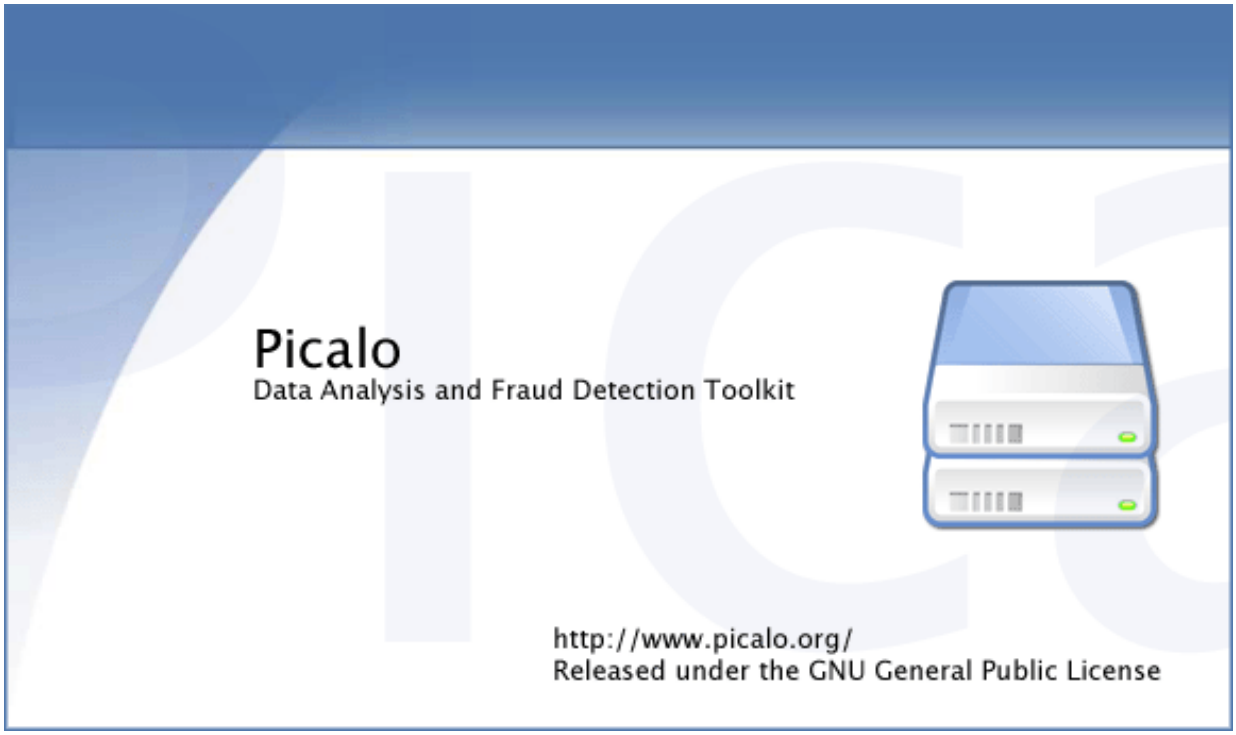


Picalo Manual

Version 3

Conan C. Albrecht

March 31, 2010

Contents

1 Introduction **5**
- 1.1 A Short FAQ . . . 6
  - 1.1.1 Why Picalo? . . . 6
  - 1.1.2 What is the Picalo philosophy? . . . 7
  - 1.1.3 What is Picalo's relationship to ACL and IDEA? . . . 7
  - 1.1.4 Why not use MS Excel? . . . 7
  - 1.1.5 How does Picalo compare with EnCase or The Forensic Toolkit? . . . 8
  - 1.1.6 What is Picalo's relationship to Numerical Python/Numarray/SciPy? 8
  - 1.1.7 What is Picalo's relationship to Python? . . . 8
  - 1.1.8 Why Python (instead of VB, Perl, Java, .Net)? . . . 9
  - 1.1.9 Why open source? . . . 9
  - 1.1.10 Where does the name Picalo come from? . . . 10
  - 1.1.11 What are Picalo's Limitations? . . . 10
- 1.2 Acknowledgments . . . 11
- 1.3 License . . . 11

2 A Tour of Picalo **12**
- 2.1 The Picalo Workspace . . . 12
  - 2.1.1 Object Tree . . . 12
  - 2.1.2 Table/Script Tabs . . . 12
  - 2.1.3 Shell, Command Log, and Script Output . . . 12
- 2.2 Working With Picalo Commands . . . 14
  - 2.2.1 Menu and Toolbar . . . 14
- 2.3 Expressions . . . 17

3 Detectlets **18**
- 3.1 Example Detectlets . . . 19
- 3.2 Detectlet Installation . . . 19

4 Best Practices Usage **20**

5 Picalo Preferences **22**
- 5.1 Chapter Learning Objectives . . . 22
- 5.2 Access General Preferences . . . 22

5.3 Disable Splash Screen . . . 23
5.4 Automatically Check for Updates . . . 24
5.5 Change Preferred Language . . . 25
5.6 Add commands to run at startup . . . 26
5.7 Access the Script Editor Preferences . . . 26
5.8 Change Font Preferences . . . 27
5.9 Tab Preferences . . . 28
5.10 Method Preferences . . . 29

6 Introduction to Picalo Tables **30**
6.1 Learning Objectives . . . 30
6.2 Picalo Tables . . . 30
6.3 Loading and Saving Picalo Files . . . 32
6.4 Sorting Tables . . . 33
6.5 Editing and Adding Records . . . 35
6.6 Copying and Pasting Table Data . . . 36
6.7 Importing Data . . . 36
6.8 Closing Tables . . . 36
6.9 Rows and Columns: What are they? . . . 37
6.10 Inserting and Deleting Rows . . . 38
6.11 Insert and Delete Fields (*or Columns*) . . . 40
6.12 Saving Updates and Changes to Table Properties . . . 43

7 Changing Table Properties **46**
7.1 Learning Objectives . . . 46
7.2 Table Properties . . . 46
7.3 Adding Columns . . . 48
7.4 Removing Columns . . . 49
7.5 Data Types . . . 50
7.6 Selecting Precision . . . 51
7.7 Selecting Format . . . 52

8 Filtering **54**
8.1 Learning Objectives . . . 54
8.2 Why is filtering important? . . . 54
8.3 Creating a Filter . . . 55
8.3.1 Right Click . . . 56
8.3.2 Filter Box . . . 56
8.3.3 Select By Value . . . 57
8.3.4 Select By Expression . . . 57
8.4 Simple Expressions . . . 58
8.5 Complex Expressions . . . 59

9 Sorting Picalo Tables **60**
9.1 Learning Objectives 60
9.2 Ascending vs. Descending 60
9.3 Sort Using a Right Click 60
9.4 Sort Using the File Menu 61

10 Database Access **64**
10.1 Connecting to MySQL 64
10.2 Connecting to MySQL via ODBC 68
10.3 Querying a Database for Information 71

11 Joining Two Tables Together **74**
11.1 Learning Objectives 74
11.2 Joining Tables 74
11.3 Joining Tables Using Fuzzy Match 76
11.4 Analyzing Resulting Tables 78

About This Manual

The purpose of this manual is to introduce new users to Picalo. It begins by explaining the Picalo philosophy, continues with a best practices model, and then exhibits tutorials for different Picalo functions. This manual is the first stop for new users. So if you are new to Picalo, *congratulations – you are in the right place!*

This manual is only the beginning of your journey in Picalo. This program can do many things that are beyond the scope of this introductory manual – as you use Picalo, you'll discover its many abilities. Because Picalo is based in the standalone Python language, it inherits many capabilities, such as scraping of web pages, parsing of email and log files, and extensibility.

Once you feel comfortable with this manual and with Picalo basics, read through the Advanced Picalo manual for additional capabilities.

A Moving Target

This manual is current as of Summer, 2006, or Picalo 3.0. Since Picalo is a work in progress, it is constantly improving and changing. Therefore, the dialogs in the program may be slightly different that what you see in this manual due to change in the program. We'll try to keep the manual updated with the program, but we simply do not have enough time to update the manual in perfect unison with the program. Since the Picalo program itself is the main priority, the manual may be slightly behind the actual features or user interface of the program.

Acknowledgement

I'd like to thank my Information Systems 413 course students for their help in the tutorials contained in this manual. Many of the tutorials were written exclusively by them, and others are combinations of my own writing with text or graphics from various students.

Chapter 1

Introduction

Welcome to the world of Picalo, a collaborative, open-source effort to produce a data analysis application suitable to auditors, fraud examiners, data miners, and other data analysts! Picalo specializes in data retrieved from corporate databases, such as employee records and sales records, but it can be used generally for many different types of analyses.

The foremost goal of Picalo is to create a worldwide repository of fraud detection and analysis routines called *Detectlets*. Technical users of Picalo are under a (friendly) moral obligation to codify their analyses as Detectlets and submit them to the central web site. You can choose whether to provide your Detectlets for cost or for free. Appendix **??** details how to write detectlets. Certainly, learn the Picalo framework first. But once you are comfortable with it, contribute to the effort.

If you have even limited skills in programming, please consider submitting a few Detectlets. Together, we'll create a set of routines that cover the entire analysis field: from fraud detection to data mining to statistical analyses. We'll create a repository of routines that is unrivaled by any single company or organization. This is the spirit of open source that I hope you'll find invigorating and satisfying.

First, a little background on Picalo. For years, I used many different applications for analysis – Excel, Paradox for DOS, Visual Basic, Microsoft Access, ACL, IDEA, PostgreSQL, Numeric Python, SAS, and many others. Each of these applications had strengths, but none was specifically written for analysis of corporate data or for serious data mining. Some were generic databases; others were audit-oriented or statistical applications. When I went searching for an analysis application, I had the following requirements:

- Cross platform so analyses could run on Windows and Mac OS X laptops as well as corporate Unix servers.
- Routines specific to the detection of fraud and corruption.
- Support for routine automation in a robust, powerful (yet easy to use) language.
- Ability to analyze huge amounts of data.
- A way for technical analysts to support non-technical analysts without the typical "have the techie do all the hard work" result.

- Both a graphical user interface and bare-bones, console interface.

Finding no product that suited my needs, I developed Picalo, an open architecture that all can contribute to and use. Picalo is conceptually split into three levels of routines: Level 1) foundation routines that support basic analysis steps (but are not specific to corruption or fraud); Level 2) fraud routines that combine level 1 routines in intelligent ways so non-technical users can perform powerful analyses; and Level 3) automatic discovery of fraud and corruption using expert system rules applied to Level 2 routines.

Picalo is currently focused on analysis for fraud and corruption detection. However, it is an open framework that could actually be used for many different types of data analysis: network logs, scientific data, any type of database-oriented data, and data mining.

1.1 A Short FAQ

This short list of "frequently asked questions" should answer questions you may have about Picalo, its purpose, and its relationship to other software packages.

1.1.1 Why Picalo?

Why not use 'your favorite app here' to do analyses? Why use Picalo instead? Picalo may or may not be the right choice for what you are doing, depending upon your goals. The following are features of Picalo that seem somewhat unique:

- Picalo is an open framework. Users can either use the built-in routines or write their own. Those who write their own can share their routines with others in the Picalo community. The goal is to create a large set of analysis routines that meet many different needs–on a scale that a single company could never do.

- Picalo's Detectlet framework allows those with scripting interest and ability to create wizard-based routines for others in their organizations. This helps bridge the gap between power users and others.

- The philosophy of Picalo is to help users learn how to script. Data analysts who know basic scripting routines (for loops, for example), are more efficient and effective than those who do not. Picalo shows you the script code for everything you do in the GUI, and it includes a function composer to help you create function calls.

- Picalo includes advanced analysis routines not found in competing products. For example, it supports grouping by a number of days for analysis of labor and time card data. Picalo can also automatically group records to achieve a specified degree of smoothness in data.

- Picalo's scripting is based in Python, a powerful and easy-to-learn computer language. Rather than creating its own language (like competing packages do), Picalo rises on the shoulders of an extremely well-done language. You can download any of thousands of Python libraries from the Internet to use in your analyses.

- Picalo runs on Windows, Mac OS X, Linux, and many other systems. Most competing data analysis applications run only on Windows.

1.1.2 What is the Picalo philosophy?

The Picalo community believes that data analysis is best done through scripting. Everything in Picalo is designed to help you learn basic scripting techniques so you can become more productive. Picalo certainly includes a powerful graphical user interface, but the focus of the toolkit is to help you write powerful, 10- to 20-line scripts.

Picalo is based in open source principles. This doesn't mean the designers can't make money with Picalo, it just means that the software code is open for others to fix bugs, code review, and improve upon. Profits should be made in *using* the software (on the job or in consulting practice) rather than in *selling* the software.

1.1.3 What is Picalo's relationship to ACL and IDEA?

Picalo is a competitor. ACL and IDEA are two of the most popular data analysis applications used in corporate data analysis. Each has unique features and abilities. The scripting ability of each is great. However, both applications are primarily *audit* applications rather than general data analysis applications. Picalo contains routines that can be used in many fields, including auditing, fraud detection, and other areas.

Picalo's Detectlet features differentiate it from ACL and IDEA. The end-goal of Picalo is to create a worldwide repository of Detectlet's that thousands of different anlayses using a wizard interface.

One of the primary differences between these applications and Picalo is the latter is open source. Users can help solve bugs, contribute new modules, and do analyses not possible in closed-source software.

Finally, Picalo is built upon Python, a full-featured, mature programming language. While ACL and IDEA define their own languages, Python has existed for over 10 years. Modules to do all sorts of things have been contributed by programmers around the world. For example, the regular expression module provides powerful text searching not found in many off-the-shelf products – since it is part of the core Python language, you can be sure it is well tested, efficient, and mature. Python is not only simple, but it is also extremely powerful.

1.1.4 Why not use MS Excel?

Microsoft Excel (or, insert your favorite spreadsheet here) has become a powerful, mature application for number analysis. Spreadsheets are widely known and used, and they are visual in their analysis. However, spreadsheets are best suited for ad-hoc analysis rather than formal database-oriented analysis. For example, Excel is an excellent choice for tracking your investments or calculating a home mortgage schedule. It is less suitable for querying, stratifying, summarizing, joining, matching, trending–routines Picalo specializes in.

Picalo is meant to work with data retrieved from large databases; Excel is meant to work primarily with small sets of numbers in free-form. While Excel can only handle about 65,000 records, Picalo can handle millions upon millions of records (limited only by available memory in your computer).

A simple example illustrates this purpose[1]. Most home owners don't own expensive woodworking equipment, such as saws, routers, and so forth. Instead, they use smaller, nonindustrial tools for their weekend jobs like fixing their fence. However, professional carpenters *do* own expensive equipment. They are willing to invest time and money in industrial tools to do their daily work with. Excel is a general purpose tool meant for the weekend analyst. Picalo is meant for the professional. It's startup cost is higher, but it provides economies of scale not possible with ad-hoc programs like Excel.

1.1.5 How does Picalo compare with EnCase or The Forensic Toolkit?

EnCase and TFK are really different products with different purposes. They are useful to inspect a computer, gather data, and document evidence gathering. Picalo is a data analysis package. It assumes you've already gathered the data into a data source and need to combine it in different ways to generate useful information.

1.1.6 What is Picalo's relationship to Numerical Python/Numarray/SciPy?

The two are different, with different goals and different communities. Numerical Python is geared toward the scientific community whereas Picalo is geared towards the corporate data community. NumPy specializes in array storage and representation, and its functions focus on array manipulation, math, and so forth. Most NumPy routines assume you are working with large matrices of numerical data. The matrices do not normally contain empty cells, and the functions are focused on scientific applications.

Picalo specializes in database connectivity, representation of data in tables, and data analysis routines. While both arrays and tables are similar, the focus of the two projects (and resulting data structures and functions) are quite different. Picalo works with database-type data, such as text, addresses, salaries, and so forth.

1.1.7 What is Picalo's relationship to Python?

Picalo is built in Python the same way a book is built in English or some other language. More technically, Picalo is a set of modules, functions, and routines built on top of Python. A casual reviewer of Picalo may not see the power of extending the Python language. Python alone is a very powerful data and text analysis platform – Picalo adds the Table type and many data analysis functions to the core Python language.

[1]This example comes was first given in the *sed and awk* book published by O'Reilly.

Because Picalo is built in a professional language, anything you can do in Python you can do in Picalo (this turns out to be a lot!). There are thousands upon thousands of Python routines available for free download from the Internet.

1.1.8 Why Python (instead of VB, Perl, Java, .Net)?

Because it's better, of course! :)

1.1.9 Why open source?

This question actually has several answers. The surface level answer is because competing data analysis applications charge thousands of dollars per year for software that is not terribly difficult to write.

Another answer is significant money can be made *using* the software (on the job or as a consultant) rather than *selling* the software. Instead of paying a software tax each year, why not create a platform that does exactly what the community needs? Thousands of programmers from around the world can contribute thousands of data analysis routines that all can use. There's plenty of money to be made in the *use* of the software in our work. Why not collaborate on our tools?

Consider why businesses create software: to make money. They do things because there is a *business case*. Most business decisions are made based upon the return (i.e. money) they will bring. In a perfect world, software decisions should be made because of *technical* reasons, not business reasons. Normally, business and technical motivations are in line: good software usually sells well. But this is not always the case. Too many software packages have been ruined over the years because companies gave in to the almighty dollar/euro/etc. Software bloat, unstable software, and market–driven release dates are well know problems in the software development world.

Open source is different. Decisions are made for their technical merits rather than for their monetary return. For example, most open source projects release new product versions when the products are tested and ready for use. Most projects don't even give a release date – they simply say, "it will be released when it's ready". Contrast this with commercial software packages. Release schedules are usually driven by marketing reasons (when the competition will be releasing, when the market is ready, etc.). The result is often programmers are pushed by marketers and executives to push products out before they are tested and ready.

Please note that this philosophy is not an argument against free market economies. Competition is a good thing. Choice is a good thing. It is simply an argument for competition in the use of the software (consulting, etc.) rather than competition in the tool development. Creating software is different than the creation of most products, for example constructing automobiles. The former can be done with almost no investment up front, has almost no distribution costs, and benefits from collaborative development. The latter requires considerable investment to produce even one product and has significant distribution costs. Software is a different type of product and should be treated as such.

Another example is the feature creep seen in many commercial products. In order to continue making money, companies *must* release new versions every two years or so. Once a product matures, companies almost have to invent needs their new features solve! Consider Microsoft Word – basic word processing hasn't changed much since the beginning of GUI applications. However, Microsoft continues to upgrade Word with features like the infamous "clippy" to give their users reasons to upgrade, even though most users still only need the basic features. If a company declared their product "completed", their money stream would quickly dry up.

Finally, Picalo is open source because it draws upon many different products, such as Python, mx.DateTime, the Gadfly database, and so forth. It is one more contribution to an ever-growing selection of excellent, free software.

1.1.10 Where does the name Picalo come from?

(Personal note from Conan Albrecht, who named Picalo) I do a lot of programming at home. On the day I started Picalo, my two oldest girls (six and four at the time) were dancing around my home office singing the jingle, "Daddy, Daddy, there you go, let us see you piccolo (i.e. dance)!" At this point in the song, I had to get up and dance for the next minute or so. I thought it was cute, so I searched the Internet for variations on the spelling of the word. I was happy to find that Picalo (and its variants) didn't have any negative connotations and were not widely used. I finally settled on the "picalo" spelling. The term has no relation to the piccolo musical instrument or other variations and uses of the word.

1.1.11 What are Picalo's Limitations?

Picalo is relatively new software. It has several limitations, a few of which are listed here:

- Picalo may still have bugs, especially in the GUI, that have not been found by the users. You should always double check your analyses, print control totals, and list intermediate results to ensure that your analyses run the way you expect. (Although you should do this regardless of the software package.)

- Picalo has no warranty, support, or guarantee. It is released with the hopes that it will be useful, but you are responsible for anything it does to your computer, your data, or anything else. Although there is no formal support, open source projects like Picalo have shown that user support for one another is usually superior than company support.

- Unless you write scripts to use database results intelligently, Picalo loads all data into memory for analysis. Your computer's memory may limit the number of records you can analyze. In practical use, Picalo has been shown to analyze millions of records easily, so this limit is quite high.

1.2 Acknowledgments

Picalo is built upon the shoulders of many great projects. Thousands of individuals have contributed time and energy to these projects, and the Picalo effort is grateful for their work. These are listed as follows:

- Python: `http://www.python.org`
- wxPython: `http://www.wxpython.org`
- mxDateTime: `http://www.egenix.com`
- pyODBC: `http://pyodbc.sourceforge.net`
- psycopg2: `http://initd.org/projects/psycopg2`
- MySQLdb: `http://mysqldb.sourceforge.net`
- Statistics package pstat.py by Gary Strangman
- Nuvola Icon Set by David Vignoni: `http.icon-king.com`
- Matplotlib: `http://matplotlib.sourceforge.net`
- pyExcelerator by Roman V. Kiseliov

1.3 License

Most of Picalo is released under the GNU General Public License (GPL). The GPL is a restrictive, open source license. I've released this package under the GPL to protect it. You can download the source code comes from the Picalo webiste (`http://www.picalo.org`); you are encouraged to improve and add to the application.

Detectlet libraries can be released under licenses other than the GPL. Since Detectlets will be built by companies, organizations, and individuals, it is up to the developers to decide whether to sell, open source, or even public domain their routines.

The restriction is that you cannnot use any Picalo code in your own products unless those products are also released under the GPL. If you are using a closed-source license or even one of many incompatible open source licenses, you cannot use Picalo code. The license protects the code from being "stolen" by any individual or company.

This short description is obviously an overgeneralization of the license; please see the LICENSE.TXT file for more information.

Chapter 2

A Tour of Picalo

This section introduces you to the Picalo GUI. It is designed to be similar to other programs you may have used, such as Excel, ACL, or Access.

2.1 The Picalo Workspace

The workspace, shown in Figure 2.1, is separated into three primary areas: the object tree/disk explorer, the table/script tabs, and the shell. These are described in this section.

2.1.1 Object Tree

The object tree keeps track of all database connections and tables you've created in each session. Double-clicking a table in the list brings up a spreadsheet view in the tab section on the right.

Note that tables in the object tree are not automatically saved and are deleted when the program closes. This is done because many routines create hundreds or even thousands of subtables. Saving each of these to disk is not practical or desirable. Be sure to save the tables you wish to use again.

2.1.2 Table/Script Tabs

The main section of the screen is a tab browser of the tables and scripts you have open. Tables show in a spreadsheet-like view. You can use this view to add/remove/modify columns and types, edit cell values, filter data, and view data.

Scripts show in editor files and have the typical commands of cut, copy, paste, undo, and redo. See Section **??** on scripting for more information on scripts.

2.1.3 Shell, Command Log, and Script Output

The shell allows you to type commands one at a time to produce results–it's like running a script one line at a time. In addition, when you run a full script in Picalo, all of the variables created in the script are available in the shell at its completion. This allows you

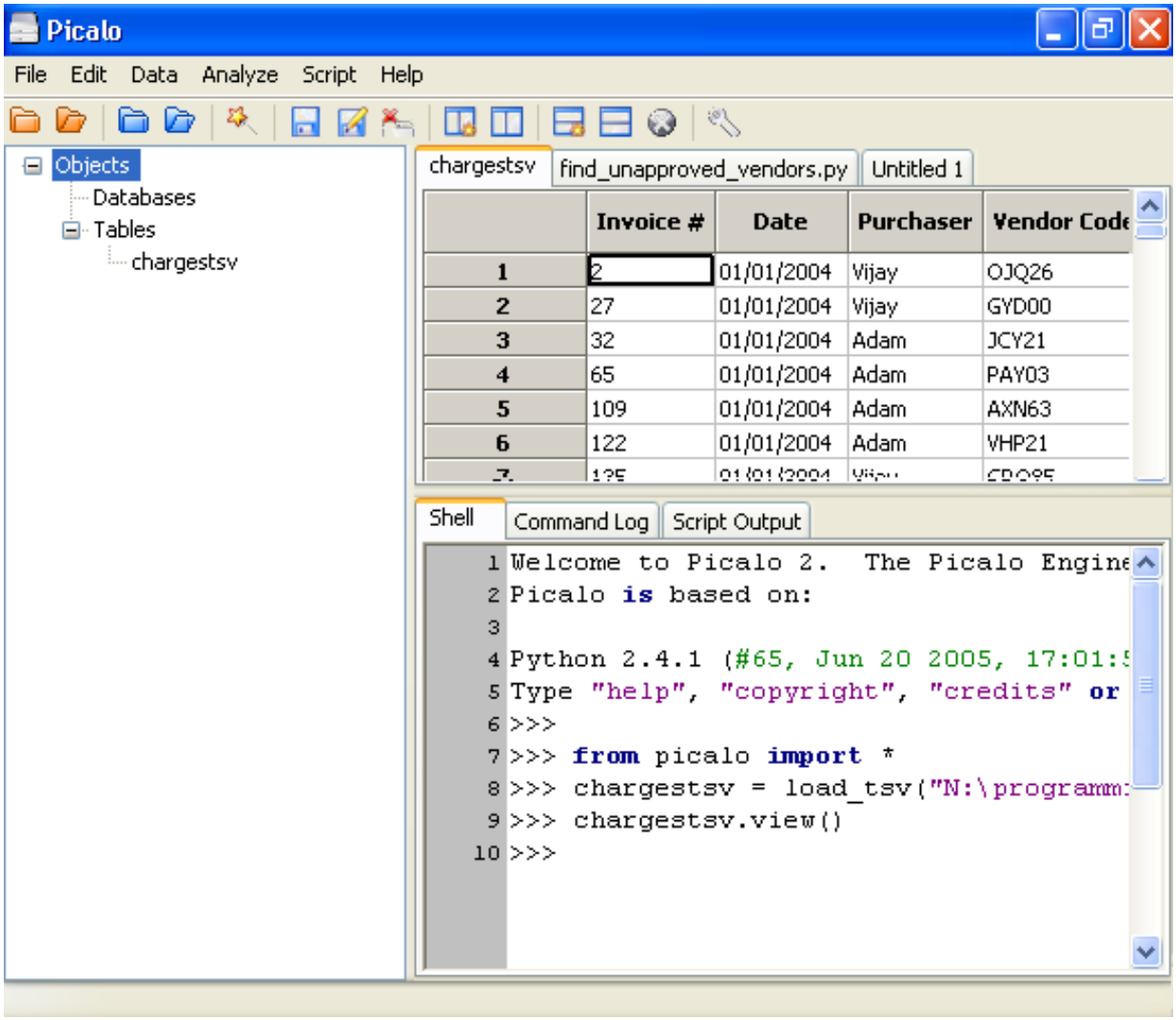


Figure 2.1: The Picalo Main Screen

to perform additional commands or inspect result data without having to rerun the entire procedure.

Everything you do with the Picalo menu and dialogs just translates to a call in the shell. As you navigate the program dialogs, watch the shell to see how Picalo is doing its work. Over time, you'll learn what each command does and become more familiar with the shell. You'll pobably find that working from the shell is faster than working using the menues.

The interface and the shell are two-way, meaning that anything you change in the interface (such as modifying a cell value in a spreadsheet view) also changes in the shell variables. In like manner, anything you change in the shell variables is immediately reflected in the interface.

The shell is an excellent place to perform ad-hoc analyses or to test commands. Its use is described in the Advanced Picalo manual.

2.2 Working With Picalo Commands

There are four ways to run Picalo commands in your analyses. These are described as follows:

- **Menu and Toolbar**: The Picalo menu (and toolbar buttons) provides access to dialogs that guide you through each function. These functions are described in Section 2.2.1 below.
- **Function Composer**: The Function Composer is a graphical way to construct Picalo command calls. It is provided for medium to advanced users.
- **Shell**: The shell, shown at the bottom of Figure 2.1, allows you to type Picalo script commands one line at a time. It is an excellent way to perform ad-hoc analyses or test command calls. The shell is described in Section 2.1.3.
- **Scripts**: Scripts are the most powerful interface to Picalo's commands. Scripts allow you to use programming constructs like for loops and if statements to fully automate different analyses.

2.2.1 Menu and Toolbar

The Picalo menu options and related toolbar buttons provide access to dialogs that guide you through various Picalo functions. Each time you select an option from the menu, Picalo shows you the related command in the shell. In fact, the menus and dialogs actually just construct Picalo commands and run them for you in the shell.

File Menu The file menu provides manipulation routines for 1) Tables and 2) Databases. Rows and columns can be added to tables, table properties like column types can be modified, database connections can be created, and database queries (resulting in tables) can be run. Also, preferences/options for Picalo can be set in this menu.

Edit Menu The edit menu contains the standard functions and features of most edit menus: Copy, Cut, Paste, Find, Replace, etc...

Data Menu The data menu provides functions that manipulate the data within tables. Different methods of stratification allow you to separate a table into a list of subtables. For example, you could separate a sales transaction table into separate tables for each sales clerk.

Summarization allows you to recombine stratified tables using summary functions like sum, average, standard deviation, counts, and so forth. Summarization is useful to smooth the time axis during trending (see the example in Section **??**), calculate the total purchases per purchaser during the period, or find which vendors are being used the most. Summarization is similar to GROUP BY in structured query language.

The join options allow you to join tables together. Picalo's join features are much slower than database query joining, but they are much more powerful and flexible. For example, rather than simply joining where values match, Picalo can join tables based on fuzzy text matching and custom expressions. With custom expressions, the ways you can join tables are limited only by your creativity.

Analyze Menu The analyze menu provides functions that run specific analyses. The first set of options on this menu provides initial views into data sets. These should when you first look at a data set. Descriptives provides basic statistical descriptives on a table like counts, standard deviations, and column totals. Digital analysis uses Benford's law of numbers to evaluate whether column values are "natural" or not. See Section **??** for more information.

The Find submenu provides audit procedures of finding unordered values, duplicate values in columns, and gaps in sequence. These functions are useful in discovering duplicate invoices or payments, missing documents, or sorting problems.

The Find submenu also helps you find matching and nonmatching columns in tables. While SQL matches records easily, it only partially supports nonmatching. In addition, matching does not join tables together – it only highlights matching records. An example of nonmatching analysis is to find vendors used for purchases who were not approved vendors. Finding the rows in the purchases file that do not have values in the approved vendor file highlights these transactions (see Section **??**).

The Select submenu allows you to select certain rows from tables to analyze. It is useful to "filter" tables before analysis and to find within tables.

The Outliers submenu helps you discover outliers in record sets. While many methods of finding outliers exist, the zscore calculation is one of the simplest and best. With these options, you can add a zscore column, select outliers, and select nonoutliers.

The Trending submenu provides different methods of discovering the direction of a trend. While any human being can look at a graphed trend and know its direction, these methods allow the computer to discover direction. This is useful when you have thousands of trends to analyze and want the computer to do the preliminary analysis for you. For example, suppose all of your employees have corporate credit cards. You want to discover employees who exhibit an increasing trend in spending over a three month period. You

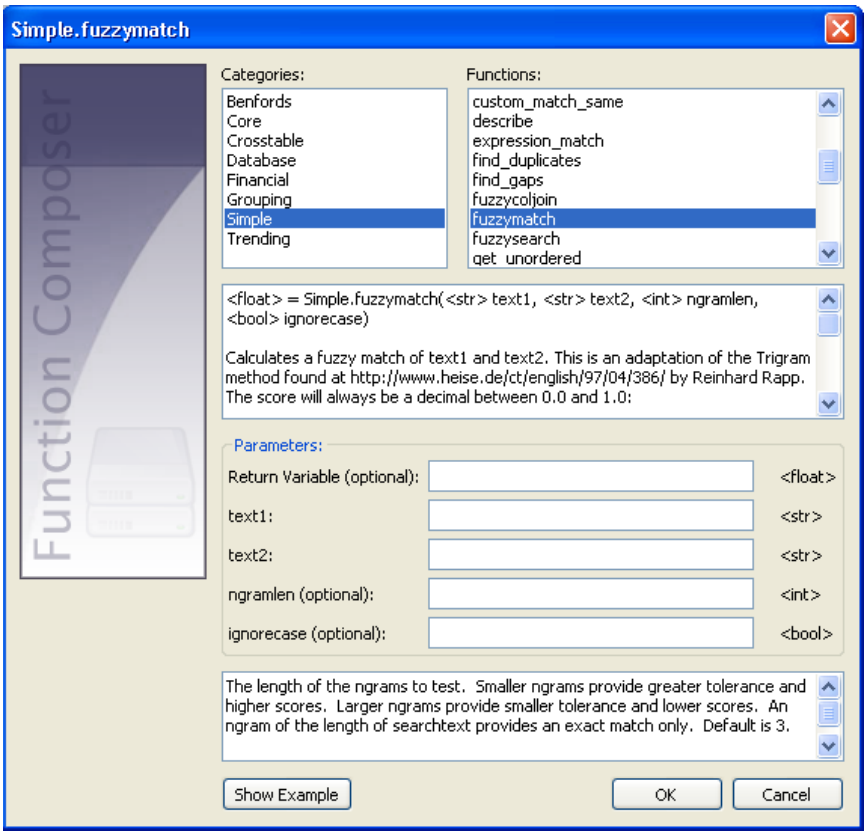


Figure 2.2: The Function Composer

first stratify the credit purchases table into subtables, one for each employee. Using one of the slope algorithms on this menu (regression, for example) and a `for` loop, you run a regression on each subtable and calculate the slope. You sort the results by slope and look at those tables with the highest slope values.

Detectlets Menu Picalo's Detectlets are one of its greatest strengths. Detectlets are wizard-driven, step-by-step guides that walk you through different analyses. Upon startup, Picalo automatically discovers the detectlet modules you've installed and makes them available on this menu.

If you want to do powerful analyses within a simple, wizard interface, Detectlets are the solution. Be sure to read Chapter 3 on Detectlets for more information.

Script Menu The script menu provides options to create, modify, and run scripts. Scripting is the subject of the Advanced User manual.

Function Composer

The function composer, shown in Figure 2.2, is like a wizard that helps you construct Picalo commands. While only some of the available functions are available from menu options,

every Picalo command can be accessed in the Function Composer. The Function Composer is a little more difficult to use than the menu options, but it is much more powerful.

The Function Composer is an intermediate step in learning the direct commands of Picalo. It gives a short description of each command, a description of each parameter you need to send, the return value, and one or two examples of its use.

2.3 Expressions

Expressions are mathematical formulas that run calculations on rows, tables, or columns. Expressions are given in Picalo as strings, such as `"id+1"`. Expressions are used to define the value of a new, calculated column or to summarize a subgroup of records in a table. They are used in most Picalo functions.

Expressions provide you with total flexibility and control in joining, selecting, stratifying, and most other analysis routines in Picalo. Since expressions can be more difficult to write, Picalo also includes shortcuts to expression writing, such as matching by value or stratifying by date.

Some variable names are assumed inside of expressions. Suppose you are selecting records from a table using an expression, Picalo needs to evaluate your expression for every record in your table. It will iteratively call your expression, starting with record 1, then record 2, and so on through the end of your table. For each record, the expression is evaluated. For example, the expression `price > 5000` will select all records with the price column value over $5,000.

Chapter 3

Detectlets

Detectlets are one of Picalo's greatest strengths. They provide a wizard-based, step-by-step guide through complex analyses. Normally, the following tenants are orthogonal to one another:

- Powerful analyses are, by definition, complex. As is stated in other areas of this manual, complex routines normally require scripting. This means that users must learn Picalo's scripting language, its available functions, and their parameters.
- Most analysts don't want to learn the complex algorithms and concepts, such as scripting. In a perfect world, we'd all learn how to script and write detailed algorithms. But the reality is that most analysts do not have the time or interest to walk the learning curve.

Detectlets solve this paradox by allowing the power users to support the regular users. In most organizations, there are at least two or three technical individuals who are willing to scale the learning curve required to create powerful and complex analyses. These individuals use the declarative Detectlet language to prepare custom wizards for their organizations. Once these modules are installed on other analysts' computers, they are widely available throughout the organization.

When a Detectlet is run, it first gathers the information required to run the analysis function. Once the information is gathered and verified, the analysis function is called. Detectlets always create a table (or list of tables). Upon successful completion, an "Interpreting the Results" window displays that helps you interpret the resulting table.

Picalo automatically detects any Detectlets installed on your computer. Be sure to look through the available Detectlets in your installation as they provide powerful analyses at the click of a button. The Detectlets menu provides access to these routines.

If you are technically skilled, please help create additional detectlet libraries! Picalo provides the foundation for a worldwide repository of detectlets for all types of domains. It is impossible for the Picalo designers to know what fraud routines work in every country, every domain, and every business. While some fraud schemes seem common to most areas, others are highly specialized to very specific medical fields, areas of the world, etc.

If you have any technical skills at all and are using Picalo in your work, you are under a (friendly) moral obligation to codify your analyses as Detectlets and submit them for

others to use. If each user submits just a few well-thought-out detectlets, we'll all benefit and be more effective in our work. Appendix **??** details the process of writing Detectlets.

3.1 Example Detectlets

The standard build of Picalo comes with a few example Detectlets (currently related to Benford's Law of numbers. These Detectlets provide examples of additional Detectlet libraries that can either be downloaded or purchased and added to Picalo. Individuals and companies that develop Detectlets are free to license their libraries as they wish (free, open source, or charge). Available libraries are listed on the main Picalo web site.

3.2 Detectlet Installation

To use additional Detectlet libraries, you must download and install them. Installation of new libraries is easy. To install new Detectlets, select Detectlet — Install from the Picalo menu and browse to the .py or .zip file. Most Detectlet libraries are provided as .zip files. The Detectlets will be immediately available from your menu – there is no need to restart Picalo.

Chapter 4

Best Practices Usage

I am often asked for the "best practices" method of using Picalo. While your needs will certainly be different than mine, I can give some indication of what has worked best for me and other Picalo users. Let me first describe my purpose for programming and for using Picalo. I use Picalo to data mine large corporate databases for fraud and irregularities. These databases are often very large and require a significant amount of processing. I do not modify the data very often – I normally see them as a read-only data source.

The best practices architecture is shown in Figure 4.1. This diagram shows Picalo being used to query corporate data sources and populate a data warehouse, where Picalo is then used for analysis. Other applications are also used to present and query data. The process is as follows:

1. The first step is to create a data warehouse.
    - You should set up a server, which is simply a computer with a database like PostgreSQL, MySQL, SQL Server, or some other production database. In particular, PostgreSQL and MySQL are free; MySQL is quite easy. However, any database works. For beginning users, you can use Microsoft Access.
    - Based upon the types of analysis you want to do, design a database that will support your needs. Learn from your database help manuals how to create databases, tables, and queries.
2. Using ODBC or Picalo's data import wizard, transfer the appropriate data from the corporate server to your data warehouse. ODBC is set up in the Windows control panel. Once you query the data into a Picalo table, you can use Picalo's Upload Table feature to upload the data into the data warehouse. If you are using Microsoft Access, you can simply use Access directly to query the data from the corporate server to your data warehouse.
3. Once the data are in your warehouse, you don't need to worry about corrupting the source data. Connect Picalo to your data warehouse using ODBC; use queries and Picalo's analysis routines to complete your analysis. While Picalo is a powerful program, each software package has unique capabilities. Other programs, such as MS Access, Excel, and Crystal Reports, can also be useful.

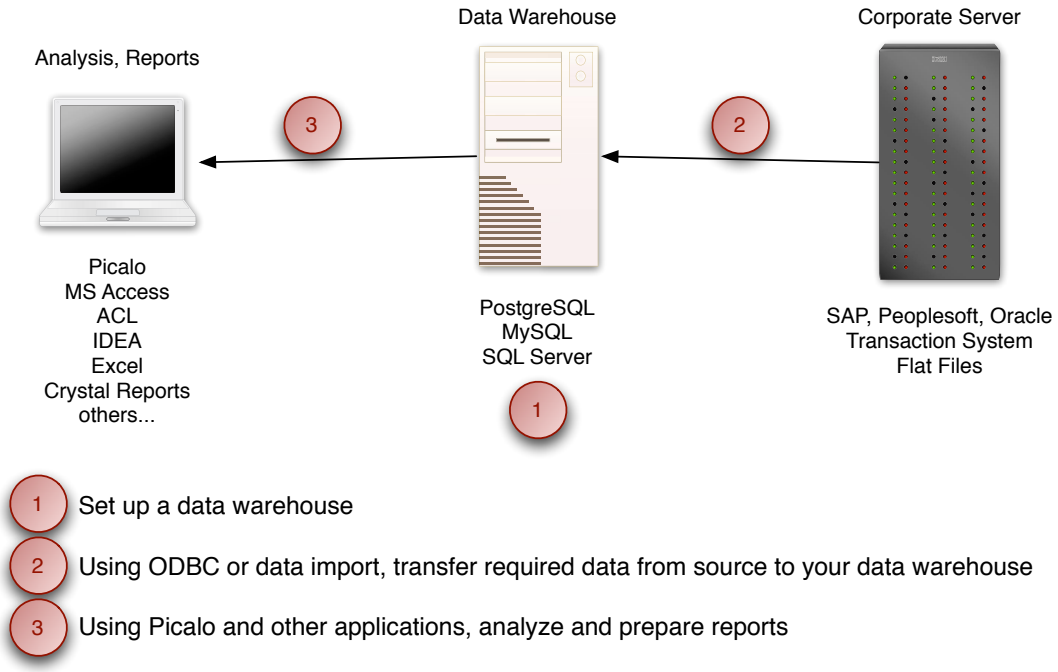


Figure 4.1: Best Practices Use of Picalo

Note from the process above that Picalo's native file format, `.pco`, is not used directly. Use Picalo's native format for ad hoc data storage and temporary holding. Normally, data warehouses using MS Access, MySQL, or PostgreSQL hold your data. Realize that Picalo's primary function is *analysis*. It is most powerful when paired with a database that specializes in data storage, quick access, and powerful queries.

Chapter 5

Picalo Preferences

In this tutorial, you'll learn how to change the preferences in Picalo. This tutorial is structured in a participatory manner – it expects that you have Picalo installed and that you're following along with your computer.

5.1 Chapter Learning Objectives

1. Access general preferences
2. Disable / Enable the splash screen at startup
3. Disable / Enable the automatic update feature
4. Change the prefered language
5. Add commands to run at startup
6. Access the Script Editor
7. Change Font Properties
8. Tab settings
9. Method Properties

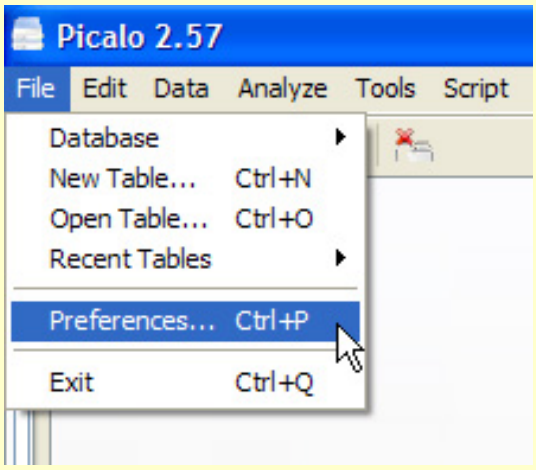
5.2 Access General Preferences

General Preferences allow you to customize Picalo to better meet your needs.

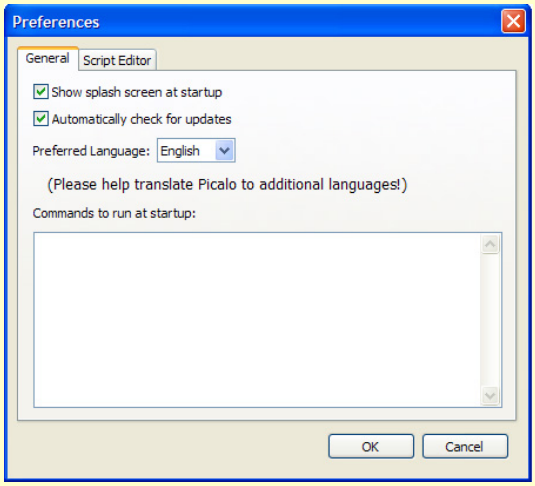
To access Picalo's general preferences:

Access Picalo Preferences

1. Select File, Preferences.



2. The Preferences" dialog comes up. Displayed are general preferences.



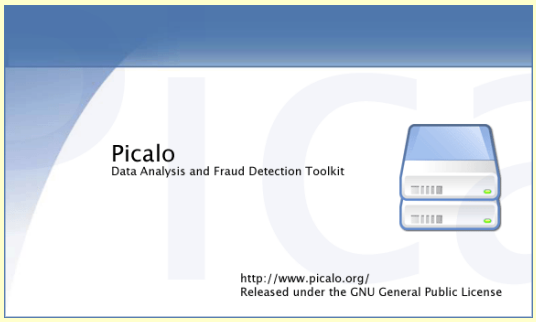
5.3 Disable Splash Screen

When Picalo starts, a splash screen is displayed while the program loads.

The splash screen looks like this:

Splash Screen

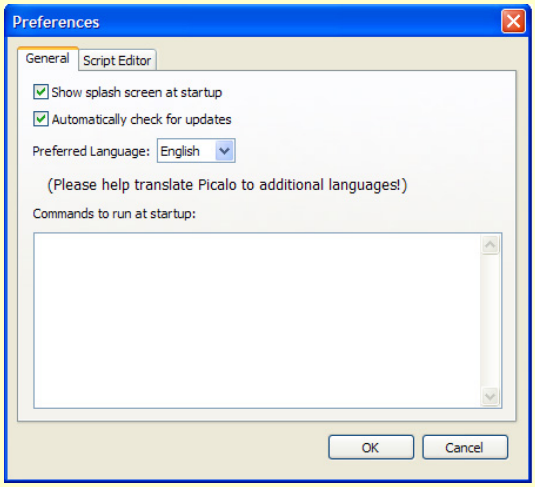
1. This image is displayed every time Picalo is started



Some users choose to disable the splash screen to speed up the loading of Picalo. To disable the splash screen:

Disable Splash Screen

1. Click the check box next to "Show splash screen at startup". When finished, click the ok button.



You can re-enable the splash screen at any time by re-checking the box.

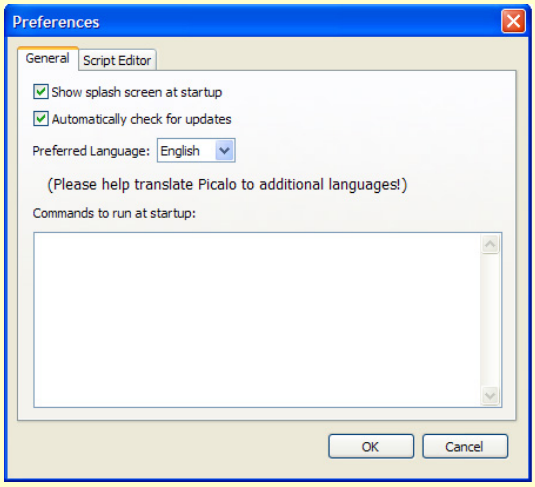
5.4 Automatically Check for Updates

When Picalo starts, the program automatically connects to the internet and checks for software updates. If updates are found, Picalo will download and install the updates to insure your software is current and working its best. You can disable Picalo from automatically checking for updates.

To disable automatic updates:

Disable Automatic Updates

1. Click the check box next to "Automatically check for updates". When finished, click the ok button.



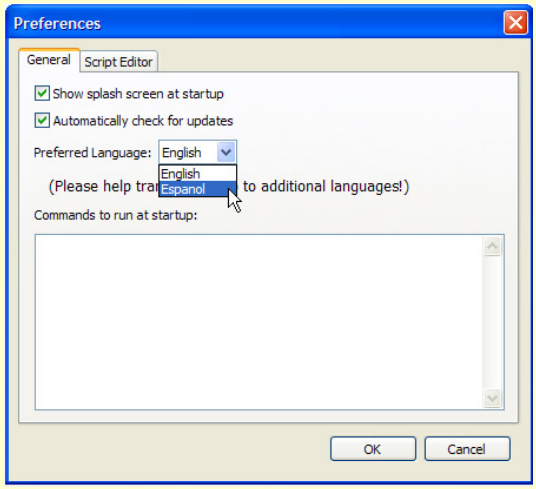
You can re-enable automatic updates at any time by re-checking the box.

5.5 Change Preferred Language

You can change the preferred language in Picalo.
To change the preferred language:

Change Language

1. Choose the desired language from the "Preferred Language" drop down box. When finished, click the ok button.



5.6 Add commands to run at startup

When Picalo starts, commands added to the "Commands to run at startup" text box will automatically run. By using this feature, you can make desired actions occur each time Picalo starts. For example, suppose you want to create a simple table each time you start Picalo. You would put this into the startup commands in preferences:

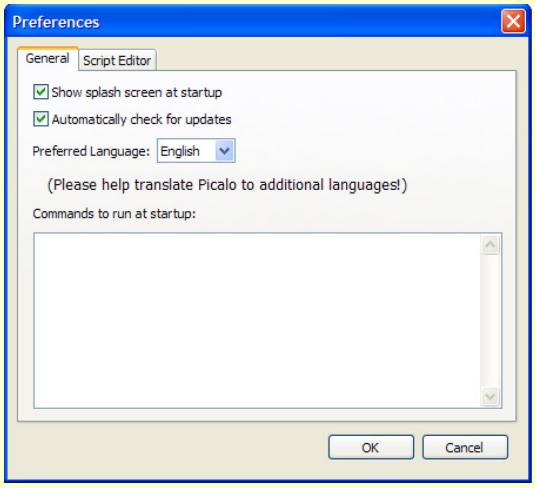
```
1 mytable = Table(['id', 'name'], [[1, 'Homer'], [2, 'Marge']])
```

Now, each time you start Picalo, you get this table automatically. You can do the same with loading tables from disk, setting up database connections automatically, etc.

To add commands to run at startup:

Add commands to run at startup

1. Click in the box entitled "Commands to run at startup". Enter the desired commands and click ok when finished.

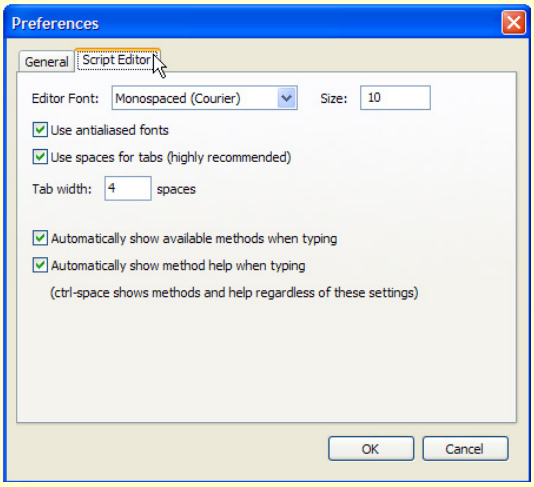


5.7 Access the Script Editor Preferences

To access the script editor preferences:

Access Script Editor

1. Select the "Script Editor" preferences tab by clicking on the tab as shown. You now have access to the script editor preferences



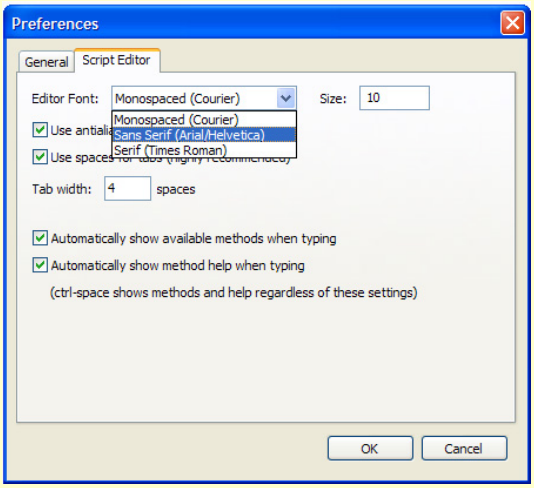
5.8 Change Font Preferences

You can edit font type and size in Picalo. You can also change the font from antialiased to non antialiased. Any changes to font type will be applied to the font displayed in the scripting window.

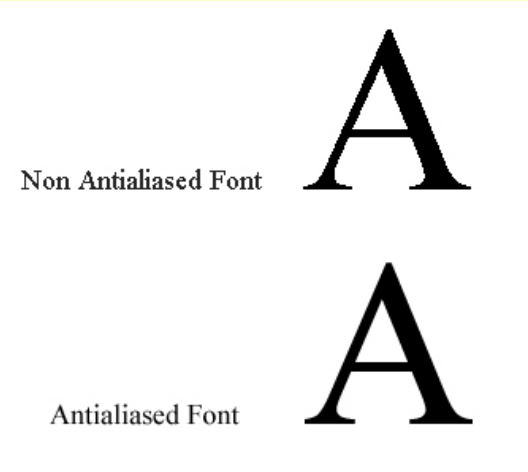
To change font preferences:

Change Font Preferences

1. Select the font type from the "Editor Font:" drop down box. Select the desired font. You can also change the font size by entering the size in the "Size" text box.



2. You can change the font from antialiased to non antialiased by clicking the check box entitled "Use antialiased fonts". Antialiased fonts are smoother and more visually pleasing as shown in the picture.



When you are finished changing the font properties, press ok. Any changes made to the font will be applied to the font displayed in the scripting window.

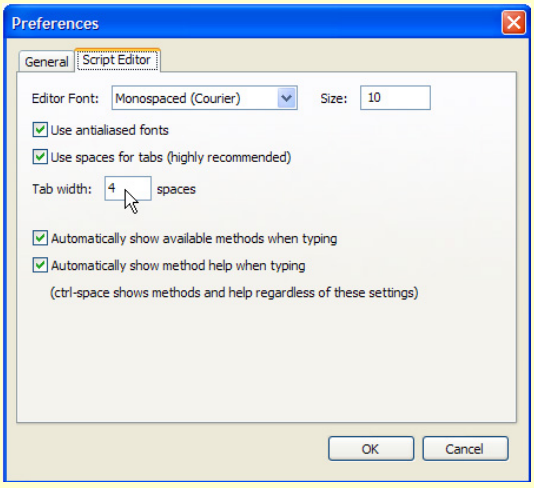
5.9 Tab Preferences

Each time the tab key is pressed, the cursor will jump a certain number of spaces. You can set this number in the "Tab Width:" text box.

To change tab properties:

Change Tab Preferences

1. Enter a number in the "Tab Width:" text box. This number will be the number of spaces the cursor moves in the script editor when the tab key is pressed.



When you are finished changing the tab preferences, press ok.

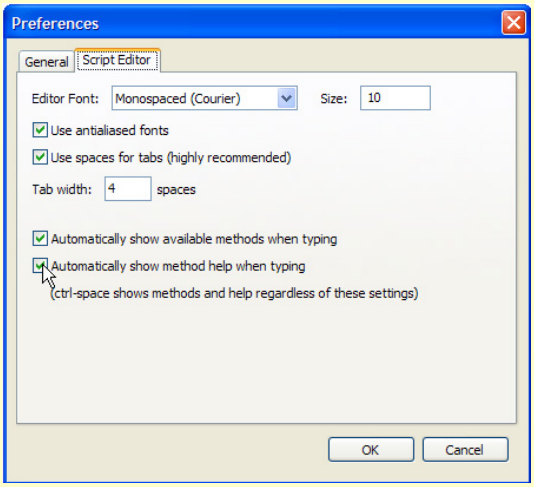
5.10 Method Preferences

When scripting in the scripting window, available methods are automatically displayed while typing. Method help is also displayed. These features are to help you script more efficiently and acurately. You can disable these features if you like.

To change scripting method preferences:

Change Method Preferences

1. To remove the automatic method and help display functionality, click the check boxes that say "Automatically show available methods when typing", and "Automatically show method help when typing." You can reactive these features by re-checking the boxes.



When you are finished changing method preferences, press ok.

Chapter 6

Introduction to Picalo Tables

This tutorial explains the primary data structure in Picalo: the table. Tables are the first thing you should try to understand as you learn Picalo because all functions and routines work on tables of data. Tables are very similar to spreadsheets in applications like MS Excel – they are two dimensional grids made up of columns and rows. In this tutorial, you'll learn how to work with Picalo tables.

6.1 Learning Objectives

1. Understand the table structure and the PCO format
2. Load and save tables from and to disk
3. Sort tables by one or more columns
4. Edit data, add rows, and remove rows
5. Close tables with varying degrees of permanence.
6. Define rows and columns and understand their differences
7. Insert and delete rows
8. Insert and delete fields (or columns)
9. Define field properties
10. Save table property changes

6.2 Picalo Tables

Picalo tables are two dimensional grids of data. The data can be of any type, and all cells in the table do not have to be the same. With a few exceptions (covered elsewhere), you

should keep your data in tables because almost all functions and routines in Picalo expect tables for input and usually produce tables for output.

In some ways, you can think of tables as spreadsheets – similar to the way you might use MS Excel. Indeed, many times you'll bring data into Picalo from Excel, Access, or similar data source because Picalo is an primary analysis application rather than a data entry application. We assume you know the basic principles of working with spreadsheets. However, Picalo tables have some important differences from spreadsheets. These are described as follows:

- Each row (also called a *record*) in a Picalo table usually represent a real world thing, such as a person, a transaction, information about a movie, or a weather reading. Each record is an individual entry. If you want to hold information about 20 movies, your table will have 20 records.

- Columns (also called *fields*) in Picalo tables are always named and usually hold the same type of data. If your table holds information about movies, the first column might be the movie title, the second column might be the release year, and so forth. The column names are not records in the table like they would be in a spreadsheet. They replace the 'A', 'B', 'C', etc. as the titles of the columns.

- Picalo tables have only as many columns and rows as you have data for. In contrast, spreadsheets always have 65,000 rows and A-IV columns. If you want to add another record to a Picalo table, you first have to append a new row to the end of it.

- Picalo tables normally hold large amounts of data with limited calculations. Spreadsheets specialize in calculations between cells, and they are often used for tasks like loan amortizations or tax schedules. In contrast, calculations in Picalo are normally done on entire columns rather than individuals cells. For example, one column might contain the birth dates of all employees in a business. In Picalo, you might add a calculated column that returns the age of each person, based on today's date and the value in their birth date column.

Each Picalo table can theoretically hold up to 2.1 billion rows and 2.1 billion columns, but most users cannot reach this theoretical limit. The primary limitation on table size in Picalo is your available memory in your computer. The realistic limit on number of rows and columns depends upon the size the data in each cell and the ratio of columns to rows[1]. Practically, you should expect to hold at least hundreds of thousands of records with a relatively modern computer.

Tables are always named in Picalo. When you load or create a table, Picalo will ask you to enter a name that it can be referenced by in dialogs, routines, and other operations. This name is usually similar to the filename the data was loaded from, but you can set it to anything.

[1]If you need to work with huge tables (in the hundreds of millions to billions of records range), you should store your records in a production database. Picalo's Database module can iterate through records individually rather than load entire tables into memory at once.

In order to enable scripting capabilities with your tables, Picalo imposes some limitations on the format of table (and column) names. Table names must start with an alphabetical character, such as A, b, or z. The remaining characters in the name can be any letter, numbers, or underscore. Tables names are case sensitive, meaning that `myTable` is different than `MyTAble`.

The following are valid table names:

- `mytable`
- `MyTable`
- `emp_records`
- `emp_records15`
- `weather12data`

The following are invalid table names:

- `7eleven` (starts with a number)
- `mydate/6` (includes a slash)
- `Dark Woods` (includes a space)

These same rules apply to column names.

6.3 Loading and Saving Picalo Files

Picalo's native format, .pco, is a cross-platform file format that saves tables to disk. This format is readable only by Picalo, but it saves all information related to the table, including column names and types, cell data, and column calculations. Files saved on Picalo for Windows can be opened in Picalo for Mac or Picalo for Linux and vice versa.

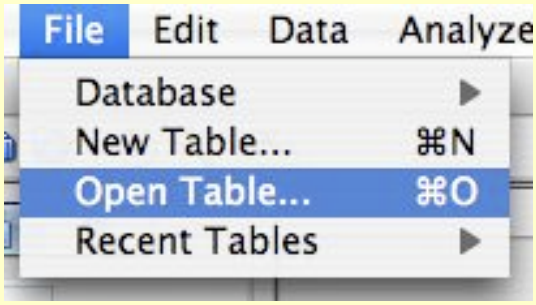
The best way to save data in Picalo is to use a production database like MS SQL Server. These databases provide features like data security, searching, and indexing that cannot be matched by any desktop product. If you do significant work with Picalo, we suggest that you invest the required time to set up a data warehouse using one of these databases (check out the open source MySQL or PostgreSQL products). There is a tutorial dedicated to databases.

So why include a native Picalo format? Because databases and data warehouses can be difficult to set up and maintain, and many times these databases are overkill for smaller projects. In these smaller situations, the .pco format is the perfect solution. While .pco files can hold tables of any size, they are most efficient for smaller tables.

In addition to being cross platform and native to Picalo, .pco files are automatically compressed to save disk space. This means that you do not need to "zip" .pco files to email (or otherwise transfer) them to colleagues. They can be sent easily and directly.

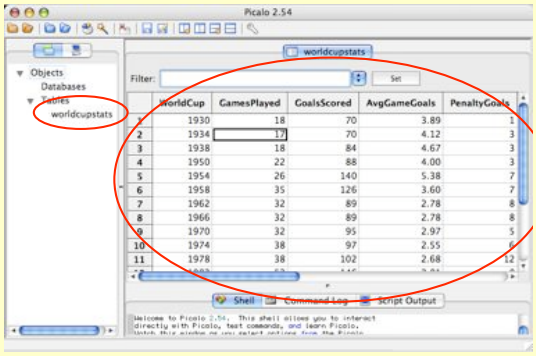
Load a Picalo Table

1. Select File, Open Table. The "Load Table" dialog comes up.



2. Find the `worldcupstats.pco` file on your disk and click OK. This file ships with these tutorials. After selecting the file, click OK in the "Load Table" dialog to finish the dialog and return to Picalo.

3. The table should load into Picalo. This table contains statistics from the World Cup tournaments through the last century. Note that the table is now listed in the list of tables (left red circle) and is showing in the table viewer (large red circle). You can move around the data with your mouse or cursor keys and edit data in cells.

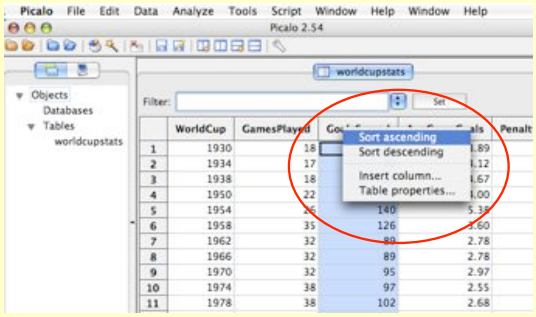


6.4 Sorting Tables

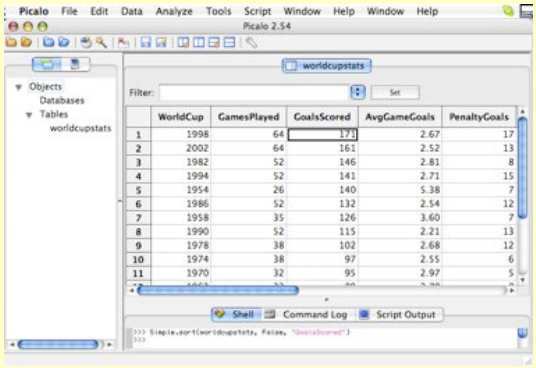
Picalo tables can be sorted to highlight the low or high values in a column. The easiest way to sort a table is using the right-click menu, as shown in the following example.

Quick Sorting

1. Suppose you want to know which World Cup had the most goals scored. With the `worldcupstats` table open, right click the `GoalsScored` column and select "Sort descending".



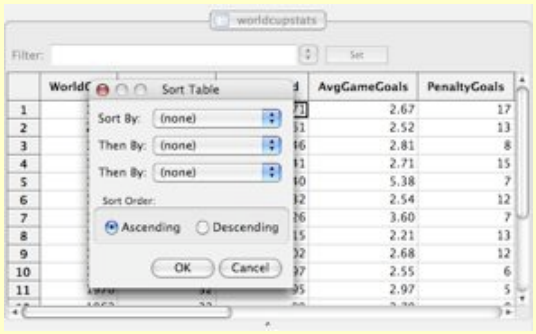
2. The table should now be sorted by the number of goals scored, showing a high number of 171 in 1998.



A more advanced way to sort is via the File menu. The sort option only appears in the File menu if you are currently viewing a table. The following example shows this method of sorting:

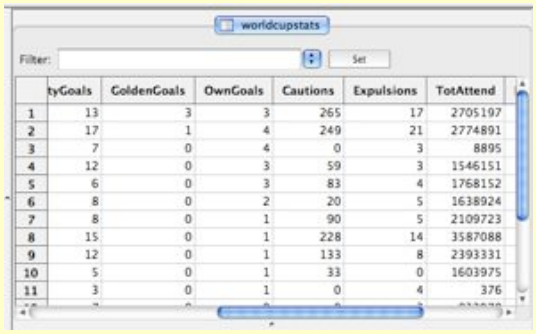
Advanced Table Sorting

1. Begin with the `worldcupstats` table open. Select File, Sort... to bring up the sort dialog, as shown in the picture. This dialog lets you sort by up to three columns.



2. Suppose we want to sort first by `GoldenGoals` and then by `OwnGoals`. Select these two column names, click the 'Descending' option, and click OK.

3. The table is now sorted by these two columns.



	tyGoals	GoldenGoals	OwnGoals	Cautions	Expulsions	TotAttend
1	13	3	3	265	17	2705197
2	17	1	4	249	21	2774891
3	7	0	4	0	3	8895
4	12	0	3	59	3	1546151
5	6	0	3	83	4	1768152
6	8	0	2	20	5	1638924
7	8	0	1	90	5	2109723
8	15	0	1	228	14	3587088
9	12	0	1	133	8	2393331
10	5	0	1	33	0	1603975
11	3	0	1	0	4	376

When sorting tables, it is important that you pay attention to the data types of the columns you are sorting on. Integers, floats, dates, and strings sort very differently. For example, the following numbers are sorted in ascending mode: 4, 12, 22, 165. However, if these values are typed as strings (rather than integers), the individual characters are seen as text rather than as numbers. Therefore, the sequence sorts as '12', '165', '22', '4'. See the tutorial on data types for more information about column types.

A third way to sort is directly through the Shell with Python commands. This method is the most powerful way to sort, but it is beyond the scope of this tutorial.

6.5 Editing and Adding Records

While Picalo is not a data entry application, it allow you to directly modify the data and add or remove records. To edit data the data in a cell, double click it or simply start typing in the cell. This behavior should be familiar to any reasonably-experienced spreadsheet user.

To add or remove records in your table, use the icons in the toolbar or the options in

the File menu. Picalo will remove the row your cursor is currently on. When inserting a new row, it inserts just before the current record. When appending a row, it adds the new row to the end of the table.

Each cell in a new record is given the *None* value, which shows as <N>. This value indicates the absence of data. See the tutorial about data types for more information about this value.

6.6 Copying and Pasting Table Data

Data in tables can be copied and pasted in the normal way. In fact, one of the easiest ways to transfer data from another application (like MS Excel) is to select the cells in the source application, copy to the clipboard, and paste to an empty table in Picalo. Similarly, cells can be copied from Picalo to other applications.

Note that when pasting data into Picalo, it starts the pasting at the currently-selected cell. If you want to paste an entire table of values, your cursor needs to be on the top-left cell. Picalo will automatically add as many records as needed for the pasted data. It will not, however, add any columns to your table. Only as much data as will fit across the columns is pasted, and the rest is discarded.

6.7 Importing Data

Data can be imported into Picalo in several other ways. While these methods are beyond the scope of this chapter, they are listed as the following:

- An ODBC connection to a database, then query the tables into Picalo.
- Import an Excel file into Picalo using the data import wizard.
- Import a delimited text file (.csv, .tsv) using the data import wizard.
- Import any other type of file using Python directly. For example, the Python language can import email messages into Picalo tables using the `email` module.

6.8 Closing Tables

Picalo tables that are loaded into the interface can be closed with varying degrees of permanence. These are described as follows:

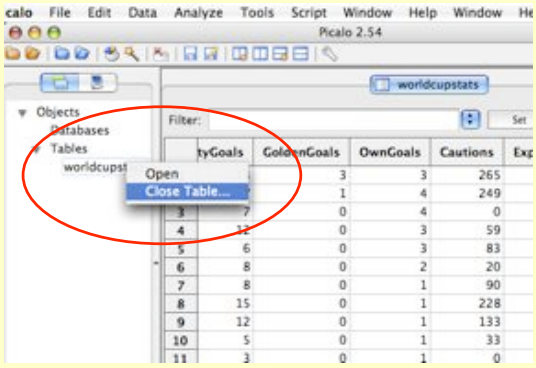
1. **Closing the Table View Tab**: When a table is visible in the interface, you can close it by 1) selecting Window, Close from the menu, 2) right-clicking the name tab for the table and selecting Close, or clicking the 'Close Tab' icon on the toolbar. This option only hides the table from view; it leaves the table data in memory and leaves the name in the table list on the left side. To reopen the table, simply double click the table name in the list. Any changes (saved or not) you made to the table are retained.

2. **Closing the Table**: While the previous method keeps the interface clean, it uses your working memory, which you have a limited amount of. When you are finished working with a table, you should remove it from working memory completely to make room for other tables. To do this, 1) right-click the table name in the left-side list and select Close Table, or 2) select File, Close Table from the menu. The table will no longer be accessible until you explicitly load it again. You must have saved the table (i.e. changes you have made to it in Picalo) to disk if you want to load the table again in the future.

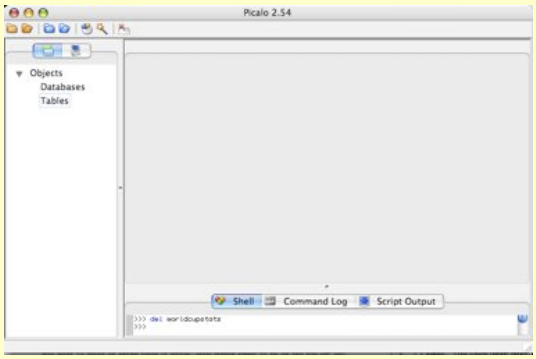
3. **Deleting the Table**: This method permanently deletes the table from working memory and from disk. The table is deleted if you loaded the file from a .pco, .tsv, or .csv file. Unless you have another copy of the table (such as on a backup disk), the table is gone forever.

Closing a Table

1. This example shows how to remove a Picalo table from memory but not from disk (method #2 in the text). With the `worldcupstats` table open, right-click the name in the left-side list of tables and select Close Table...



2. Picalo will prompt to ensure you want to close the table and then remove the table from working memory. The table is now removed from the interface as well as from the left-side list of tables. To reopen the table, select File, Open Table as we did earlier in this tutorial.



6.9 Rows and Columns: What are they?

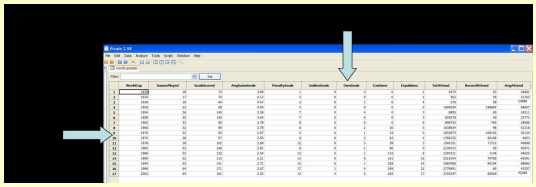
Upon inspection of Picalo tables, one might best relate these to spreadsheet documents. Tables are simply two dimensional grids of data. Just like a spreadsheet, the columns

and rows in a Picalo table represent different values. The rows (as indicated in the picture shown below) are frequently called *records* and represent an actual real world person, place, or thing. For example, if you had 436 employees working at GSL Inc., a Picalo table would have 436 rows or records; each row representing one and only one employee.

Columns, on the other hand, are not representative of a real world person place or thing, but rather one specific attribute to be defined for each specific person place or thing. These columns define characteristics and data that can be generalized for each row (or *record*). As a continuation of our example with GSL Inc. above, consider some general characteristics shared by all employees. Likely each will have a name, a date of hire, a date of birth, a department, an employee identification number, etc.... Notice that these specific attributes are not specific to any one employee, but are general to all employees; in other words, every employee has a name, a hire date etc.... For this reason, columns are often called *fields*. These columns, or *fields*, are also shown in the diagram below.

Rows (records) and Columns (fields)

1. Rows are shown by the arrow at the left, Columns are shown by the arrow at the top.



6.10 Inserting and Deleting Rows

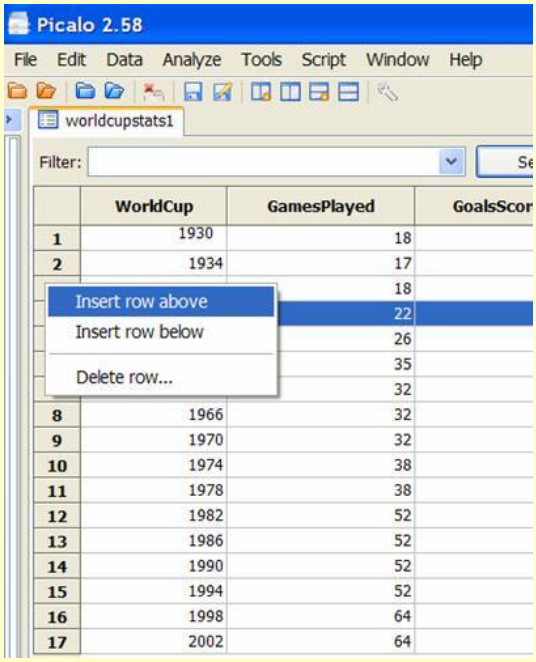
Users of the Picalo system will often find it necessary to insert or delete a particular record. To continue our example above, perhaps a particular employee record in our GSI Inc. data is missing, and so we will need to insert a place within the table to create a new record. On the other hand, maybe on of the employee records is invalid, therefore necessitating that we delete the record entirely from the table. The process of inserting and deleting records from a table is really quite simple.

We'll first address the topic of inserting rows into a table and afterwards address deleting rows. Rows can be inserted at any place within the table. You may have your table sorted alphabetically by name and many want to ensure that the record you add continues the alphabetical pattern. In this case, Picalo allows you to insert the record in the exact location you want the record to be placed in. In other cases you may want the record added to the very bottom or top of the table; Picalo allows this as well.

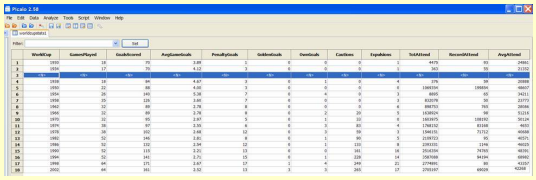
The following diagrams are meant to be a step by step guide for inserting new records into any Picalo table:

Inserting Rows in Tables

1. Identify the position at which you would like to add a row. Right click on the numeral in the row bar to the left. A popup menu will appear allowing you to insert a row above or below the currently selected row. In this case, we'll insert a row above. If you wanted to insert at the begining of the table, you would select the first record in the table, then right click on the number "1" in the row bar, then select the "Insert Row Above" option. Similar steps would be followed for inserting a row at the bottom on the table.



2. Notice that after selecting to create the row, the new row appears. We can now add attribute to this row by selecting each particular cell and typing the desired information. The letter "N" appears by default on newly created rows.

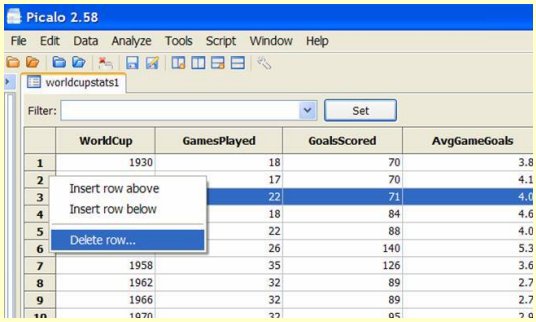


Now that we know how to insert a row into a table, the next topic to be discussed is deleting unwanted rows. Perhaps after creating a row, we realized that we've made a mistake and that we really don't need a new row after all. Deleting an existing record is very similar to creating a new record, except that we are going to select the "Delete" option instead.

The following diagrams are meant to be a guide for deleting records from any Picalo table:

Deleting Rows from Tables

1. To delete a specific row, right click on a specific row numerial, then select the "delete option."



2. A dialouge will ask whether you really want to delete the selected row. Selecting "Yes" will permenantly delete the selected row. Selecting "No" will return you to the table without any changes.

6.11 Insert and Delete Fields (*or Columns*)

Not only is it valuable to know how to insert and delete records in Picalo tables, but also how to add and delete fields (*or columns*) from tables. Inserting and deleting fields is a slight more involved process, but is still very userfriendly and simple to understand. Because each field is defining a particular attribute that is associated with each individual record in the table, field require that a data type be specified. Specifying the data type for an attribute ensures that the correct kind of data is being input into the attribute fields. For example, we don't want to have a Picalo user inadvertently enter a number in the field that is suppose to contain an employee's name; nor do we want an employee name in the "Date Hired" field. To reduce the risk of error in data entry, data types are defined at the time of each field (*or Column*) creation.

A list of these data types and a brief explanation of each is shown below:

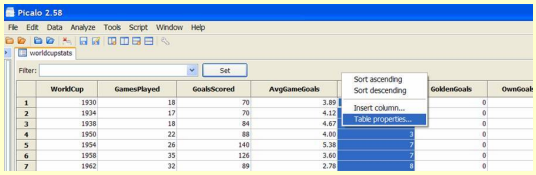
- `Sting = A series of text characters such as an employee name like "Bob White"`
- `Integer = A whole number such as 4 or -22(can't be a decimal, but may be negative)`

- `Floating Point = A number which can be a decimal such as 28.9047`
- `Currency = A monetary value`
- `Date = A specific date such as 2006-03-02`
- `Date Time = A Specific date and time`
- `True/False = Use in the case when an attribute must be either true or false`

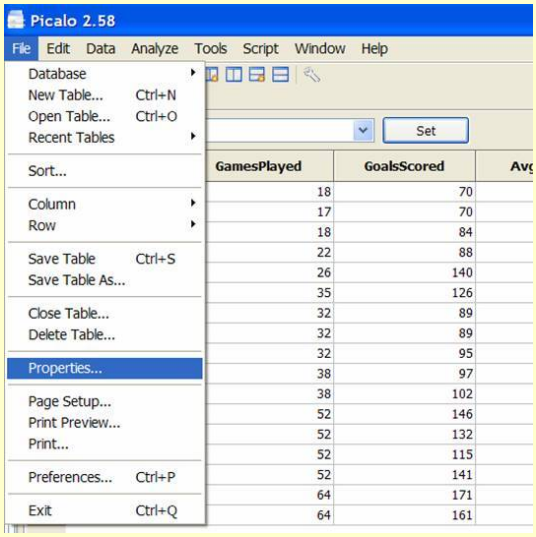
To insert a new field and declare its associated data type, you must go to the "Table Properties" interface. There are two ways to access the "Table Properties" interface. These two methods are shown below:

Accessing the Table Properties Interface

1. One way to access the Table Properties interface is to right click anywhere on the field bar at the top of the table, then select the "Table Properties" option.



2. The second way to access the Table Properties interface is to simply click on the "File" menu on the menu bar, then select the "Properties" option as shown.



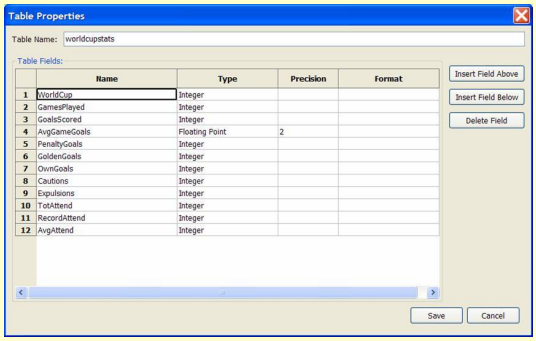
Once the Table Properties interface is displayed, the options to insert and delete specific fields becomes available. The insertion and deletion processes are very much like those of inserting and deleting rows shown in previous examples. For example, you may insert a new field at the beginning, at the end, or anywhere in the middle. This is done by simply selecting an existing field at which you want to insert a new field before or after, and then select the approprate insert button (*"Insert Field Above" or "Insert Field Below"*)

As stated previously, once a new field has been created, a data format will need to be specified. At the time of creation, a new field is given the "String" data format by default, but should be adjusted by the user as appropriate. In addition, if the "floating point" data type is selected, the user has the ability to specify the precision of that decimal number (e.g.; only 2 decimal points should be displayed).

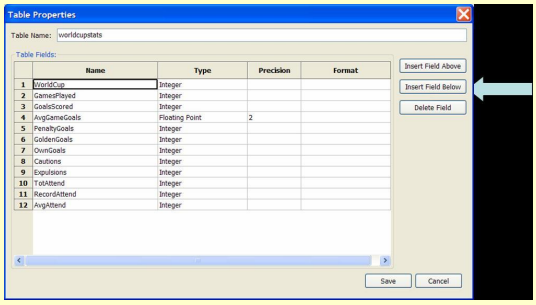
The following diagrams serve as a step by step guide to insert and defining a new field or column:

Inserting and Defining a New Field

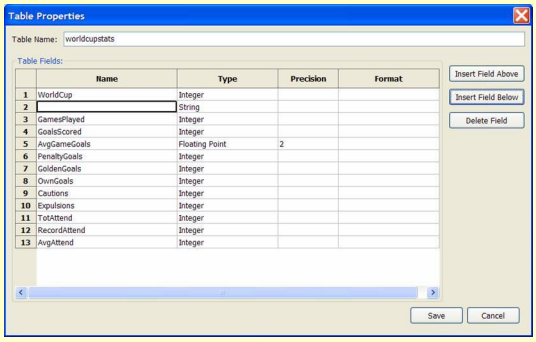
1. After following one of the two methods for access the Table Properties interface shown above, the following screen will appear. This is the Table Properties interface.



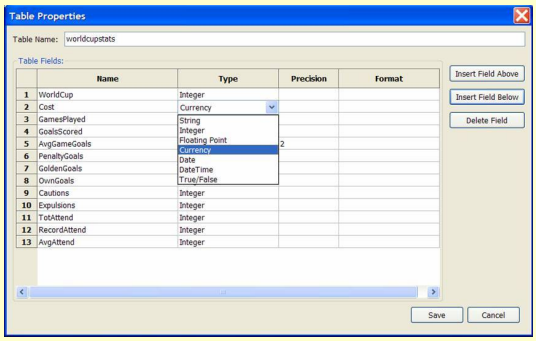
2. Lets say we want to insert a new field right under the first field currently listed. We select the first field, then click on the "Insert Field Above" button.



3. Notice that we now have a new field inserted under under the "World Cup" field. Also notice that even though this field doesn't yet have a name, its defaul data type is "String". You can also see that the "AvgGameGoal" field has a data type of "floating point" and therefore can be specified with precision on the decimal point. In this case, the decialm precision is set at "2".



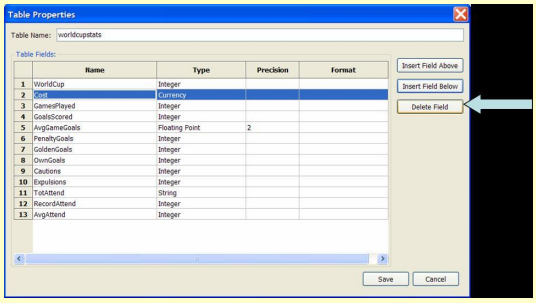
4. Now we will enter a name for our newly created field. Let's give it the name "Cost" and specify the data type as "Currency" since that is the most logical data type for money. We have now created a new field (*or column*) in our table.



Now that we have explored the functionality for creating a new field, let's take a quick look at deleting a field. The following servers as a guide for deleting fields:

Deleting Fields from Tables

1. To delete a field, simply select the field you wish to delete and then click on the "Delete Field" Button as shown.



2. After pressing the delete button, a dialouge will appear asking if you really wnat to delete that field. Pressing "Yes" will remove the field and pressing "No" will leave the table unchanged.

6.12 Saving Updates and Changes to Table Properties

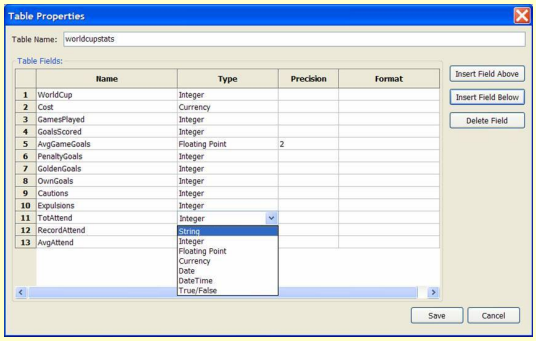
You may have also noticed that there is a "Save" button on the Table Properties window, clicking this save button will save any changes made and update the table accordingly. Perhaps we want to change the data type of a particular field along with our insertion of

the new "Cost" field shown previously. Perhaps we want the "TotAttend" field to be a type String instead. We may change this value by clicking on the data format cell, then selecting "String" from the dropdown menu. Now that all our table values are set properly, we can hit the "Save button" and the changes will appear in our table.

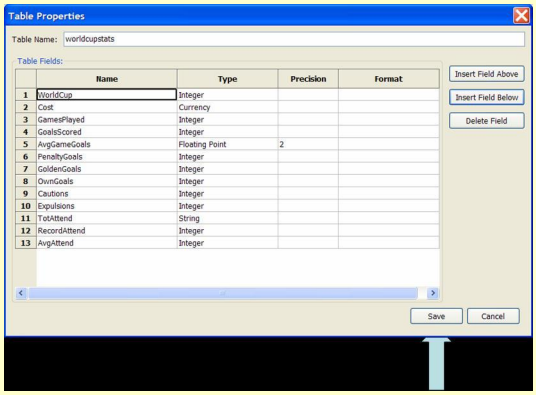
The following diagrams present a step by step guide for changing the properties of a field and saving those changes to the table:

Saving Updates and Changes to a Table

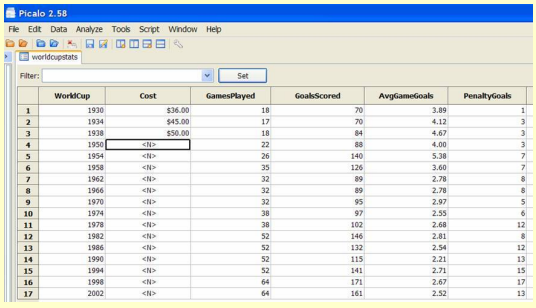
1. Here we change the data format of the "TotAttend" field from Integer to String. Notice the dropdown box that allow the user to select the desire data type.



2. Now that all desire changes have been made, we will click on the "Save Button" to save these new updates.



3. Shown here is the table after updating the Table Properties. Notice that the new field "Cost" is now available for each record in the the table. Also, as we enter values for the "Cost" field, Picalo auto-formats these values to appear as currency amounts because that is the specified data type. Notice that the letter "N" appears in those cells that have not yet been edited by default.



In closing, it should be noted that even though we have pressed the "Save" button in

the Table Properties window, we have not yet saved the actually table. If we were to close out of Picalo at this point without saving the file (*by going to File, then Save*), none of the changes made would actually be saved for future access.

Chapter 7

Changing Table Properties

This tutorial explains the process of changing table properties. This process can include adding or removing columns in a table, changing the type of data in a column, changing the format of data in a column, and changing the precision of data in a column. These functions are all facilitated by the Table Properties window. This interface allows for quick, flexible editing of table structures.

7.1 Learning Objectives

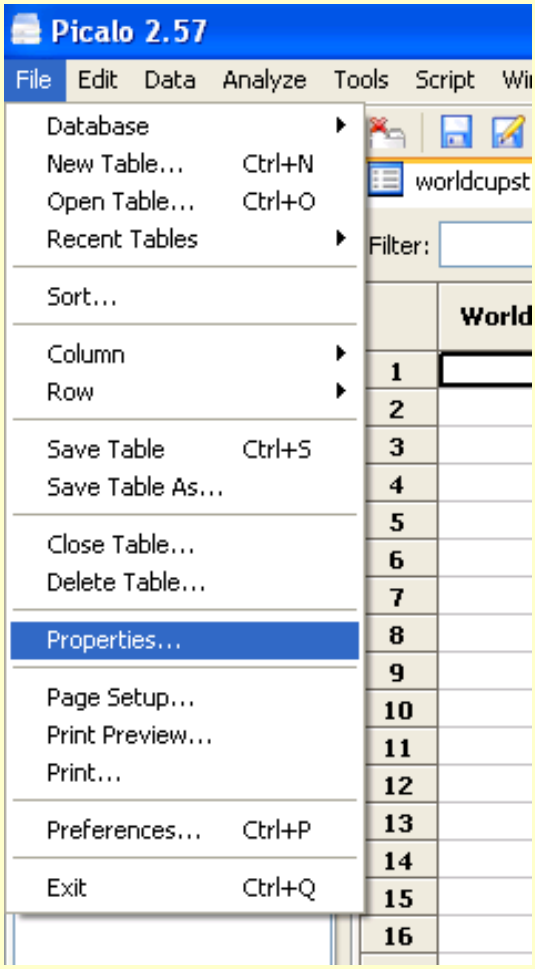
1. Become familiar with the Table Properties window
2. Add a new column to the table
3. Remove a column from the table
4. Change the type of data in a column
5. Adjust the precision of data in a column
6. Select the format of data in a column

7.2 Table Properties

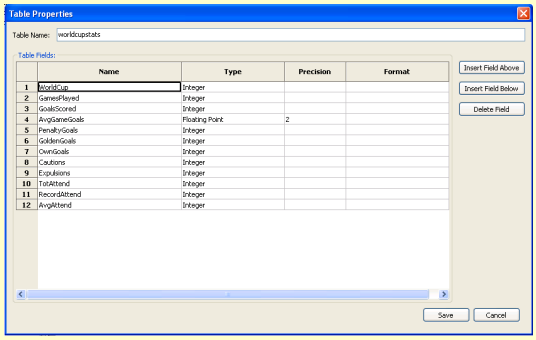
The Table Properties window facilitates all changes made to a table's columns. In this window you can perform all functions discussed in this tutorial, so it is important to become familiar with it. To access the Table Properties window, simply select the tab of the table you wish to edit. Then select File, Properties... The Table Properties window should appear, displaying all of the columns in the selected table. Within this window, you will be able to edit the columns of a table.

Opening the Table Properties Window

1. Select File, Properties...



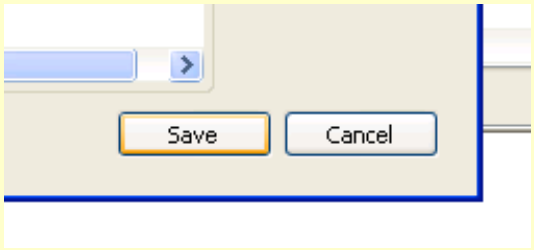
2. The Table Properties window should appear



Once all changes to the current table have been completed, select the *Save* button in the bottom corner of the window. This will save all changes made to the table and exit the Table Properties window.

Saving Changes

1. Click the *Save* button to commit all changes made to the table or select the *Cancel* button to undo all changes in the Table Properties window



2. The changes will either be saved or discarded and the Table Properties window will close

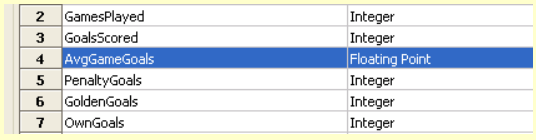
7.3 Adding Columns

Adding a column, also called a *field*, to the table is a common practice when working with data. This task is very simple with the Table Properties window. It also allows for insertion of columns at any point in the table, so a consitent or logical order can be maintained.

Once in the Table Properties window, select the row where you would like to insert a new column. Then select either *Insert Field Above* or *Insert Field Below* depending on where you would like the new column to appear. Once the new column is inserted, the column name can be entered and its data type selected (see section 1.5 for further explanation of data types).

Inserting Fields

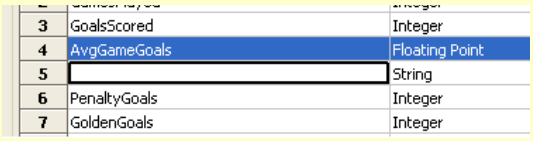
1. Select the row where you would like to insert a new field



2	GamesPlayed	Integer
3	GoalsScored	Integer
4	AvgGameGoals	Floating Point
5	PenaltyGoals	Integer
6	GoldenGoals	Integer
7	OwnGoals	Integer

2. Select the *Insert Field Above* button to insert a new field above the selected field, or select the *Insert Field Below* button to insert a new field below the selected field

3. A new row should appear once you select a button



7.4 Removing Columns

Deleting a column, or *field*, from a table is very similar to inserting a new column. Simply select the field to be removed. Then select the *Delete Field* button. A confirmation box will appear. Select *Yes* to delete the field, or select *No* to cancel the removal of the field from the table. The field should be removed if the deletion was confirmed.

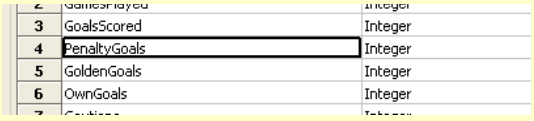
Deleting Fields

1. Select the field to be removed

2. Select the *Delete Field* button

3. Select *Yes* to confirm the removal of the field or *No* to cancel the action

4. If the deletion was confirmed, the field will be removed from the table



7.5 Data Types

The Table Properties window allows for many possible data types. An understanding of these data types is necessary in order to select the best option for each field. These data types are as follows:

- *String*: A string consists of any series of characters or numbers. Strings are useful for data that will not need to be used in mathematical calculations. Common uses for strings are names, addresses, phone numbers, and id numbers.
- *Integer*: An integer is a whole, non-decimal number. Integers can be used in mathematical functions. Common uses for integers are quantities and ages.
- *Floating Point*: A floating point is a decimal number. Floating points are useful for more precise data, such as averages and percentages.
- *Currency*: A currency number is useful for any monetary data. Currency data types will automatically round data to two decimal places for accurate storage of monetary values.
- *Date*: A date type is used to record dates. This saves the trouble of simply inserting strings for dates. Date types can be used in date calculations.
- *DateTime*: A DateTime type stores both the date and time for more precision. This data type is sometimes called a *timestamp*. It is useful for recording the exact times of transactions.
- *True/False*: A True/False type can only contain two values: true or false. This data type is similar to having a check box for each record in the table. It is helpful for determining if a certain condition is true for calculations or transactions.

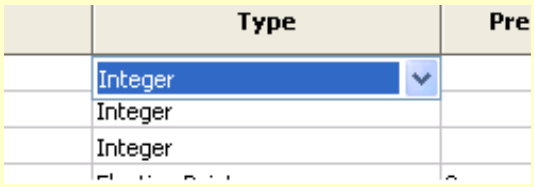
To change the data type of a field, select the field you would like to edit. Once the field is selected, click the *Type* column. An arrow will appear in the box. Click the arrow and a drop-down menu of all possible data types will appear. Select the desired data type and click a different row. The other columns (*Precision* and *Format*) will then be set to the defaults for the newly selected data type.

Selecting Data Types

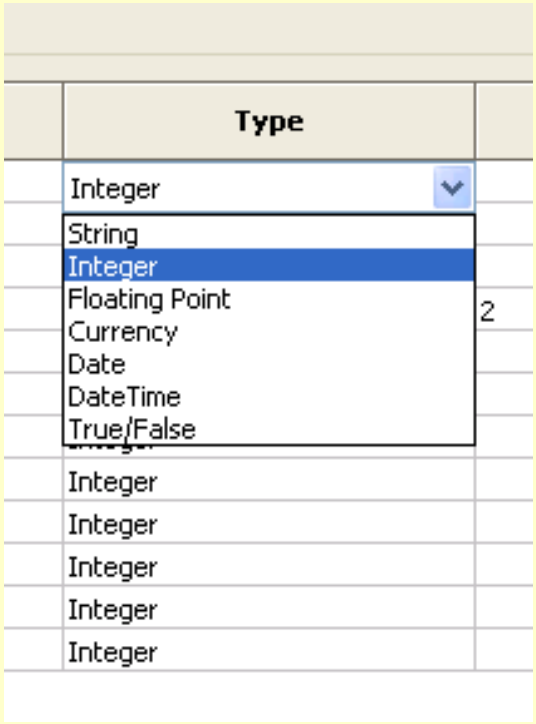
1. Select the field to be changed

2	GamesPlayed	Integer
3	GoalsScored	Integer
4	AvgGameGoals	Floating Point
5	PenaltyGoals	Integer
6	GoldenGoals	Integer
7	OwnGoals	Integer

2. Select the *Type* field and an arrow will appear



3. Click on the arrow and a drop-down menu will appear



4. Select the desired data type

7.6 Selecting Precision

The precision of the floating point data type can be specified. The number entered determines the number of decimal places the floating point will be rounded to. The default precision for a floating point data type is 2. Precision must be specified using an integer.

To change the precision of a floating point data type, select the row to be changed. Click on the *Precision* value for the field twice. The current precision will become highlighted and a cursor will appear. Type in the new precision value and click out of the cell.

Changing Precision

1. Select the field to be changed

2	GamesPlayed	Integer
3	GoalsScored	Integer
4	AvgGameGoals	Floating Point
5	PenaltyGoals	Integer
6	GoldenGoals	Integer
7	OwnGoals	Integer

2. Click the *Precision* value for the field twice

3. Enter the new precision value

4. Click out of the cell

7.7 Selecting Format

The format of a data type can be specified for the Date and DateTime data types. Despite the format, the same data is held in the field. The format is simply provided for the convenience of the user.

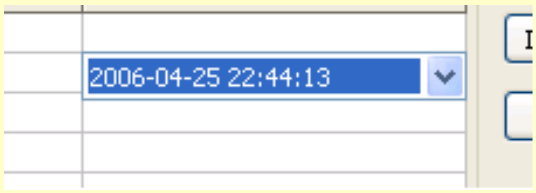
To change the format of a field, select the field to be edited. Click the *Format* cell twice and an arrow will appear in it. Click the arrow and a drop-down menu will appear with all possible format options. Select the desired format and click out of the cell.

Selecting Format

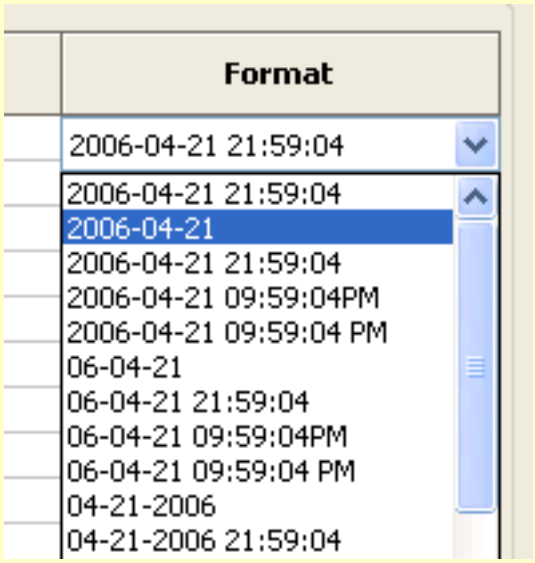
1. Select the field to be edited

2	GamesPlayed	Integer
3	GoalsScored	Integer
4	AvgGameGoals	Floating Point
5	PenaltyGoals	Integer
6	GoldenGoals	Integer
7	OwnGoals	Integer

2. Click the *Format* cell twice and an arrow will appear



3. Click the arrow and select the desired format



4. Click out of the cell

Chapter 8

Filtering

This tutorial explains the basic filtering methods in Picalo. Filtering is an essential function for analyzing data in tables. Filtering removes all records (rows) from a table that are not pertinent to your analysis. The removed records are not actually deleted, but only temporarily hidden. These records can be restored at anytime. Tables are filtered by expressions submitted by the user. In order to filter effectively, you will need to know a few of these basic expressions.

8.1 Learning Objectives

1. Why is filtering important
2. Different ways to create a filter
3. Simple filtering expressions
4. Complex filtering expressions

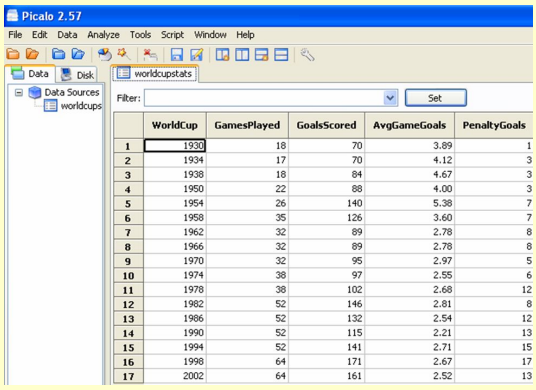
8.2 Why is filtering important?

Filtering allows you to see the most critical data in your table without any unnecessary data cluttering your view. Let's say you have a large table of data with World Cup statistics and you would like to see the statistics of only those World Cups that took place during the 1990s. Without any filtering applied to the table you would have to search manually for those World Cups in the 1990s amidst all the other data while trying not to get distracted or confused by the unrelated data. If you apply filtering, you would only see the data for those World Cups that occurred during the 1990s.

The two figures below demonstrate this example and the usefulness of filtering.

Unfiltered World Cup Stats Table

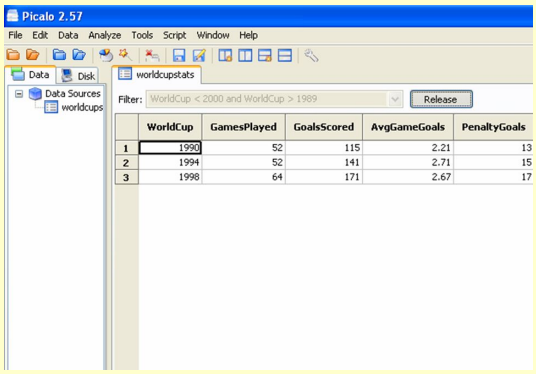
1. This is a table of data without any filters applied. As you can see, there are a lot of records–not all of which are pertinent to your search.



	WorldCup	GamesPlayed	GoalsScored	AvgGameGoals	PenaltyGoals
1	1930	18	70	3.89	1
2	1934	17	70	4.12	3
3	1938	18	84	4.67	3
4	1950	22	88	4.00	3
5	1954	26	140	5.38	7
6	1958	35	126	3.60	7
7	1962	32	89	2.78	8
8	1966	32	89	2.78	8
9	1970	32	95	2.97	5
10	1974	38	97	2.55	6
11	1978	38	102	2.68	12
12	1982	52	146	2.81	8
13	1986	52	132	2.54	12
14	1990	52	115	2.21	13
15	1994	52	141	2.71	15
16	1998	64	171	2.67	17
17	2002	64	161	2.52	13

Filtered World Cup Stats Table

1. This table has been filtered to only display World Cups that occurred during the 1990s. This table is much easier to read and makes it easier to find the data pertinent to your search.



	WorldCup	GamesPlayed	GoalsScored	AvgGameGoals	PenaltyGoals
1	1990	52	115	2.21	13
2	1994	52	141	2.71	15
3	1998	64	171	2.67	17

8.3 Creating a Filter

There are several different ways to create a filter. In this section you will be introduced to four of those ways. The following are the four methods which will be discussed:

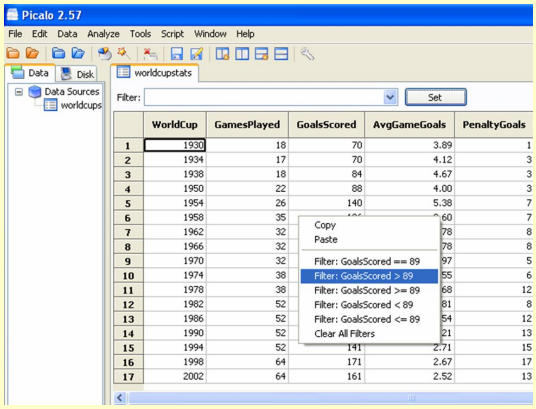
1. Right click the record
2. Use the filter box
3. Select by value
4. Select by expression

8.3.1 Right Click

The simplest and least powerful way to filter a table is by right clicking.

Right Click Method

1. Right click on the cell you would like to use as the delimiter.
2. Select the appropriate expression.

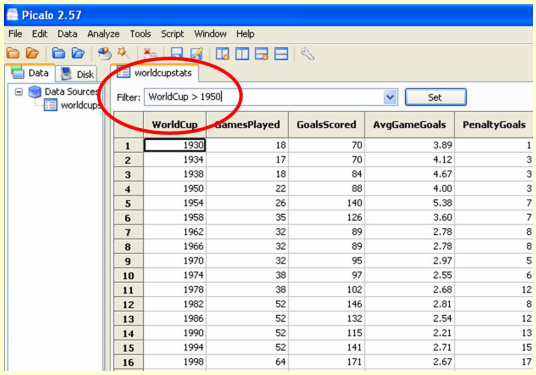


8.3.2 Filter Box

The second way to filter is by using the filter box. Filtering using the filter box is more customizable and, therefore, more powerful than using the right click method. But in order to use the filter box you must manually enter the expression. So the filter box is only as powerful as the extent of your knowledge of filtering expressions.

Filter Box Method

1. Type an expression into the filter box.
2. Either hit enter, or press the 'Submit' button.

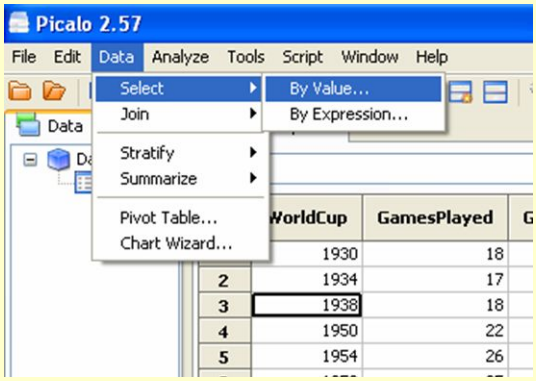


8.3.3 Select By Value

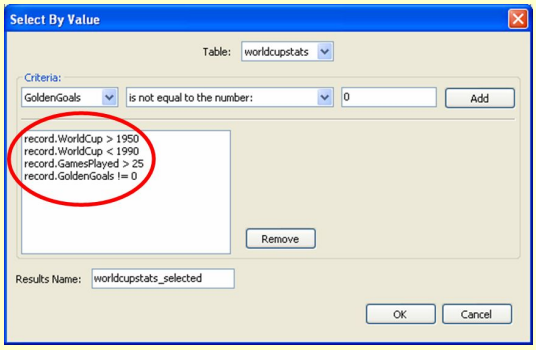
Another way to filter is by performing a select operation on the data. You can select by either value or expression. In this example we'll select by value. Selecting by value provides a simple way to create complex filters. Selecting by value allows you to combine several simple functions into one complex function. It is possible, for example, to filter the table so that you only view records where the World Cup happened after 1950, but before 1990; the games played that year were greater than 25, and there were some golden goals were scored.

Select By Value Method

1. Select 'Data' from the menu bar at the top of the window.
2. Highlight 'Select'.
3. Click on 'By Value...'



4. Choose the column value you are interested in using as a delimiter.
5. Choose the delimiting expression from the drop down menu.
6. Type the delimiting value in the box provided.
7. Press the 'Add' button.
8. Repeat these steps for as many expressions as you would like to combine together.
9. Press the 'Okay' button.



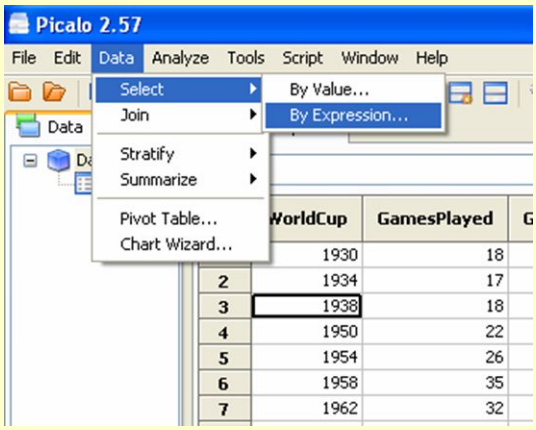
8.3.4 Select By Expression

You can also select by expression. This is very similar to filtering using the filter box. It is only as powerful as the extent of your knowledge of filtering expressions. Select by

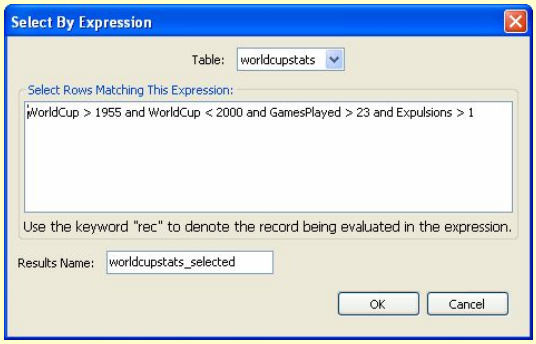
expression allows you to enter large expressions for complex filtering.

Select By Expression Method

1. Select 'Data' from the menu bar at the top of the window.
2. Highlight 'Select'.
3. Click on 'By Expression...'



4. Type the filtering expression in the box provided.
5. Press the 'Okay' button.



8.4 Simple Expressions

Simple expressions include comparatives such as greater than, less than, equal to, greater than or equal to, less than or equal to, and not equal to. Simple expression compare column values with a given value. A table of these expressions can be found below. Please take careful notice that equal to must be written with TWO equal signs (==), NOT just one equal sign (=).

Expression	Literal	Example
>	Greater Than	WorldCup > 1957
<	Less Than	GoalsScored < 40
>=	Greater Than Or Equal To	GamesPlayed >= 35
<=	Less Than Or Equal To	AvgGameGoals <= 2.8
= =	Equal To	Expulsions == 5
!=	Not Equal To	GoldenGoals != 0

8.5 Complex Expressions

Complex expressions can be made by comparing multiple columns against each other or by using operands like 'and', 'or', 'and not', and 'or not' to combine two or more simple expressions. See the table below for examples of each expression. Please note that each operand must be in all lower case. The filters are case sensitive. All column names must also be typed as they appear. For example, if the column name appears like so–'WorldCup'–the filter will not recognize 'worldcup'. It must be typed EXACTLY how it appears.

Expression	Example
and	WorldCup > 1957 and GoalsScored < 40
or	GoalsScored < 40 or GoalsScored >= 50
and not	GamesPlayed >= 35 and not GamesPlayed >= 55
or not	AvgGameGoals <= 2.8 or not AvgGameGoals > 3.1
Multiple Columns	PenaltyGoals + Expulsions < 9
	PenaltyGoals < Expulsions
	GoalsScored / AvgGameGoals > GamesPlayed

Chapter 9

Sorting Picalo Tables

This tutorial explains how to sort column information in a Picalo table. Sorting tables is an easy and efficient way to view information in an organized fashion.

9.1 Learning Objectives

1. Understand the difference between ascending and descending order
2. Sort tables using a simple right click
3. Sort tables using the file menu.

9.2 Ascending vs. Descending

Before sorting a table it is important to understand the difference between ascending and descending order and how they relate to numbers and letters.

Ascending will sort by placing the lowest numbers or first letters (A B C . . .)at the top of the table.

Descending will place the largest numbers or latest letters (Z Y X . . .) at the top of the table.

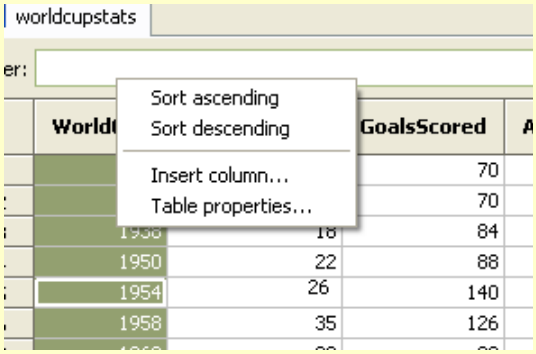
If a column contains both numbers and letters (this should only occur when the column is of type String), when in ascending order, numbers will come before letters (1 2 3 A B C) and when in descending order, letters will come before numbers (Z Y X 9 8 7).

9.3 Sort Using a Right Click

Sorting using a right click is the quick and easy way to sort the table by only using one column in the sort criteria. To sort by more than one column, see Sort Using the File Menu.

Sort Using a Right Click

1. To begin, right click the column name of the column to be sorted by. The white option box to the right will appear. Choose to sort the table by either ascending or descending order.



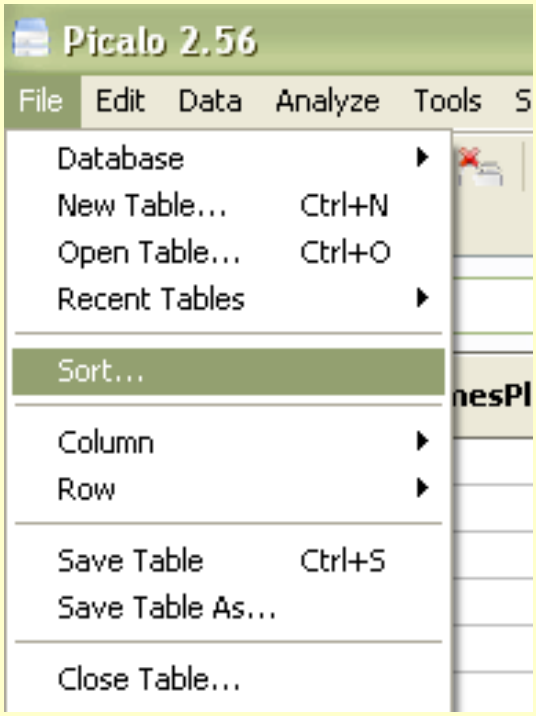
The Picalo table is now sorted by the chosen column.

9.4 Sort Using the File Menu

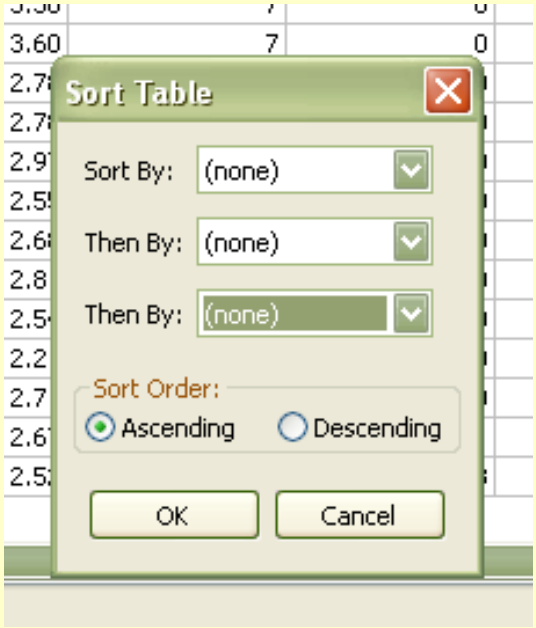
Sort using the file menu offers more criteria when sorting a table. In Picalo, a table can be sorted by up to three table columns.

Sort Using the File Menu

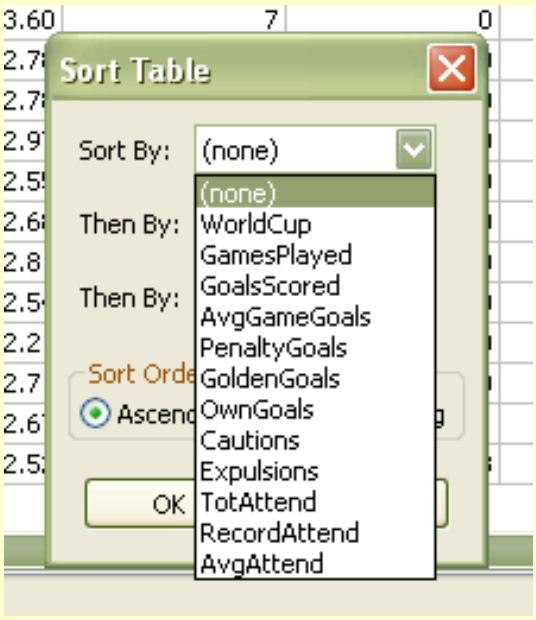
1. First, go to the File Menu at the top of the screen and choose Sort.



2. The Sort Table to the right will appear.



3. The Sort Table shows three drop-down menus. Each menu contains the names of all of the columns in the open Picalo table (see the image to the right). In the first drop-down menu choose the column to sort by first. In the second drop-down menu choose the column to sort by second. In the third drop-down menu choose the column to sort by third. Next choose if the columns should be sorted by ascending or descending order.



The Picalo table is now sorted by the criteria chosen in the Sort Table. For more complex sorting and viewing of the information in a Picalo table, see the tutorial on Pivot Tables.

Chapter 10

Database Access

In this chapter you will learn how to connect to a specific type of table referred to as a database. A database is similar to a table in that a database has rows and columns. Each row in the database is one record or instance of data. A database differs from a table in many ways. Unfortunately, discussion of those differences are not in the scope of this document. There is one key difference which you need to know: A database requires a connection in order to retreive the data.

This chapter will provide instructions on how to connect to various database services (e.g. MySQL, Microsft SQL Server, etc.).

10.1 Connecting to MySQL

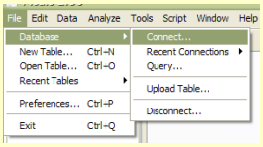
Connecting to a MySQL database is very simple. Before connecting to the database there are a few things to check first:

1. Know the name of the table you will be looking at
2. Ensure you have sufficient access to the database (username and password)
3. Verify your internet connection is active (if your database is on another computer)
4. Check to make sure the table you are accessing is located on the computer you are connecting to

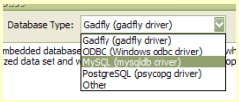
Once you have ensured all of the above items are fine, you can start your connection to a MySQL database.

Load a MySQL Database

1. In the menu, select File, Database, Connect.... The "Connect To Database" dialog comes up.



2. At the top of the dialog box is a dropdown list. This contains various database connection types. Select the item named "MySQL (mysqldb driver)".

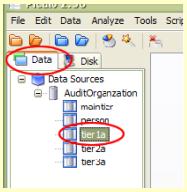


3. If the MySQL database driver is correctly configured, then the status message "Driver Status: Installed and Ready" will appear as shown in the figure. Fill out each of the empty fields.

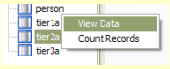
   - Connection Name: Make this name up. This name helps you quickly access the data in the future. Do not use any spaces, backslashes, or other characters of this type in the name.
   - Hostname: The name of the server you are connecting to.
   - Database: This is the name of the specific MySQL database you are connecting to.
   - Username: Your username for the MySQL database.
   - Password: The password that coincides with the username.

   Once you have completed filling out all the fields properly, click on the button labeled "Connect".

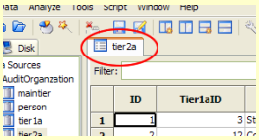
4. After the database connection has been established, the database will appear in the utility window on the left side of the Picalo application. Make sure the tab labeled "Data" has been selected. To view the tables stored in the database, expand the database by clicking on the "+" next to the database. As shown in the figure, tier1a is a table in the database AuditOrganization.



5. To view the data contained in a table, simply right-click the table you want to view and a menu will appear. Select View Data.



6. The data from the table will load up into the main area of Picalo. As shown in the figure, the name of the table selected is shown on the tab. If the name on the tab is different then the table you want to view, simply go back to the utility tab and find the table you desire.



10.2 Connecting to MySQL via ODBC

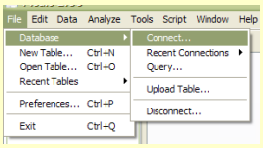
Microsoft uses ODBC as its standard for connecting to databases. It is possible to connect to a MySQL database using ODBC. Before you begin, make sure that the following have been satisfied:

1. Know the name of the table you are looking at
2. Ensure you have sufficient access to the database (username and password)
3. Verify your internet connection is active (if your database is on another computer)
4. Check to make sure the table you are accessing is located in the Window's ODBC Data Source Administrator

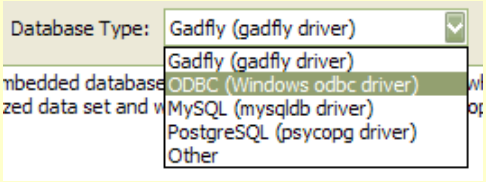
If you are unsure whether or not your Window's ODBC Data Source Administrator contains your datasource, open up Control Panel, Administrative Tools, and then Data Sources (ODBC). You will see a screen that has many different datasources. Scroll down until you see your datasource. (Note: This tutorial was written using MySQL ODBC Connector 3.51.12; if you experience any problems with your MySQL connector, please see your MySQL documentation.)

Load a MySQL ODBC Database

1. In the menu, select File, Database, Connect.... The "Connect To Database" dialog comes up.



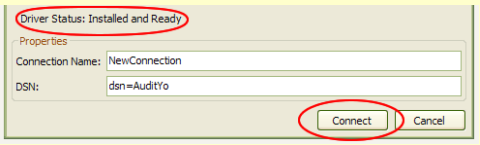
2. The dialog box has a dropdown list at the top. This contains various database connection types. Select the item named "ODBC (Windows odbc driver)".



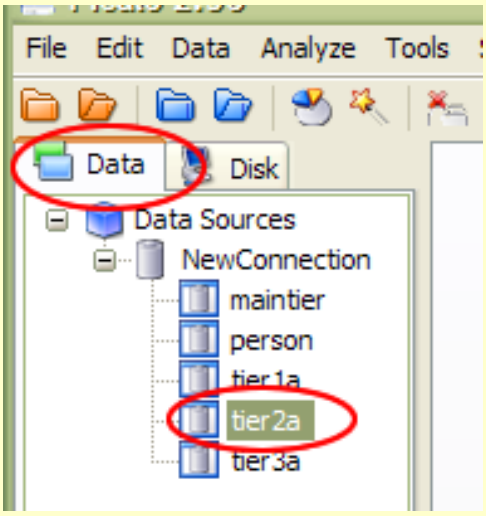
3. If the Windows ODBC driver is correctly configured, then the status message "Driver Status: Installed and Ready" will appear as shown in the figure. Fill out each of the empty fields.

   - Connection Name: This name helps you quickly access the data in the future. Do not use any spaces, backslashes, or other characters of this type in the name.
   - DSN: Type dsn=datasource where datasource is the name of the datasource found in Window's ODBC Data Source Administrator.

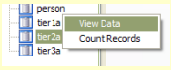
   Once you have completed filling out all the fields properly, click on the button labeled "Connect".



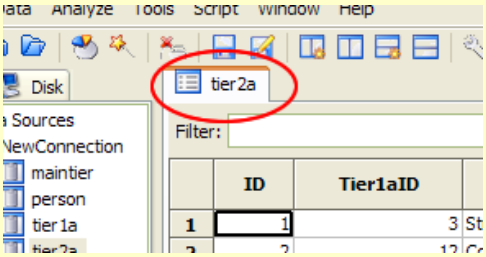
4. After the database connection has been established, the database will appear in the utility window on the left side of the Picalo application. Make sure the tab labeled "Data" has been selected. To view the tables stored in the database, expand the database by clicking on the "+" next to the database. As shown in the figure, tier2a is a table in the database AuditOrganization.



5. To view the data contained in a table, simply right-click the table you want to view and a menu will appear. Select View Data.



6. The data from the table will load up into the main area of Picalo. As shown in the figure, the name of the table selected is shwon on the tab. If the name on the tab is different then the table you want to view, simply go back to the utility tab and find the table you desire.



10.3 Querying a Database for Information

Database queries follow a format known as Structured Query Language, or SQL. There are numerous books that explain and itemize SQL format. This being, the purpose of this tutorial is not to explain multiple queries, but rather to give very basic guidelines on the subject.

SQL focuses on readability. As such, it is not difficult for someone completely new to SQL to understand the logic of what is happening behind a query. There are certain reserved words in SQL. These words are to be used as commands, so databases, tables, and attributes cannot be given these names. This generally isn't a problem, as database engineers are very familiar with SQL.

The reserved words that we will cover are as follows:

- `Select` - Used to establish what attributes we want selected from the query (Hence the word. Readability is key)
- `From` - Used to establish which table or tables we are querying for information.
- `Where` - Establishes the criteria of our search. A query *can* take place without this word, but is much more robust is the user supplies a criteria.

An SQL query has a certain structure. The word `select` comes first, then `from`, then `where`. The structure is as follows:

- `SELECT` *attribute1, attribute2* `FROM` *tablename* `WHERE` *condition = 'x'*

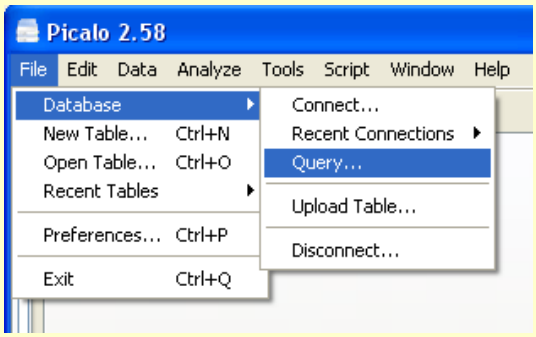
Notice how you can select multiple attributes by separating them with commas. This is useful if you want to display certain attributes of a record but not others. In the select clause, you may also include an asterix (*), and all attributes will be displayed. The tablename, attributes, and condition criteria must be spelled the exact same as they are spelled in the database (that includes upper- and lower-case letters). If a condition criteria is a numeric value, no apostrophes are needed, but if it has characters in it, it must be surrounded by apostrophes.

Basically, a select query is saying "SELECT these attributes FROM this table, WHERE these criteria are met."

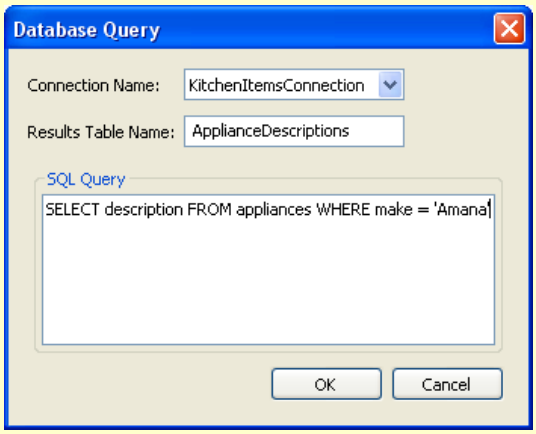
Keeping with the kitchen items example, lets use our *KitchenItemsConnection* that we established in the previous example and write a query to display the description of each appliance.

Query Appliance Information from the Kitchen Items Database

1. Click on *File, Database, Query...*



2. The following menu will appear.



In the "Connection Name" box you will select the connection that you wish to use.

Since we wish to use the connection that we created to the Kitchen Items database, we will select *KitchenItemsConnection*.

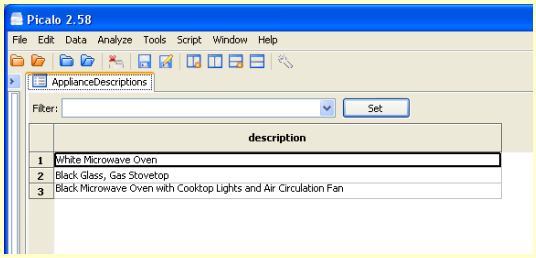
The data will be returned in a table, so we must name it. Name it something descriptive of what the query will accomplish.

The SQL Query text area is where we will actually type our query. What will be displayed with the query that is shown? Remember that `SELECT` determines what will be shown, so we will be seeing a description, but a description of what?

`FROM` shows us where the description will be coming from, so it will be from the table *appliances*.

`WHERE` tells us what the criteria of the search are. In this case we are limiting the search to those records that have Amana as their "make" attribute. So if we take them all together, we are looking for the description of any appliance that is made by Amana.

3. The results of the query are shown. There are three products that matched our criteria. Their descriptions are shown.



4. It's that easy! Simply select a connection, name the table that the information will be displayed in, and enter the query! If we had written `SELECT` * `FROM` appliances `WHERE` id = '123451', what would have been displayed? All the information about the appliance who's ID number is 123451.

Chapter 11

Joining Two Tables Together

This tutorial explains how to join two tables together with Picalo. This feature allows to you take information from two tables and combine it into one table according to values you specify. This allows you to make quick comparisons between tables and use information from both at the same time. This tutorial will explain how to do this with a regular join and with a fuzzy match.

11.1 Learning Objectives

1. Understand the usefulness of joining tables
2. Join tables with a regular join
3. Join tables using a fuzzy match
4. Analyzing resulting tables

11.2 Joining Tables

You can join two Picalo tables together according to certain values. If you want to determine what results come about by matching a value in one column of a table with the same or different column in another table, a join will assist you in doing so. Once you specify which value you want to match in both tables, the joined table will only show the rows that have a match for those two values.

As you begin to determine which tables you want to join, there are some important facts you must consider:

- If there are no matching values in the two columns that you specify, the resulting table will come up blank. This isn't a mistake - it just means that there were no matching values in those columns.

- It is possible for you to join by more than one criteria. This can save you the time it would take to sort your results and determine which rows you wanted.
- Remember to give the table you're going to join the values to a logical name that describes the join of two tables. If you do so, it will be less confusing, and the program won't give you an error message when you try to join the tables.
- In a regular join, the two values in the tables must match exactly. In a fuzzy join, the two values can be similar, but not necessarily exact.

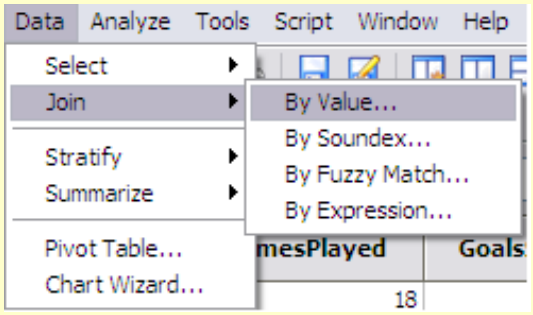
Using joins can save you a significant amount of time if you're trying to sift through a large number of records. For example, if you have a table that contains customer information, and another table that has information about transactions, a join would aid you in determining specific information about each customer and his/her transactions. You could simply do a join where the customer ID in the first table matches the customer ID in the second table. The resulting table would have that customer ID, information, and each transaction that customer participated in.

This specific join would save you the trouble of having to sort each table by customer ID, then try to find the desired customer ID in the first table and compare it to the customer IDs in the second table to see which transaction the customer participated in. A join allows you to see all of this information at one time, reducing errors and frustration trying to switch between tables.

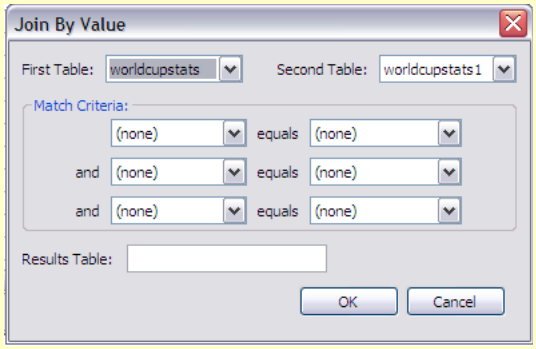
The following is an example of a regular join in Picalo.

A Regular Join in Picalo

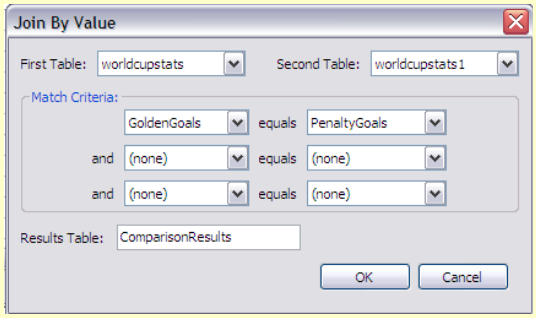
1. Select Data, Join. Select the 'By Value...' option.



2. Select the two tables from the drop-down lists that you want to join. You are going to choose values from a column in each of these tables that you want to match.

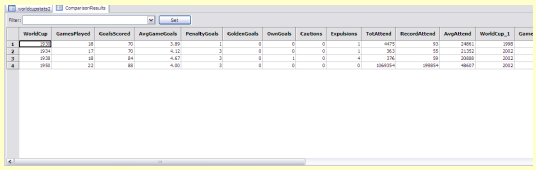


3. In the first set of drop-down lists, specify which columns in the two tables you want to have matching values for your new join table. If you want to retrieve rows that have matching values in more than one column, specify those columns you wish to have matching values in the second and third set of drop-down lists.



4. Name the results table - this will be the name of the new table that has the values you joined together from the two tables

5. Press OK. Your joined table should now appear with the name that you specified in its corresponding tab at the top of the page. The table only has fields where the values you specified match. This table can now be sorted and edited like any other table.



11.3 Joining Tables Using Fuzzy Match

Because there may be times when you want to join tables, but there may not be exact matches, Picalo provides the capability of fuzzy matching. This allows you to compare columns in the tables, but make joins where matches are not exact, but similar. If the value in one column was 'match', the following could be valid names in the column in the other table that would allow a fuzzy join:

- `match`
- `matched`
- `matcher`
- `matching`
- `matchbox`

Warning: Picalo's fuzzy matching algorithm is a powerful way of matching. However, it is exponential in its algorithm. We are currently working on ways of speeding it up, but it is not possible to use normal indices (which speed things up considerably) like databases

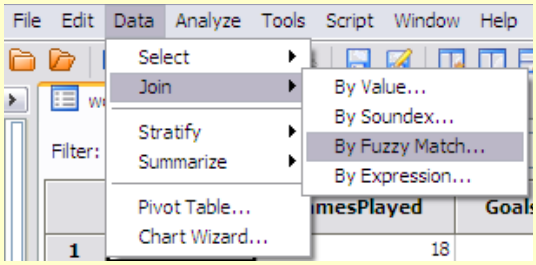
or Picalo's regular join command use. Instead, the fuzzy match algorithm must match each value in the first table with each value in the second table.

To illustrate, suppose you have two tables with 10 records each. You'll have to compare 100 records to compare each value in the first table to each value in the second table. Now suppose you have two tables with 1000 records each. You now have 1,000,000 comparisons to make. Suppose you have two tables with 10,000 records each. This equates to 100 million comparisons.

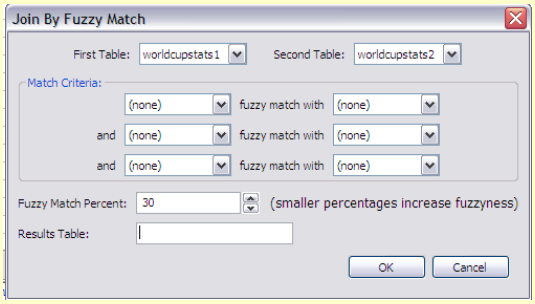
The following is an example of how you would join a table by fuzzy match:

A Fuzzy Join in Picalo

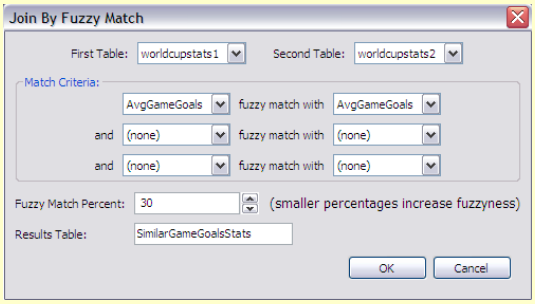
1. Select Data, Join. Select the 'By Fuzzy Match...' option.



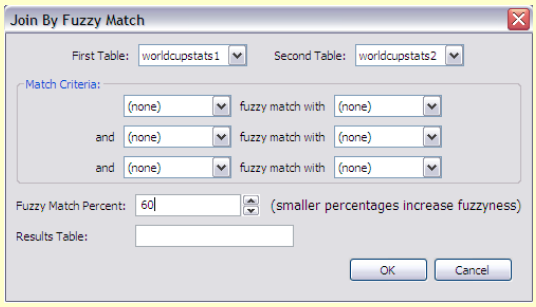
2. Select the two tables from the drop-down lists that you want to join by fuzzy match.



3. In the first set of drop-down lists, specify which columns in the two tables you want to have similar values for your new join table. If you want to retrieve rows that have similar values in more than one column, specify those columns you wish to have similar values in the second and third set of drop-down lists.

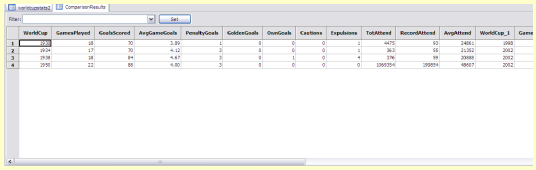


4. Specify how similar you want the values to be. If you put a smaller percentage in this box, the values can be less similar. If the percentage is very high, the values have to be quite similar if the new table is going to consider them a match.



5. Name the results table - this will be the table that has the values you joined together from the two tables

6. Press OK. Your joined table should now appear with the name that you specified in its corresponding tab at the top of the page. The table only has fields where the values you specified are similar. This table can now be sorted and edited like any other table.



11.4 Analyzing Resulting Tables

Tables are designed to eliminate redundant data. If we use the customer example again, you don't want to have to list each customer's information in the transaction table for every transaction that customer participated in. Instead, you have the customer's ID in that table that references all the customer's information in a different table. That way, you save space and the time it would take to enter that customer's information in over and over.

Unfortunately, this makes it more difficult to do data analysis. Joins and fuzzy joins provide the means for you to join those tables back together, but only the fields that you need.

Continuing with our customer example, if we want to know the city names where customers who bought a certain type of candy live, then we'd filter the transaction table to only show that type of candy. We'd then join the customer table with the transaction table where the customer ID in each table was equal. That way, we can see the customer information for each customer that bought that candy.

The table that was joined can then be analyzed. Because you're only viewing customers that bought the type of candy you're analyzing, you can see which cities most of those customers live in, or the household size of customers who bought that candy. By joining the tables, you saved yourself a great deal of time because now all the information can be viewed and analyzed in one table.