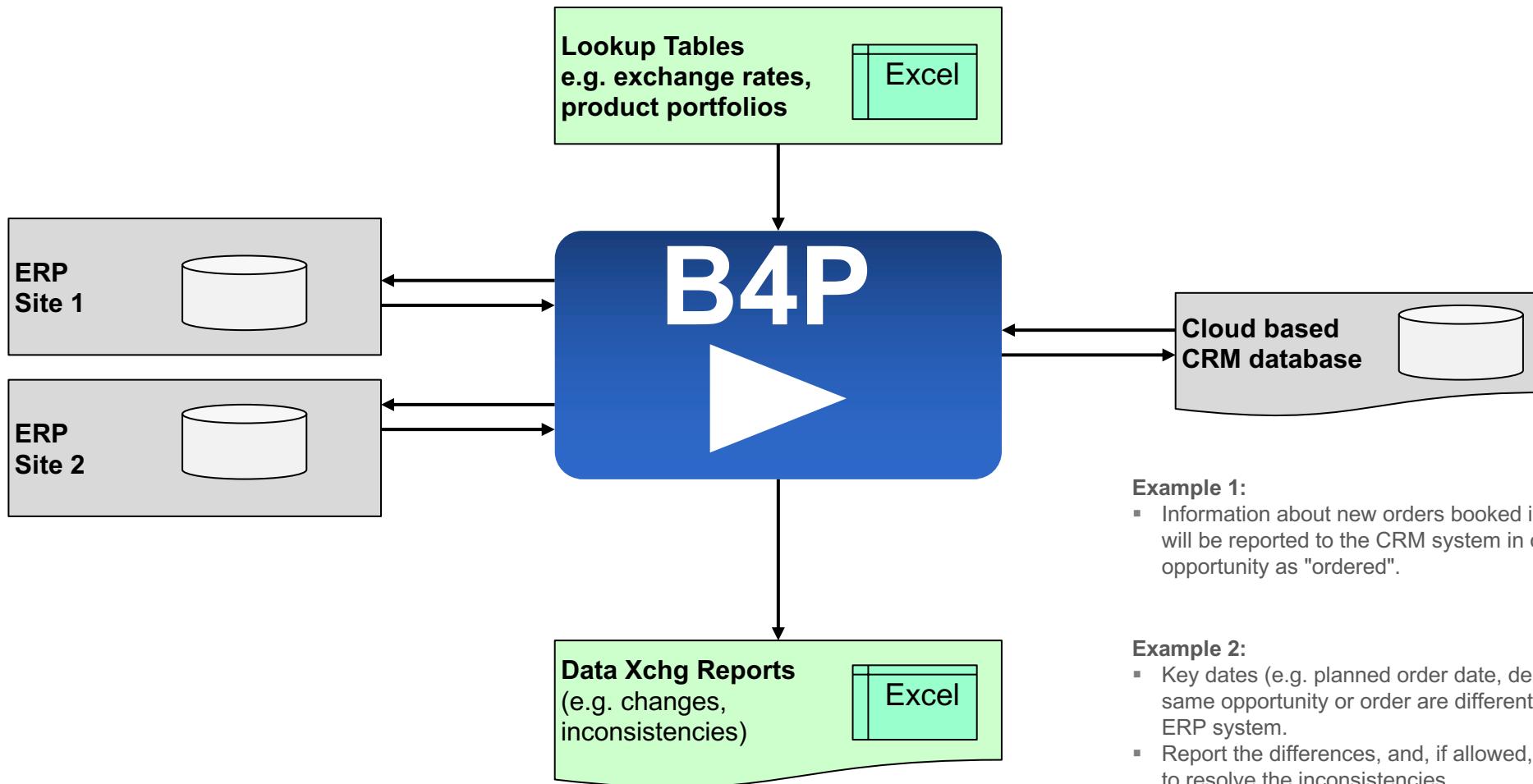
- If supply info is available, then match demand with their delivery plans

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:

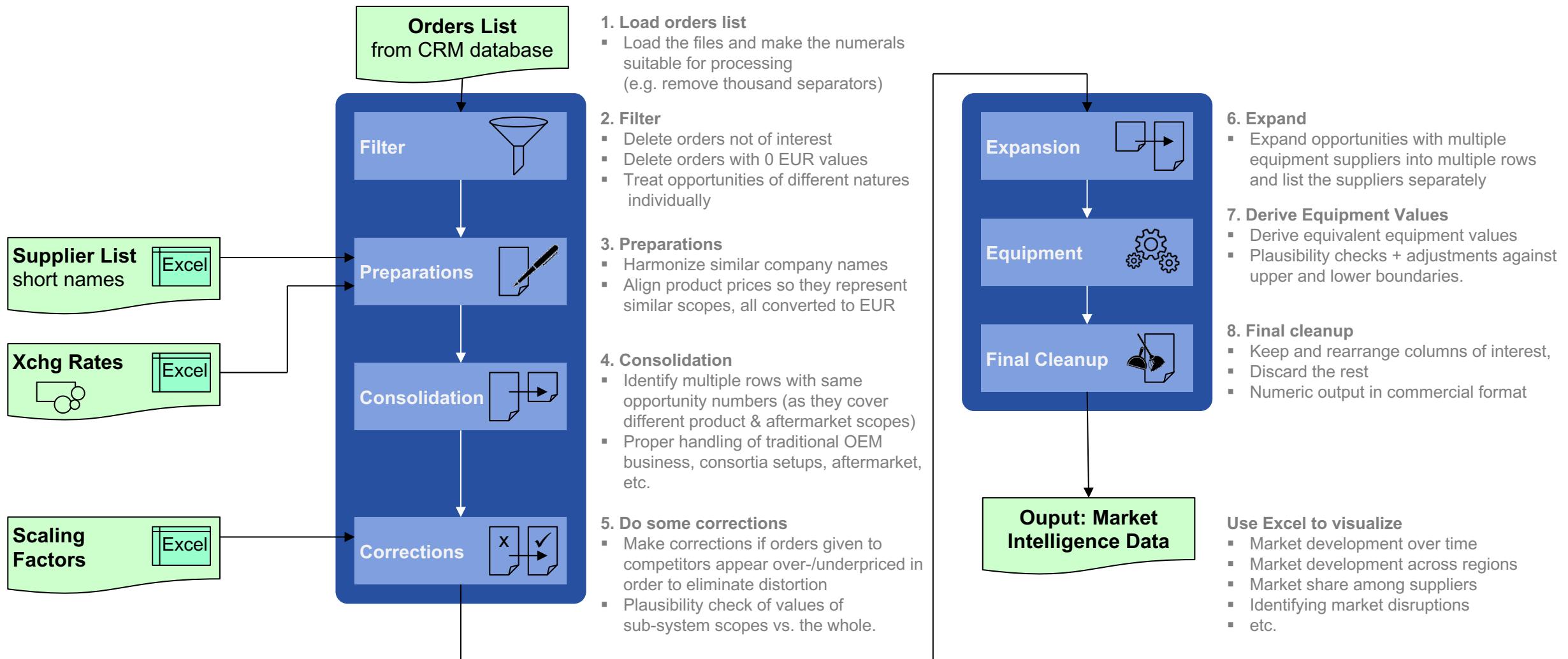
- Information about new orders booked into the ERP system will be reported to the CRM system in order to mark the opportunity as "ordered".

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



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