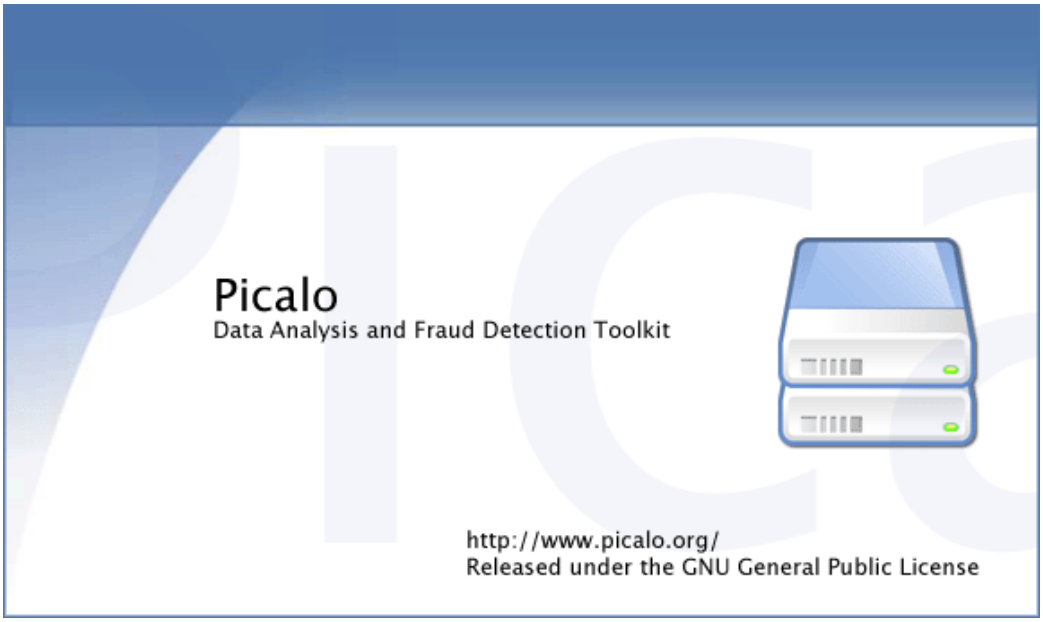

Picalo Workbook

Conan C. Albrecht, PhD

December 2, 2009

Contents

About This Book

The purpose of this book is to lead you through a number of exercises that will help you learn Picalo. This is the perfect place for new users to start; the workbook doesn't assume that you know anything about programming, data analysis, or Picalo. The exercises start with very basic concepts (like opening a table) and continue through more complex tasks (like searching for fraud).

The datasets included with this workbook are entirely fictitious. They were generated through Picalo scripts. Any seeming relation to people, addresses, names, or other real world objects is purely coincidental. Be aware that since the datasets were generated artificially, they may not make sense at times. They are sufficient for the purposes of this book, but probably not much more than that.

The workbook is set in the context of fraud detection. Fraud is one of the initial problem areas I'm targeting Picalo towards. However, please realize that Picalo is a general-purpose data analysis toolkit. It is useful in many different contexts.

Philosophy

I have taught people to use the computer for many years. In this time, I have found that students learn best when they discover the computer for themselves. If I simply give you a sequence of steps to follow, you'll become very proficient at following directions, but you won't learn much about Picalo.

As you look through the exercises in this book, note how they tell you *what* to do, not *how* to do it. When you are asked to do a task, explore the Picalo menus. Brainstorm creative ways to solve the problem – there's usually more than one solution anyway. This workbook is an experience, not a reading book.

To repeat this philosophy, I believe that the best way for you to learn is to struggle with these exercises *before* you go through the answer key. While this may be a little frustrating at times, I promise you'll learn more effectively than if you simply follow the answer key.

If you have always been the computer user that needed a list of instructions for any task, give yourself the freedom to struggle a bit; get frustrated; walk away and think about it a bit; discuss the problems with others. Don't go straight to the answer. You may just find what it means to be a computer nerd. "Technology-enabled" people get frustrated with the computer all the time! Despite their frustration, they continue to explore, try different things, and hit their head into the technology walls until they find the solution. Over time, they get better and better at solving problems, and that's what makes them different. They are usually no smarter than the "technology-disabled" group – they have simply learned to work through their walls through hard experience.

Contributions

I welcome contributions to this manual. If you have new exercises to submit, email them to me and I'll attribute it to your name in the text. I reserve the right to make final decisions on the content of this book.

A Moving Target

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. I'll try to keep the workbook updated with the program, but I simply do not have enough time to update the workbook in perfect unison with the program. Since the Picalo program itself is the main priority, the workbook may be slightly behind the actual features or user interface of the program.

Exercise 1

Picalo Tables

Goal: To learn basic table operations like opening, saving, and modifying tables.

This exercise uses the Purchases dataset, which includes the following tables:

- `approved_vendors.pco` - Picalo table containing the names of vendors that have been approved by our company.
- `purchases.pco` - Picalo table containing a record for each purchase made by our company.

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

1.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `Purchases`. How many initial tables are in this project?

2. Look at the content of the tabs along the bottom. What are two differences between the Shell and History tabs?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `Purchases`. How many initial tables are in this project?

   Two tables. To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the Purchases directory. Once you set the directory, you'll see the tables in the project listed on the left. You may have to click the little 'plus' icon next to 'Tables' if they are not showing.

2. Look at the content of the tabs along the bottom. What are two differences between the Shell and History tabs?

   The Shell is where Picalo runs commands. Every time you make a selection from the menus, you'll see a command run in the shell. While most programs hide their command engine, Picalo shows it on the main screen to help you learn the commands. Advanced users places these commands into scripts or even enter them directly in the shell. The History is simply a record of all the commands you have run. This is different than the Shell because 1) it contains a timestamp for each commands, and 2) it keeps your commands forever (even from past sessions in Picalo).

1.2 Get Help

There are several manuals that will help you learn how to use Picalo. These are available on the `http://www.picalo.org` web site or on the *Help* menu of the program. These include the following:

- Introductory Manual - Contains step-by-step instructions for beginning users.
- Picalo Cookbook - Presents task-oriented solutions in a familiar "recipe" format. This is one of the best manuals for Picalo because it provides both menu-driven and script-oriented solutions for problems.
- Picalo Workbook - This manual.
- Advanced Manual - Contains information for script writers and detectlet developers. Once you learn Picalo, come find out how powerful its scripting language is. This is where you'll see the true power of Picalo!
- Online Python Tutorial - Contains a great introduction to Python (Picalo's scripting language). The tutorial is general to Python and does not contain any Picalo specifics.
- README.TXT - Contains important information about the current release as well as the change log for all past releases.

In addition, there is an email list available on the Picalo web sites. Questions asked on this list usually get prompt responses from both the developer and users. This is often the best way to get specific questions answered.

1. Open the different manuals from the Help menu. Focus specifically on the Introductory Manual and Picalo Cookbook. What are the difference between these two manuals?

2. According to the Cookbook, what is the difference between a "calculated column" and a "static calculated column"? Which can be created from the GUI? Which can't?

Answer Key: Get Help

1. Open the different manuals from the Help menu. Focus specifically on the Introductory Manual and Picalo Cookbook. What are the difference between these two manuals?

   The Introductory Manual gives a true introduction to the Picalo environment, including why it was developed, a tour of the parts of the main screen, and introduction to concepts like tables. The Picalo Cookbook is useful when you have a specific task you want to accomplish. It gives recipes for most of the things users want to do with the program.

2. According to the Cookbook, what is the difference between a "calculated column" and a "static calculated column"? Which can be created from the GUI? Which can't?

   A calculated column provides an active calculation, which means that if the data it is calculated from changes, it will change automatically. A static calculated column runs your calculation once and records the result in the new column. This result will never change because the calculation is not saved. Static calculated columns are much more efficient because Picalo only has to run the calculation once. Right now, static calculated columns can only be created with Picalo scripts.

1.3 Opening Tables

Tables are the basis for most of what happens in Picalo. Data are kept in rows and columns in tables. Most Picalo commands start with one or more tables (the input tables) and end with a table (the results).

Picalo tables can also be split into table lists: groups of tables with the same fields but different data in each table. These are explained later in the workbook.

1. Double click the "purchases and "approved_vendors tables to open them in Picalo. How does the left-side list change when you open a table? What is a variable name?

2. How are these two tables related?

3. Extract records 5-10 from the Purchases table to a new table (hint: see recipe 1.6, Retrieve Several Table Records by Index, in the Picalo Cookbook).

4. Why do Picalo tables start with record 0 instead of 1?

Answer Key: Opening Tables

1. Double click the "purchases and "approved_vendors tables to open them in Picalo. How does the left-side list change when you open a table? What is a variable name?

   You may have to click the little plus icon to show the available tables underneath the Tables item in the left-side tree. When you double-click a table, it opens into a tab on the right. The table is now "open" and ready to be used. It is assigned a variable name based on its file-name, and this variable name is how it will be referenced throughout the program. You can even type the name into the shell below to get information about the table.

2. How are these two tables related?

   The purchases table contains a set of purchases made by three different company employees: Vijay, Adam, and Suzie. The VendorCode field should match an approved VendorCode from the approved_vendors file. The ProductCode lists the product ordered. The approved_vendors table contains the list of vendors who have been approved for use by our company.

3. Extract records 5-10 from the Purchases table to a new table (hint: see recipe 1.6, Retrieve Several Table Records by Index, in the Picalo Cookbook).

   You should have a table with 6 records, with row index 0-5.

   - *With the purchases table open, select Data — Select — By Record Index.*
   - *Include from row 5 to row 10.*
   - *Enter 'results' for the results table name (any name will do here). Click OK.*

4. Why do Picalo tables start with record 0 instead of 1?

   This is one of the most common questions new users ask. Note that your results table starts with 0, not 1 as you might expect. Picalo starts tables at zero because its underlying scripting language starts everything with zero (instead of one). Many computer languages (like Java and

C++) do this same thing. It takes some getting used to, but those users who script with Picalo come to appreciate this consistency with the language.

1.4 Modifying Values

Picalo tables can be read-only or modifiable. Auditors, for example, generally prefer tables to be read-only so they don't change any data. Data analysts often want to change values or even build new tables from scratch.

1. Change the purchaser of the first record (row zero) to Vijay. Then right-click and edit a value in a separate dialog.

2. Make the table read-only. Try to change the purchaser of the first record back to Adam.

3. Remove the read-only flag (see the previous task) so the table is once again editable. Find all values of AAC09 and replace with AAC10 (hint: find and replace). How many replacements did you make?

4. Insert a new record in the area for Invoice #10. The data are 2009-01-13, Suzie, SOO37, 25.30. How many records are now in the table?

5. Close the table (dont save changes).

Answer Key: Modifying Values

1. Change the purchaser of the first record (row zero) to Vijay. Then right-click and edit a value in a separate dialog.

   This exercise showed you two ways to edit/view cell data in Picalo. The first method is simply to select the cell you want to change and start typing. The second method is to right-click the cell and select 'Edit in Separate Dialog'. This second method is useful when cells contain more data than you can see on the screen (such as full paragraphs).

2. Make the table read-only. Try to change the purchaser of the first record back to Adam.

   To make a table read-only, click the little lock icon just above the table cells and under its tab (to the left of the Filter Expression text). This button stays down as long as the table is read-only. Click it again to make the table editable again. When you try to edit a record on a read-only table, either through the GUI, a script, or otherwise, Picalo will give you an error message.

3. Remove the read-only flag (see the previous task) so the table is once again editable. Find all values of AAC09 and replace with AAC10 (hint: find and replace). How many replacements did you make?

   You'll make 18 replacements.

   - *Ensure the purchases table is showing (not the results table from before).*
   - *Select Edit — Find and Replace menu.*
   - *Enter AAC09 (that's A-A-C-zero-nine) in the 'Search For' box and AAC10 in the 'Replace With' box. Picalo can search within partial cell values on some tables, but since this table contains both text and numbers, you need to check the 'Match Entire Cell' check box.*
   - *Then click the Replace All button. Picalo will make the changes and report that it made 18 replacements, assuming you started at the top.*

4. Insert a new record in the area for Invoice #10. The data are 2009-01-13, Suzie, SOO37, 25.30. How many records are now in the table?

8,371 records should be in the table.

- *Invoice #10 is in record 16 of the table. Select one of the cells in this record.*
- *Select File — Row — Insert Row. A new record will be inserted into the table just above the one you had selected. (This could also have been done by right-clicking row 16's row header.)*
- *Enter the data as described. If you look in the bottom-right corner of the Picalo window, you'll see that the table has 8,371 records.*

5. Close the table (dont save changes).

   Picalo tables can be closed in two ways:

   - *You can simply click the little X icon in the table's tab header (this currently only works in Windows and Linux).*
   - *You can click the close icon in the application's icon bar (fourth from the left).*

   Either of these options will close a table in the Picalo notebook area. If a table has unsaved changes, you will be prompted to save, continue, or cancel.

1.5 Sorting

Picalo can easily sort tables by any column. Once a table is sorted, it can only be changed again by sorting on another column. Tables can be sorted on any data type: integer, date, string, etc.

1. Sort the purchases table by the Amount column, with the largest amounts first (hint: right-click the column name). What is the largest amount in the table?

2. Sort the table by two columns in ascending order: first by Date, then by InvoiceNum (hint: File menu). What purchaser name is listed first after the sort?

Answer Key: Sorting

1. Sort the purchases table by the Amount column, with the largest amounts first (hint: right-click the column name). What is the largest amount in the table?

   The largest amount is 99,970.00. When you right-click a column name in a table, a context-sensitive menu shows up. Right-click the Amount column and select Sort Descending. This will list the highest values first.

2. Sort the table by two columns in ascending order: first by Date, then by InvoiceNum (hint: File menu). What purchaser name is listed first after the sort?

   The first purchaser after the sort is Vijay.

   - *With the purchases table showing, select File — Sort....*
   - *In the dialog that comes up, first sort by Date, then by InvoiceNum. Click OK.*

1.6 Filtering

Filtering is one of the most useful and powerful features of Picalo. Tables can be quickly filtered on any formula you can enter. If you click the 'magic wand' icon to the right of the filter box, a dialog will help you create a wildcard filter formula.

When a table is filtered, note that the record count (bottom-right corner) shows the number of records showing in the filter followed by the total number of records in the table. With the exception of saving tables (which always saves every record), all Picalo commands work only on the filtered portion.

Filtering is a great alternative to selecting (*Data — Select*. Selecting always creates a new table of matching records (and leaves the original table unchanged). Filtering shows only matching records within the same table. When the filter is cleared, all the records return.

1. Show only Adams records (hint: right-click a cell with Adam in it). How many records are now showing in this filtered view? What is the difference between == and = in Picalo?

2. Show all records with amounts greater than 50,000. How many records are now showing?

3. Open the filter wizard (*File — Filter* or magic wand icon). Use a star (*) to show only records with VendorCodes starting with the letter Q. How many records are now showing?

4. Again with the filter wizard, show only records with the letter Q anywhere in the VendorCode. How may records are now showing?

Answer Key: Filtering

1. Show only Adams records (hint: right-click a cell with Adam in it). How many records are now showing in this filtered view? What is the difference between == and = in Picalo?

   Adam has 2,274 records in the table. Right-click a cell with Adam's name in it. You'll see a number of quick filters. Select the one with the double equals (==). You should now see only Adam's records. The table record count (bottom-right corner) shows the filter as active. To release the filter, click the 'Release' button above the table cells. The double equals (==) tests whether two things are equal in Picalo. In this filter, we are looking for cells where the Purchaser column has Adam in it. The single equals (=) is useful to Picalo programmers only (it sets variables). Most users will always use the double equals, and most of the time Picalo will take care of this for you.

2. Show all records with amounts greater than 50,000. How many records are now showing?

   4,263 records show in the filtered table. Since no records have amounts exactly equal to $50,000, you can't use the right-click, quick filter trick. This one requires that you manually enter the formula into the filter box. However, you can use the quick filter to get you almost there.

   - *Ensure you've released the previous filter and are seeing all records.*
   - *Right-click any value in the Amount column and set a filter where Amount is greater than this value. For example, if you right-clicked the first row, you'll set a filter where "Amount ¿ 66005.00". You are almost there.*
   - *Release the filter by clicking the 'release' button above the table. Note that the previous formula stays in the filter box, and it is now editable.*
   - *Edit the formula to show for 'Amount ¿ 50000' and click the 'Filter' button again.*

3. Open the filter wizard (*File — Filter* or magic wand icon). Use a star (*) to show only records with VendorCodes starting with the letter Q. How many records are now showing?

175 records show in the filtered table. This task shows how to use the Filter dialog to help you create a wildcard match filter.

- *Ensure you've released the previous filter and are seeing all records.*
- *Open the Filter dialog (File — Filter).*
- *Select VendorCode for the column.*
- *Ensure 'Simple Wildcard' is selected. Regular expressions are much more powerful but are a topic for more advanced workbooks. ;)*
- *In the pattern box, enter 'Q*'. Click OK.*

4. Again with the filter wizard, show only records with the letter Q anywhere in the VendorCode. How may records are now showing?

   829 records show in the filtered table.

   - *Ensure you've released the previous filter and are seeing all records.*
   - *Open the Filter dialog (File — Filter).*
   - *Select VendorCode for the column.*
   - *The pattern can be entered in two ways:*
     (a) *Enter only a 'Q' into the pattern box. Then select 'Partial Match', which will make the pattern search for the Q anywhere in the cell value.*
     (b) *Enter '*Q*' in the pattern box. This uses the star wildcard to search for anything, a Q, then anything. Since the * matches empty text, this pattern finds the Q at the beginning or end of the cell as well as in the middle.*
   - *Click OK.*

Exercise 2

Import and Export

Goal: To learn how to get your data into and out of of Picalo.

Picalo can import many different formats, including delimited text, fixed width text, EBCDIC mainframe files, Excel spreadsheets, and email. In addition, Picalo scripts that use regular expressions can import almost any type of file when you use them in a script.

This exercise uses the EmpVendor dataset, which includes the following data:

- `Employee.tsv` - A tab-delimited text file containing employee records in our company. This has been exported from a (fake) corporate database and needs to be imported into Picalo.
- `Vendor.tsv` - A tab-delimited text file containing vendor records that we purchase from. This has been exported from a (fake) corporate database and needs to be imported into Picalo.
- `email_thread.txt` - A heavily sanitized thread of emails about what it takes to be a geek. This has been exported from a (fake) corporate email system and needs to be imported into Picalo.

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

2.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `EmpVendor`. How many initial tables are in this project?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `EmpVendor`. How many initial tables are in this project?

   There are no initial tables in this project (we need to import them). To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the EmpVendor directory.

2.2 Import Delimited Text

Delimited text files are arguably the most common and popular import/export format used today. When you need to transfer data from a database into Picalo, this is likely the format you'll use. In most organizations, data analysts request data from the IT department and receive it in delimited text format. The two most popular delimited types are as follows:

1. *Comma-Separated Values* - CSV files contain one record per row in the file, with field values separated by commas. If commas exist in the actual data, double-quotes are placed around the field. Some CSV files have headers in the first row, some do not.

2. *Tab-Separated Values* - TSV files are the same as CSV files, with the exception that tab characters are used in place of the commas. Some users prefer this format because tabs rarely occur in most data sets, so the double-quote exception is rarely needed.

Some users mistakenly try to find an extremely uncommon character to use as the delimiter, such as the tilde character (~). This is not necessary–the format has rules to fall back on when delimiter characters are found in the values (the double-quote thing). IT departments that use uncommon characters often do export using the fall-back rules (i.e. they take the easy way out and just expect the uncommon character doesn't exist in the dataset). This is a risky move because the moment you assume a tilde won't be in your dataset is the moment it will appear in one of the values. :) Be assured that Picalo follows the rules correctly for both import and export of delimited files.

1. Open Picalo and set the project directory to `EmpVendor`. How many initial tables are in this project?

Answer Key: Import Delimited Text

1. Open Picalo and set the project directory to `EmpVendor`. How many initial tables are in this project?

   There are no initial tables in this project (we need to import them). To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the EmpVendor directory.

1. Importing is done with *File — Import — Text File or Excel Spreadsheet*. Import the Employee.tsv file (hint: its delimited text with a tab character). Set the field types to types you feel appropriate. Save the table as a regular Picalo table. Are any fields not a string (text) type?

2. Repeat with the Vendor.tsv file. Are any fields not a string (text) type?

Answer Key: Import Delimited Text

1. Importing is done with *File — Import — Text File or Excel Spreadsheet*. Import the Employee.tsv file (hint: its delimited text with a tab character). Set the field types to types you feel appropriate. Save the table as a regular Picalo table. Are any fields not a string (text) type?

   All the fields in this table are String fields (see below for an explanation of the zip field). Importing is done with the Picalo Data Import Wizard. Start the wizard with File — Import — Text File or Excel Spreadsheet. Following are the pages of the wizard:

   (a) *Click the Select... button and locate the Employee.tsv file. Leave the file encoding to automatic, and accept the default table name (Employee). Click Next.*

   (b) *Picalo will automatically detect that this file is a delimited text file. Leave this setting and click Next.*

   (c) *On this next page, note that the data in the preview window are not correct. Picalo runs all the column headers together because it thinks the fields are delimited with commas. Select the Tab radio button. The preview window will immediately show the correct formats. Click Next.*

   (d) *The final page of the wizard allows you to set the field names and types. Most fields in this table are of type 'String', or text. The zip field could be set as an integer field (an integer type is a number field that contains whole numbers (no decimal point). However, since we're not doing any math with this column (like we would with an amount column), there's no reason to convert it to a number. All fields can remain String type. Click Finish to import the records into Picalo.*

2. Repeat with the Vendor.tsv file. Are any fields not a string (text) type?

   The import of the Vendor file is the same as with the Employee file. Simply repeat the process. As with the Employee table, all fields are String type. The employer_id_num and vendor_id fields must be strings because they contain dashes and other text characters.

2.3 Run Descriptives

Descriptives give you a summary view of a table. They are usually run immediately after loading a new table into Picalo. While descriptives do not usually find fraud, they provide a foundation view for all other analyses that are run.

1. Why do we run descriptives on newly-imported tables?

2. Open the Employee table and run descriptives on it (hint: Analyze — Descriptives). Ensure the correct data types and numbers of records exist in each field. Do you see any problems?

3. Use a filter to verify the Employee ID field (hint: exclude matching records). This field should have two numbers, dash, three numbers. Are there any records that do not have the right format? Which ones?

Answer Key: Run Descriptives

1. Why do we run descriptives on newly-imported tables?

   First, descriptives provide an overview of your dataset. They give a sense of maximum values, minimum values, and averages. Having this overview is important when running other analyses. Second, descriptives allow you to check imports from text files and databases. If you have the sender (usually an IT department) calculate the number of total records and field totals, you can verify these to ensure your import didn't lose records or data.

2. Open the Employee table and run descriptives on it (hint: Analyze — Descriptives). Ensure the correct data types and numbers of records exist in each field. Do you see any problems?

   Once you have the Employee table open, click Analyze — Descriptives. You'll get a new table. Note that the position_number field has 22 blank records. This may or may not be a problem, but it is something to take note of and verify against the source records.

3. Use a filter to verify the Employee ID field (hint: exclude matching records). This field should have two numbers, dash, three numbers. Are there any records that do not have the right format? Which ones?

   There are three employees with invalid employee_id numbers: 20-56, 1839x, and 30-3562. These may be indicative of foul play, or they may simply be data entry errors. To get these invalid ids, click on the magic wand icon (or select File — Filter...) to bring up the Filter Table dialog. In the pattern text box, enter '##-###', which is the correct pattern for the field. Select 'Exclude Matching Records' for the Direction. Click OK.

2.4 Export Delimited Text

Picalo can export to a number of formats, including delimited text and Microsoft Excel (classic .xls right now). This is useful when you need to move data from Picalo to another application for further analysis. Most data analysts use a variety of tools (of which Picalo is only one) to accomplish their tasks, so import and export is extremely important.

1. Export the Vendor table (hint: File — Export) as an Excel file. Open it in MS Excel. Did it work?

Answer Key: Export Delimited Text

1. Export the Vendor table (hint: File — Export) as an Excel file. Open it in MS Excel. Did it work?

   Open the Vendor table by double-clicking it in the left-side tree. Select File — Export.... In the Format box, select 'Microsoft Excel File'. Enter a name (note the directory it is going to!) and click OK. Now start Excel and load the file.

2.5 Copy and Paste

As an alternative to import and export, Picalo can copy data directly to and from the clipboard. Sometimes this is the easiest and fastest way to get data from one application to another.

1. Select the rows of all vendors that live in Chicago (hint: sorting can group them together, or filtering can show only those records). Select Edit — Copy to copy them to the clipboard. Open MS Excel and select Edit — Paste in an empty spreadsheet. How many records did you copy?

Answer Key: Copy and Paste

1. Select the rows of all vendors that live in Chicago (hint: sorting can group them together, or filtering can show only those records). Select Edit — Copy to copy them to the clipboard. Open MS Excel and select Edit — Paste in an empty spreadsheet. How many records did you copy?

   There are two vendors based in Chicago. The easiest way to find the ones in Chicago is to do a quick filter with the magic wand icon (or File — Filter...). In the Filter Table dialog, select 'city' for the Column and enter 'Chicago' for the pattern. Click OK. You'll see the two records. Select them with your mouse (an easy way is to click the top-left box in the table where the row and column headers meet). Once they are selected, select Edit — Copy to copy them to the clipboard. You can now paste them into any other application.

2.6 Import Email Messages

A common task in today's cases is to analyze electronic communication like email messages, instant message logs, and phone text messages. Picalo can import the two most common email formats today: MBox and Maildir.

MBox is an older standard. It keeps all messages for a person's mailbox in a single file, separated by hard returns and the 'From' keyword. The files are simple text files that import into Picalo easily. Many email clients use MBox internally to store email on the user's computer.

Maildir is the newer standard. It stores each message in a separate file, and it keeps folders in directories on the server and client. Picalo can parse these messages into an easy-to-use table.

Microsoft Outlook uses a proprietary format inside of .pst files. To import .pst files into Picalo, first locate a .pst to MBox converter. There are many available on the Internet, and they can be found with a simple search. Other email platforms, such as Lotus Notes, are similar to Outlook. Notes uses a .nsf file that you can transfer to MBox with any number of tools online.

Email messages come in two types: text and html. Text emails include the content only; they don't have any formatting, fonts, or colors. Html emails are formatted using HTML codes. Some messages contain both types in the same email. When Picalo imports email messages, it creates a table that contains a column for the text content and a column for the Html content.

1. These messages have already been exported from an email client. All of the messages are included in a single file called "email_thread.txt. Import the email messages into Picalo with *File — Import — Email* Use the Mbox format (first text box). How many messages are in HTML format (see the HtmlBody field)? Are all these emails related to the original mail? How do you know?

2. Filter the resulting table on records with "0x2A" in the TextBody field. How many messages contain this text?

Answer Key: Import Email Messages

1. These messages have already been exported from an email client. All of the messages are included in a single file called "email_thread.txt. Import the email messages into Picalo with *File — Import — Email* Use the Mbox format (first text box). How many messages are in HTML format (see the HtmlBody field)? Are all these emails related to the original mail? How do you know?

   Start by opening the Import Email wizard with File — Import — Email.... Click Next. This file is an MBox file, so click the first 'Choose...' button and locate the email_thread.txt file. Click Next. Give the new table a name (the default is fine), and click Finish. The emails will be imported into a table with 8 records. By looking in the Html-Body field, you'll see that 4 of them are formatted as html emails. All of these emails are part of a single thread (i.e. the users kept clicking reply all in their email programs). You can see this by looking in the 'InReplyTo' field, which contains the message id of the email the user hit reply on. Using this field, you can reconstruct the entire thread. If you are interested in the MBox format, you may want to load the email_thread.txt into a text program like Notepad or Microsoft Word.

2. Filter the resulting table on records with "0x2A" in the TextBody field. How many messages contain this text?

   Two email messages contain 0x2A. With the email table open, filter the table by clicking the magic wand icon (or File — Filter). In the Filter Table dialog, select 'TextBody' as the Column. The pattern is '0x2a' (zero, x, two, a). Because you don't know what case the matching text will be in, select 'Ignore Case' for the case. Also select 'Partial Value' for the match type (so the 0x2A can be anywhere in the field). Click OK.

Exercise 3

Databases

Goal: To learn how to connect to a database and import data into Picalo.

The best practices model of using Picalo is with Picalo as a front end to a relational database like Oracle, MySQL, PostgreSQL, SQL Server, MS Access, or another product. This lets the database do what it does best (store and retrieve data) and Picalo do what it does best (analyze).

Picalo can connect via ODBC (an open standard) to just about any database available today. All you need is the ODBC driver from your database vendor. In addition, Picalo can connect directly to several popular databases (bypassing ODBC for a more direct connection): Oracle, MySQL, and PostgreSQL.

Picalo includes a built-in relational database called SQLite. This stores your database in a single file on your machine (no server required) but still provides the power of SQL queries. SQLite is a good solution when you want relational databases for the small stuff, but you don't want to do overhead of a real database.

This exercise uses the ProcureDB dataset, which contains the records for a fictitious company. The records include bidding records (as part of a standard procurement process) and employee purchase card transactions. It includes the following data files:

- `database.sqlite3` - An SQLite database that contains 11 tables within this single file. The embedded tables are described as follows:
  - `Bid` - Contains the bids on different contracts from the contractors in the Contractor table.

- `BidDetail` - Contains the bid detail for each of the bids in the Bid table. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. This table contains the bid amounts on each line item listed in the ProjectItem table.
- `Blacklist` - Contains a number of contractors that have been blacklisted because of fraud or other problematic behavior.
- `Contractor` - Contains the records for the contractors who have bid on our projects.
- `Employee` - Contains the employee records for our company's employees.
- `Invoice` - Contains invoices for work done on projects by the winning bidder. After a contractor has won a bid, it starts work on the project. The invoices in this table are the bills for that work.
- `Project` - Contains 64 projects that the company has sent out for bids from different contractors. This can be seen as the master table of the entire database.
- `ProjectItem` - Contains the line items for projects. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. These line items are contained in this table.
- `PurchaseCard` - Contains purchase card information for our employees. Each employee is issued a credit card to make purchases with. This table is not really related to the project/bidding tables.
- `PurchaseTransaction` - Contains the actual transactions employees have made on their p-cards.
- `ValidNAICS` - Contains the valid places employees can use the p-cards at. Each NAICS code represents a type of business, and our employees can only spend money at certain types of businesses. Note that these NAICS codes are generated (they do not relate to real NAICS codes).

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

3.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `ProcureDB`. How many initial tables are in this project?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `ProcureDB`. How many initial tables are in this project?

   There are no initial tables in this project (we need to query them). To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the ProcureDB directory.

3.2 Create a database connection

1. Connect to our database with File — New Database Connection
   - Database type: SQLite
   - Filename: database.sqlite3
   - Connection name: conn
   - Filename: conn

2. Right-click the new database connection and select "Refresh. How many tables are in this database?

Answer Key: Create a database connection

1. Connect to our database with File — New Database Connection

   - Database type: SQLite
   - Filename: database.sqlite3
   - Connection name: conn
   - Filename: conn

   Follow the indicated steps to create your new database connection. It will be saved in your project directory (not the database or data, just the connection to it) so it appears the next time you open Picalo.

2. Right-click the new database connection and select "Refresh. How many tables are in this database?

   There are 11 tables that will show up. Right now, Picalo database connections do not refresh themselves automatically (this is being worked on). You need to right-click the new connection and select Refresh manually.

3.3 Copy to Picalo table

1. Copy the Contractor table to a Picalo table (hint: right-click it on the left-side list). What are the differences between the database version and the Picalo table version?

Answer Key: Copy to Picalo table

1. Copy the Contractor table to a Picalo table (hint: right-click it on the left-side list). What are the differences between the database version and the Picalo table version?

   To convert a database table to a Picalo table, right-click its name and select Copy To Picalo Table. A Picalo table contains all of the data locally. It can be used in any Picalo function. Its data can be changed. It saves to a .pco table. A database-based table stores its data on the server. Picalo reads these data in a just-in-time fashion (as you read them on the screen). Database tables are always read-only.

3.4 Create a simple query

1. Create a new query:

   - Select File — New Query
   - Set the "Query Name to ContractorSimple.
   - Double-click the Contractor table. It will show up on the right. Double-click a few of the fields listed in the Contractor table. They will be added to the query fields below.
   - On the "State field, set a filter to only pull those in VA.
   - Click Run Query to run the query.

   How many contractors are based in Virginia? How is this table of results different than a regular database table? If you ran the query in the future, would the results be "live (i.e. show any updates)?

Answer Key: Create a simple query

1. Create a new query:

   - Select File — New Query
   - Set the "Query Name to ContractorSimple.
   - Double-click the Contractor table. It will show up on the right. Double-click a few of the fields listed in the Contractor table. They will be added to the query fields below.
   - On the "State field, set a filter to only pull those in VA.
   - Click Run Query to run the query.

   How many contractors are based in Virginia? How is this table of results different than a regular database table? If you ran the query in the future, would the results be "live (i.e. show any updates)?

   There are 4 contractors in Virginia. The results would always be live and would include any new contractors added to the database. In this task, you ran your first query. Queries are ways to ask databases questions, and Picalo helps you create these queries with a visual interface. Many analysts also use MS Access to query databases because it has an incredible visual interface (and then they import into Picalo). In the Visual Query dialog, you select tables and fields to be included in the results. In this query, we type 'VA' into the 'Select records where this field equals' box (ensure the State is selected in the Query Fields area first!). This filters the records to only those in Virginia. Note that this is different than Picalo's built-in filter capabilities. Picalo's filtering is much more powerful than what you can do here, but the benefit of this query-based filtering is the database server does the work for you. Database servers are usually much faster and more powerful than end-user computers running Picalo. A best practices model is to do high-level filtering at the query level, then use Picalo's more detailed filtering on the query results.

3.5 Create a joined query

1. Create a query with more than one table:

   - Select File — New Query
   - Set the "Query Name to EmployeePurchaseTransaction.
   - Double-click the Employee, PurchaseCard, and PurchaseTransaction tables.
   - Join the tables together by dragging the common fields to one another. You will get lines denoting the join.
   - Add the fields from all three tables to the results, and run the query.

   Why does the employee name repeat so often, even though it was in the original tables only once?

Answer Key: Create a joined query

1. Create a query with more than one table:

   - Select File — New Query
   - Set the "Query Name to EmployeePurchaseTransaction.
   - Double-click the Employee, PurchaseCard, and PurchaseTransaction tables.
   - Join the tables together by dragging the common fields to one another. You will get lines denoting the join.
   - Add the fields from all three tables to the results, and run the query.

   Why does the employee name repeat so often, even though it was in the original tables only once?

   The employee name repeats many times because of the one-to-many relationship between the tables in this query. In other words, there is only one Employee record to many Purchase Transactions (one employee makes several transactions). By combining the tables back together, it repeats the values of the 'one-side' of the relationship. This task introduces the idea of joining tables together, which is a basic task done in most analyses. To join tables, simply drag the common fields to each other. You can also double-click the resulting join line to change the join type.

Exercise 4

Phantom Contractors

Goal: To learn how to use Picalo commands to find a type of fraud.

Now that you know the basics of Picalo tables, import and export, and database connections, let's explore the primary purpose of Picalo: analysis! In this exercise, you'll complete tasks related to finding phantom contractors. Use your creativity to solve these problems; there are usually multiple approaches to get to the answers.

This exercise uses the Procurement dataset, which includes the following data:

- `Bid` - Contains the bids on different contracts from the contractors in the Contractor table.
- `BidDetail` - Contains the bid detail for each of the bids in the Bid table. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. This table contains the bid amounts on each line item listed in the ProjectItem table.
- `Blacklist` - Contains a number of contractors that have been blacklisted because of fraud or other problematic behavior.
- `Contractor` - Contains the records for the contractors who have bid on our projects.
- `Employee` - Contains the employee records for our company's employees.

- `Invoice` - Contains invoices for work done on projects by the winning bidder. After a contractor has won a bid, it starts work on the project. The invoices in this table are the bills for that work.
- `Project` - Contains 64 projects that the company has sent out for bids from different contractors. This can be seen as the master table of the entire database.
- `ProjectItem` - Contains the line items for projects. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. These line items are contained in this table.
- `PurchaseCard` - Contains purchase card information for our employees. Each employee is issued a credit card to make purchases with. This table is not really related to the project/bidding tables.
- `PurchaseTransaction` - Contains the actual transactions employees have made on their p-cards.
- `ValidNAICS` - Contains the valid places employees can use the p-cards at. Each NAICS code represents a type of business, and our employees can only spend money at certain types of businesses. Note that these NAICS codes are generated (they do not relate to real NAICS codes).

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

4.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

   There are 11 tables in this project (they are the same ones we used in the database last exercise).

4.2 Address matching

When employees set themselves up as phantom contractors, they often list their home address so payment checks come to them. Others list P.O. Boxes, which real companies don't normally use. Contractors can also set up ghost companies to decrease competition, such as when all three bidders on a project are actually subs of the same parent company.

Address checking can be difficult because 50 N Maple is essentially the same as 50 North Maple Street. Doing an exact match (which is what database queries do) won't match these addresses. Some analysts modify the addresses in clever ways by removing all instances of North, South, East, West, Way, Circle, Street, Avenue, etc. Others remove all letters and compare only the street numbers.

Another way is to use fuzzy matching. Picalo supports both the Soundex algorithm and ngram-style comparisons for fuzzy matching. Both can be effective methods of finding duplicates.

1. Using the Data menu, join the employee and contractor tables to find if any have the same address and zip code (hint: use a fuzzy match join). What did you find?

2. Join the contractor table with itself to see if any have the same address and zip code. Are any the same?

Answer Key: Address matching

1. Using the Data menu, join the employee and contractor tables to find if any have the same address and zip code (hint: use a fuzzy match join). What did you find?

   There are three matches that are worrisome: 4399 North Possum Road and 5012 North Possum Road, 446 South First Way and 446 South First, and 7555 East Main Street and 7555 E Main Street. The fuzzy match algorithm can find these matches, but it takes a little experimentation with the match percent. At 30 percent, you'll likely find too many matches (it's too fuzzy for this analysis). At 90 percent, you'll likely find none. 60 percent turns out to be a good happy medium that finds the right records. Many analyses are like this: they require a little experimentation to get the thresholds right.

   (a) *Open the Employee and Contractor tables by double-clicking them.*

   (b) *Select Data — Join — By Fuzzy Match... to get the 'Join By Fuzzy Match' dialog.*

   (c) *Set Contractor as the first table and Employee as the second table.*

   (d) *Select Address as the first match criteria in both tables.*

   (e) *Select ZipCode as the second match criteria in both tables.*

   (f) *Set 60 percent as the fuzzy match percent. Give a results table name and click OK.*

2. Join the contractor table with itself to see if any have the same address and zip code. Are any the same?

   There are no matching contractors. This exercise simply showed that tables can be joined to themselves to find matching records.

4.3 Name matching

1. Create a new static calculated column in the employee table that represents the initials of each employee (hint: use FirstName[0] for the first character of first name). Join this new column with the contractor names using a Picalo expression that checks to see if any name has these initials (hint: the expression "Field1 in Field2 sees if the text in field 1 exists anywhere in field 2). What do the results mean? What further investigation should be done?

Answer Key: Name matching

1. Create a new static calculated column in the employee table that represents the initials of each employee (hint: use FirstName[0] for the first character of first name). Join this new column with the contractor names using a Picalo expression that checks to see if any name has these initials (hint: the expression "Field1 in Field2 sees if the text in field 1 exists anywhere in field 2). What do the results mean? What further investigation should be done?

   A bug in Picalo prevents this one right now. We're working on it.

4.4 Interesting addresses

1. Check for contractor addresses that are using PO Boxes. Normally, businesses have real addresses and not post office boxes. What did you find?

2. Check for zip codes that do not match the correct format for zip codes. What is the likely cause of any problems here? What should the follow-up be?

Answer Key: Interesting addresses

1. Check for contractor addresses that are using PO Boxes. Normally, businesses have real addresses and not post office boxes. What did you find?

   There are two contractors: EVL and RZK Partnerships. These may not be real contractors. These results can be found through Data — Select, Edit — Find, or by filtering the table. To use the filter method, click the magic wand icon to set a simple filter on the word 'Box'. Set it to match partial values and to ignore the case. A more robust solution is to use the detectlet (Tools — Detectlets) that is engineered to find post office boxes. It searches for dozens of different ways to write P. O. Box.

2. Check for zip codes that do not match the correct format for zip codes. What is the likely cause of any problems here? What should the follow-up be?

   There is one contractor with an invalid zip code: RPC Incorporated. To find this contractor, click the magic wand icon to set a filter on the table. The pattern is '#####-####' . Exclude the matching records. Click OK and you'll see the invalid company.

4.5 Interesting names

1. Are there any interesting names of contractors (hint: what would happen if a check were cut to a company named "Cash)? The perpetrator might have been tricky here, so be clever in your searching. Is there only one, or are there two companies of interest here? Have any payments been made to this company?

Answer Key: Interesting names

1. Are there any interesting names of contractors (hint: what would happen if a check were cut to a company named "Cash)? The perpetrator might have been tricky here, so be clever in your searching. Is there only one, or are there two companies of interest here? Have any payments been made to this company?

   There are two companies: Cash and The Cash Company. The first company, Cash, would have checks printed to cash, meaning anyone could cash them. The second one is likely a real company and not a problem. These results can be found through Data — Select, Edit — Find, or by filtering the table. To use the filter method, click the magic wand icon to set a simple filter on the word 'cash'. Set it to match partial values and to ignore the case.

Exercise 5

Purchasing Card Fraud

Goal: To learn how to use Picalo commands to find a type of fraud.

Many companies give their employees credit cards to make small purchases with. These cards save the company because the normal process (purchase orders, fulfillment, etc.) contains costly processes and paperwork. However, the cards provide more opportunity for employee fraud because employees don't need permission to use the card (most companies check the bill when it comes, but not before).

This exercise uses the Procurement dataset. We'll focus on the following four tables:

- `Employee` - Contains the employee records for our company's employees.
- `PurchaseCard` - Contains purchase card information for our employees. Each employee is issued a credit card to make purchases with. This table is not really related to the project/bidding tables.
- `PurchaseTransaction` - Contains the actual transactions employees have made on their p-cards.
- `ValidNAICS` - Contains the valid places employees can use the p-cards at. Each NAICS code represents a type of business, and our employees can only spend money at certain types of businesses. Note that these NAICS codes are generated (they do not relate to real NAICS codes).

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

5.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

   There are 11 initial tables, but we'll focus on just 4 in this exercise. To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the Procurement directory.

5.2 Unapproved NAICS codes

1. The ValidNAICS table holds the codes that employees can use on their purchase cards. Have any transactions been recorded that use unapproved codes?

Answer Key: Unapproved NAICS codes

1. The ValidNAICS table holds the codes that employees can use on their purchase cards. Have any transactions been recorded that use unapproved codes?

   There are three transactions: 4007, 4503, and 4765. These results can be found by joining tables together or by finding nonmatching. If you use the joining option (on the NAICSCode field), be sure show all the records of the PurchaseTransaction table. When the results come up, look for empty records in the ValidNAICS field portion. The nonmatching option is more direct: it specifically searches for nonmatches. To search using this option, perform the following steps:

   (a) *Open the ValidNAICS and PurchaseTransaction tables.*

   (b) *Select Analyze — Find — Nonmatching By Value.... Ensure PurchaseTransaction and ValidNAICS tables are listed as the first and second tables, respectively. Enter the NAICS Code for the match field. Enter a results table name. Click OK.*

   (c) *The resulting table list has two tables in it. The first table lists the nonmatching records from the first match table and the second table lists the nonmatching records from the second match table. You should see the three transactions in this table. If you did the tables backwards, click on the little spinner box in the top-right corner to go to the next table.*

5.3 Split purchases

Note that there are more complex ways to do this analysis. With a script, you could automatically check the date ranges to find charges at the end of one period and at the beginning of others. This would allow you to find these issues without manually inspecting the tables.

1. One way employees purchase above their limits is to split the cost across two transactions. Have any employees purchased products with two transactions just under their single transaction limit? What should the follow-up be?

Answer Key: Split purchases

1. One way employees purchase above their limits is to split the cost across two transactions. Have any employees purchased products with two transactions just under their single transaction limit? What should the follow-up be?

   This analysis can be accomplished in a number of ways. One way is to join and then filter:

   (a) Open the PurchaseCard and PurchaseTransaction tables.

   (b) Select Data — Join — By Value.... Select the PurchaseCard and PurchaseTransaction tables, and set them to join on the AccountNum field. Enter a results name. Click OK.

   (c) You now have a combined table with both records. Filter this new table on a formula that checks whether any transactions were just under the single transaction limit. To do this, enter the following formula into the Filter Expression box: 'Amount > SingleTransactionLimit - 10'. Click Set to show the records. You'll see 16 records.

   (d) Stratify the table to see the records for individual employees in individual employees. This makes it easier to see the problems. This will show 13 tables in a table list.

   (e) While viewing the new results table, use the spinner box at the top-right to go through the 13 employee tables. On the third table (table 2/13), you'll notice that the employee is just under the transaction limit (by one dollar) and the transaction date is at the end of January, then at the first of February. This transaction needs follow-up investigation to find out what the purchase was for and who purchased it.

5.4 Over limits

1. Have any employees spent over their card limit in a given month? Hint: split the date into multiple, calculated columns using field.year, field.month, and/or field.day.

Answer Key: Over limits

1. Have any employees spent over their card limit in a given month? Hint: split the date into multiple, calculated columns using field.year, field.month, and/or field.day.

   This one can't be done right now because of a bug in Picalo. We're working on it.

5.5 False billings

1. Using a Benfords analysis, see if, on average, any employee purchases seem to be too far from the expectation. Use only the first digit in the analysis. Are any purchasing cards consistently too different from the law?

Answer Key: False billings

1. Using a Benfords analysis, see if, on average, any employee purchases seem to be too far from the expectation. Use only the first digit in the analysis. Are any purchasing cards consistently too different from the law?

   The account numbers with the lowest percentages are 2038, 2058, 2082, and so forth. Their low Benford's expected number indicates the numbers do not match Benford's Law, and they may be generated (i.e. not real transactions). The highest percentage a row can get is 0.30 (ones in the first digit of all transactions). The lower the percentage, the lower the chance the numbers are real. Further investigation should be done on the top records. The process for achieving these results is as follows:

   (a) *Open the PurchaseTransaction table.*

   (b) *Select Analyze — Digital Analysis — Append Benford's Expected Col. The column to analyze is 'Amount', and the number of significant digits is 1. Call your new column 'AmountB'. Click OK.*

   (c) *Note the new AmountB column next to the Amount column. We'll average the table by this column.*

   (d) *Select Data — Summarize — By Unique Column Value.... Select the PurchaseTransaction table and check the AccountNum column. Add summarizing expression by calculating Average on AmountB. Click Add. Call this column AvgB. Add a results table name and click OK.*

   (e) *This new table has a Benford's average for each account (i.e. employee). Right-click the AvgB column header and sort ascending.*

5.6 Transaction ID problems

1. Are there any duplicate Transaction IDs? These may have been double-paid. Hint: Use the Analyze — Find menu.

2. Are there any missing Transaction IDs? These may have been lost in the process, or they may have been removed intentionally by an employee.

Answer Key: Transaction ID problems

1. Are there any duplicate Transaction IDs? These may have been double-paid. Hint: Use the Analyze — Find menu.

   Transaction 4001 has two records (it's the first one there!). Note the same data on the entire record, which indicates it is likely a duplicate. Duplicate transactions are often double-paid. They are not usually fraud, but they are problems that need to be cleared up. To find duplicates, perform the following steps:

   (a) *Open the PurchaseTransaction table.*

   (b) *Select Analyze — Find — Duplicates.... Select the TransactionID column. Click OK.*

   (c) *The resulting table shows the duplicates.*

2. Are there any missing Transaction IDs? These may have been lost in the process, or they may have been removed intentionally by an employee.

   Transaction ID 4003 is missing. Missing transactions are important when looking at receiving reports (where we had a purchase order paid but never received product). To find missing records, perform the following steps:

   (a) *Open the PurchaseTransaction table.*

   (b) *Select Analyze — Find — Gaps.... Select the TransactionID column. Click OK.*

   (c) *The resulting table shows that rows 0 and 2 have sequence problems. Row 0 is the duplicate payment we found in the last task. Row 2 is where transaction ID 4003 should be.*

5.7 Total purchases

1. Summarize all the purchases for employees during the year 2009. (Hint: Use Data — Summarize menu.)

   (a) Which employee has the highest number of purchases during the year?

   (b) Which employee has spent the most money this year?

   (c) Using a z-score, how far above the other employees is our big spender for 2009? What is his/her z-score on total purchase amount? Is it worth additional investigation?

Answer Key: Total purchases

1. Summarize all the purchases for employees during the year 2009. (Hint: Use Data — Summarize menu.)

   (a) Which employee has the highest number of purchases during the year?

   (b) Which employee has spent the most money this year?

   (c) Using a z-score, how far above the other employees is our big spender for 2009? What is his/her z-score on total purchase amount? Is it worth additional investigation?

   Since you are being asked to find employees, join the PurchaseCard and PurchaseTransaction tables together. This will allow you to have the employee numbers in all the results. To do this, select Data — Join — By Value.... The match criteria is the AccountNum field. Call your results table 'joined'. Once you have this table, perform the following steps for each question on the new joined table:

   (a) For the highest number of purchases, select Data — Summarize — By Unique Column Values.... Summarize on the EmployeeID column. Calculate a count on Amount. Click Add and give the new column a name. Enter a results name and click OK. Sort the results table in descending order by clicking on the new field header. The first record shows that Employee 1003 had 81 purchases.

   (b) For the highest spender, select Data — Summarize — By Unique Column Values.... Summarize on the EmployeeID column. Calculate a sum on Amount. Click Add and give the new column a name. Enter a results name and click OK. Sort the results table in descending order by clicking on the new field header. The first record shows that Employee 1004 spent $217,258.00.

   (c) Using the results table from the last item, add a new z-score column on the total field. With the table showing, click Analyze — Outliers — Add ZScore Column.... Select the sum column name, and enter a new column name. The z-score for employee 1004 is 3.10 (as is employee 1021!). Statistically, there is less than a 0.02 percent chance that a z-score will be above 3, so these records certainly warrant more investigation. This example shows the usefulness of

the z-score calculation. It shows how extreme the small or large values are.

Exercise 6

Procurement Fraud

Goal: To learn how to use Picalo commands to find a type of fraud.

This exercise walks you through a number of common procurement frauds. Procurement is the process of allowing multiple contractors to bid on a project. After any nonqualified contractors are thrown out (meaning anyone left is capable of doing the project), most organizations simply select the lowest bidder. Bidders are not allowed to see each others' information during the process, and they are not allowed to work together to fix prices.

This exercise uses the Procurement dataset, which includes the following data:

- `Bid` - Contains the bids on different contracts from the contractors in the Contractor table.
- `BidDetail` - Contains the bid detail for each of the bids in the Bid table. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. This table contains the bid amounts on each line item listed in the ProjectItem table.
- `Blacklist` - Contains a number of contractors that have been blacklisted because of fraud or other problematic behavior.
- `Contractor` - Contains the records for the contractors who have bid on our projects.

- `Employee` - Contains the employee records for our company's employees.
- `Invoice` - Contains invoices for work done on projects by the winning bidder. After a contractor has won a bid, it starts work on the project. The invoices in this table are the bills for that work.
- `Project` - Contains 64 projects that the company has sent out for bids from different contractors. This can be seen as the master table of the entire database.
- `ProjectItem` - Contains the line items for projects. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. These line items are contained in this table.
- `PurchaseCard` - Contains purchase card information for our employees. Each employee is issued a credit card to make purchases with. This table is not really related to the project/bidding tables.
- `PurchaseTransaction` - Contains the actual transactions employees have made on their p-cards.
- `ValidNAICS` - Contains the valid places employees can use the p-cards at. Each NAICS code represents a type of business, and our employees can only spend money at certain types of businesses. Note that these NAICS codes are generated (they do not relate to real NAICS codes).

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

6.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

   There are 11 tables in this project. To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the Procurement directory.

6.2 Blacklisted contractors

Note that there is a detectlet for this analysis that does everything automatically.

1. Are there any blacklisted contractors that have won projects?

Answer Key: Blacklisted contractors

1. Are there any blacklisted contractors that have won projects?

   Contractor 8080 won Project 9061, even though it was blacklisted. There are several ways to accomplish this solution. One method is to use matching:

   (a) *Open the Bid, Blacklist, and Project tables.*

   (b) *You first need to join the Project and Bid tables so you have access to the contractor ID for the winning bids. Select Data — Join — By Value.... Join on the BidID and AwardBidID (the bid that won the project). Call your new table 'Winner'.*

   (c) *Select Analyze — Find — Matching By Value.... Select the Winner and Blacklist tables. Match on the ContractorID.*

   (d) *The first table shows Contractor 8080 which won Project 9061. This contractor was blacklisted on November 5 for fraudulent practices, and the bid was awarded after this date. The second table in the list (click the little spinner icon at the top right to get to it) shows the matching record from the Blacklist table. This is not needed for this analysis.*

6.3 Winner not the lowest

The lowest bid is almost always the winning bid in most procurement practices. Having a bidder not be the lowest may indicate some kind of collusion going on. Note that there is a detectlet for this analysis that compares by direct winner ID rather than by amount.

1. Look for projects where the winner was not the lowest bidder. Are there any matches?

Answer Key: Winner not the lowest

1. Look for projects where the winner was not the lowest bidder. Are there any matches?

   Project 9019 went to a bidder that was not lowest (in fact, it was much higher than the lowest). This analysis can be done in a number of ways. A simple way to accomplish the task is to focus on the lowest amount. The first step, therefore, is to figure out what the lowest price on each project was. Then we'll join this with the project table and see if any don't match.

   (a) *Open the Bid and Project tables.*

   (b) *Select Data — Summarize — By Unique Column Values.... Check the ProjectID column. Calculate the Min of the TotalAmount column. Click Add, and enter 'MinAmount' for the new column name. Call the new table 'MinAmounts'. Click OK.*

   (c) *Select Analyze — Find — Nonmatching By Value.... Select Project for the first table and MinAmounts as the second table. Join on the MinAmount and ActualValue columns.*

   (d) *Project 9019 comes up as not being the lowest.*

6.4 Bidders that are too high or low

This task shows the importance of stratification. When you stratify in Picalo, the resulting table list contains a number of tables, one for each value in the stratification column. Any Picalo functions you do on the table list will be done per table. So the z-score calculation values (like the mean and standard deviation) will be calculated on each table in the list rather than on the aggregate values. This is a quick way to calculate across many different tables with a single menu selection.

1. Add a z-score column to the bids table for the TotalAmount column. Note that doing a z-score across all the bids for all projects doesnt make sense. You have to do it per project. Hint: use stratification first, do the z-score on the table list, then recombine. Sort the table by this z-score.

   (a) Do any bids seem unnaturally low?

   (b) Do any bids seem unnaturally high?

   (c) What would the next step in the analysis be?

Answer Key: Bidders that are too high or low

1. Add a z-score column to the bids table for the TotalAmount column. Note that doing a z-score across all the bids for all projects doesnt make sense. You have to do it per project. Hint: use stratification first, do the z-score on the table list, then recombine. Sort the table by this z-score.

   (a) Do any bids seem unnaturally low?

   (b) Do any bids seem unnaturally high?

   (c) What would the next step in the analysis be?

   No bids seem to be too low or high. The first step in this analysis is to stratify by project, then perform the z-score, then recombine again into one table. This is done as follows:

   - *Open the Bid table.*
   - *Select Data — Stratify — By Unique Values.... Check the ProjectID column, and call the results table 'Strat'.*
   - *Select Analyze — Outliers — Add ZScore Column.... Select the Strat table and TotalAmount column. Call the new column 'TotalZ'.*
   - *Now that the z-score is calculated correctly for each individual table, recombine using Data — Stratify — Combine Table List.... Call the new table 'Combined'.*
   - *Sort the combined table by TotalZ in descending order.*

   Using the Combined table, we can answer the task questions:

   (a) The lowest z-score is bid 5075 with a z-score of -1.5. This is getting close to the range that may be too low, but it is not so low as to warrant immediate investigation.

   (b) The most severe amount is bid 5061 with a z-score of 1.5. This is getting close to the range that may be too high, but it is not so high as to warrant immediate investigation.

   (c) Because no values were below -2 or above 2, investigation is likely not warranted. This obviously depends upon your personal thresholds for when to start investigations.

Note that in this case the z-score calculation provides useful information that will help direct the analyst, but since most projects only have 3-5 bidders, the results should not be taken as statistically significant. Use the results only as a guide.

6.5 Bids after the deadline

This task shows the usefulness of joining by expression. The normal *join by value* dialog only finds records with exact matches. In this case, you need to find records where one field is greater than another. You can use a simple Picalo formula to find any bids accepted after the deadline. The resulting table shows two bids: 5001 and 5056. Both were received after the deadline.

1. Were any bids received after the deadline? Did any of these bidders win?

Answer Key: Bids after the deadline

1. Were any bids received after the deadline? Did any of these bidders win?

   Bids 5001 and 5056 were accepted after the deadline. The process is as follows:

   (a) *Open the Bid and Project tables.*

   (b) *Select Data — Join — By Expression.... Select Bid as the first table and Project as the second table.*

   (c) *Picalo will now compare the records from the two tables, one by one. It uses 'record1' to denote the record from the first table and 'record2' to denote the record from the second table during each comparison. The formula that compares each record set is therefore* `record1.ProjectID == record2.ProjectID and record1.ReceiveDate > record2.BidEndDate`. *The first comparison ensures that the project ID is the same between the two tables and the second comparison include only records where the receive date of the bid is greater than the bid period ending date.*

6.6 Last bidder winner

A contractor who repeatedly submits the last bid and who wins bids may be getting inside information. Normally, the contractors have no idea what their competitors are bidding on the project. This provides incentive for each to bid at the lowest possible price. A bidder who is often last and is lowest indicates that someone inside the bidding process may be feeding information to it.

1. For each contractor, count the number of times it has been the winner of a bid AND the last bid to come in. Do any contractors worry you? Is there anything else fishy about the results?

Answer Key: Last bidder winner

1. For each contractor, count the number of times it has been the winner of a bid AND the last bid to come in. Do any contractors worry you? Is there anything else fishy about the results?

   (a) *Open the Project and Bid tables.*

   (b) *Join the two tables together using Data — Join — By Value.... Select the ProjectID as the matching column. Call the results table 'Combined'.*

   (c) *Figure out the day that the last bid came in on each project. Select Data — Summarize — By Unique Column Values.... Select the Bid table, and check the ProjectID column. Calculate the Max on ReceiveDate and name the new column 'LastBidDate'. Call the results table 'LastBid'. Click OK.*

   (d) *Join the LastBid and Combined tables together. Select Data — Summarize — By Unique Column Values.... Select the ProjectID as the matching column. Enter a results table name and click OK.*

   The remainder of this task can't be done due to a Picalo bug in summarizing by dates. We're working on it. Note that there is also a detectlet which does the process automatically.

6.7 Bids on the same day

1. Did any projects have all the bids come in on the same day? What might this mean? What other analysis techniques should be done?

Answer Key: Bids on the same day

1. Did any projects have all the bids come in on the same day? What might this mean? What other analysis techniques should be done?

   Coming soon...

6.8 Project values over budgeted values

1. The ActualValue field in the Project table records the actual agreed-upon amount once a winner is selected. This value is likely not the same as the budgeted amount for the project. Do any of the differences between the actual and budgeted amount worry you?

Answer Key: Project values over budgeted values

1. The ActualValue field in the Project table records the actual agreed-upon amount once a winner is selected. This value is likely not the same as the budgeted amount for the project. Do any of the differences between the actual and budgeted amount worry you?

   Coming soon...

6.9 Payments exceeding agreed value

1. Coming soon...

Answer Key: Payments exceeding agreed value

1. Coming soon...

6.10 Bills from incorrect contractors on a project

1. Have any invoices been received from contractors who were not the winner on a project? What might this mean? Could a reasonable explanation exist?

Answer Key: Bills from incorrect contractors on a project

1. Have any invoices been received from contractors who were not the winner on a project? What might this mean? Could a reasonable explanation exist?

   Coming soon...

Exercise 7

Scripting

Goal: To learn basic scripting principles like variables, for loops, while loops, if statements, and lists.

This exercise introduces you to scripting. Picalo includes a very powerful, yet easy language called Python. Python is a general purpose programming language used all over the world. It is object-oriented, strongly-typed, and robust. In fact, Picalo itself is mostly written in Python (a few of the most important libraries are in C).

While working with Picalo's menu's is fast and easy, scripting opens a new world of possibilities to analysts. Take the time to learn Python. You'll be several times more productive, and you'll be able to accomplish tasks not possible otherwise.

In this chapter's examples, you'll see the pound character used to denote comment lines, which are ignored by Picalo. These lines are for your information and are not part of the running script. The following are examples:

```
1 # this line is ignored
2
3 # as is this line
```

This exercise uses the Procurement dataset, which includes the following data:

- `Bid` - Contains the bids on different contracts from the contractors in the Contractor table.
- `BidDetail` - Contains the bid detail for each of the bids in the Bid table. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete,

labor, etc. This table contains the bid amounts on each line item listed in the ProjectItem table.

- `Blacklist` - Contains a number of contractors that have been blacklisted because of fraud or other problematic behavior.
- `Contractor` - Contains the records for the contractors who have bid on our projects.
- `Employee` - Contains the employee records for our company's employees.
- `Invoice` - Contains invoices for work done on projects by the winning bidder. After a contractor has won a bid, it starts work on the project. The invoices in this table are the bills for that work.
- `Project` - Contains 64 projects that the company has sent out for bids from different contractors. This can be seen as the master table of the entire database.
- `ProjectItem` - Contains the line items for projects. Each contract has a number of items that need to be bid on. For example, a bridge project would include line items for concrete, labor, etc. These line items are contained in this table.
- `PurchaseCard` - Contains purchase card information for our employees. Each employee is issued a credit card to make purchases with. This table is not really related to the project/bidding tables.
- `PurchaseTransaction` - Contains the actual transactions employees have made on their p-cards.
- `ValidNAICS` - Contains the valid places employees can use the p-cards at. Each NAICS code represents a type of business, and our employees can only spend money at certain types of businesses. Note that these NAICS codes are generated (they do not relate to real NAICS codes).

Please ensure you have copied this dataset to your disk before starting. Windows users can copy it to their My Documents folder.

7.1 Set the Project Directory

All of your work in Picalo is contained in a project, which corresponds to a directory (a.k.a. folder) on your computer. The left-side list will show all tables, database connections, queries, and scripts in this directory.

Picalo will create a file called `Picalo.history` in the directory to keep track of everything you do in the project. You can find this log in the *History* tab at the bottom of the screen. Picalo keeps the history log forever (unless you clear it), so even your activities in past sessions are available.

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

Answer Key: Set the Project Directory

1. Open Picalo and set the project directory to `Procurement`. How many initial tables are in this project?

   There are no initial tables in this project (we need to import them). To set the project directory, select File — Set Project Directory.... Use the dialog to navigate to the Workbook sample datasets and select the Procurement directory.

7.2 Using the Picalo Shell

The Picalo Shell is located in the first tab at the bottom-right corner of the screen. It can be used as a test area for scripting commands. You've watched Picalo issue commands in the Shell as you use menus. Now try typing a few commands yourself.

Using the Shell, calculate the following math problems:

1. Calculate $2 + 2$

2. Calculate $3 + 10.0/71.0$

3. Calculate $(3 + 10.0)/71.0$

4. Calculate $3 + 10/71$

Answer Key: Using the Picalo Shell

1. Calculate $2 + 2$

   In the shell, enter $2 + 2$. *Picalo responds with* 4.

2. Calculate $3 + 10.0/71.0$

   In the shell, enter $3 + 10.0/71.0$. *Picalo responds with* 3.14. *Note that mathematical order is correctly observed: multiply and division are done before addition and subtraction.*

3. Calculate $(3 + 10.0)/71.0$

   In the shell, enter $(3 + 10.0)/71.0$. *Picalo responds with* 0.18. *The use of parentheses in the equation overrides the normal mathematical order.*

4. Calculate $3 + 10/71$

   In the shell, enter $3 + 10/71$. *Picalo responds with* 3, *which may surprise you. The reason for the difference is we left off the decimal parts to the numbers this time, which makes Picalo use only integers. Therefore, since 10 divided by 71 is 0 remainder 10, that part of the equation evaluates to 0. The remainder (decimal part) is always thrown away in integer math. Always be sure to specify decimal numbers when you need them (by using 5.0 instead of just 5).*

7.3 Strings

Strings are any type of text, including letters, numbers, special characters, or anything else. Strings are denoted with either single quotes or double quotes. Python doesn't care which you use, as long as you are consistent on each side of the string.

1. Assign the string 'Hello World' to a variable named 'st'. Print this string back to the shell.

2. Print the fifth letter of `st`.

3. Print the last letter of `st`.

4. Print the fourth through eighth letters of `st`.

5. Print the last five characters of `st`.

6. Replace all instances of the letter 'l' with the number 5 in `st`.

7. Convert the number 4 into a string.

Answer Key: Strings

1. Assign the string 'Hello World' to a variable named 'st'. Print this string back to the shell.

   In the shell, enter: `st = "Hello World"`. *Then enter* `st` *by itself to see what the variable is set to. Once you set a variable equal to a value (a string, number, table, row, etc.), you can use the variable in place of the actual string. Think of a variable as a mailbox where you can place something.*

2. Print the fifth letter of `st`.

   In the shell, enter: `st[4]`. *Picalo responds with the letter* `o`. *Square brackets can be used to access parts of a string. Specify the character you want by index, with the first character at index 0 (not 1). So* `st[0]` *is* `H`, `st[1]` *is* `e`, *and so forth.*

3. Print the last letter of `st`.

   This can be done two ways. First, you can enter `st[10]` *as you probably expect. An easier way is to use a negative index, which starts from the right-side and works backwards. So* `st[-1]` *is* `d`, `st[-2]` *is* `l`, *and so forth.*

4. Print the fourth through eighth letters of `st`.

   Use a colon in the brackets to specify more than one character: `st[3:7]` *returns* `lo Wo`. *Note that we're still using zero-based indices.*

5. Print the last five characters of `st`.

   In the shell, enter: `st[-5:]`. *Picalo responds with* `World`. *If you leave the first index off, Picalo assumes you want to start at the first character. If you leave the last index off, Picalo assumes you want to end at the last character. If you leave both off (*`st[:]`*), you'll get the entire string!*

6. Replace all instances of the letter 'l' with the number 5 in `st`.

   In the shell, enter: `st.replace('l', '5')`. *This returns a new string with all letters replaced. Note that this does not modify* `st` *itself. To modify* `st`, *enter* `st = st.replace('l', '5')`. *This task shows one*

of many methods that strings have. Other interesting methods include the strip (removes spaces from either side), lower (turns everything lowercase), upper (turns everything uppercase), join (joins a list together), split (splits a string into a list), index (returns the index position of a character), and swapcase (switches the case). These are only a few of the available methods. See the Picalo advanced manual or Python documentation for more information.

7. Convert the number 4 into a string.

   In the shell, enter: `str(4)` *(note the lack of quotes). You can turn anything into a string by wrapping it in the* `str()` *function. The most important conversion functions are:*

   - `str()` *- Converts anything to a string.*
   - `int()` *- Converts decimal numbers or strings to integers.*
   - `number()` *- Converts integers or strings to decimal numbers.*
   - `Date()` *- Converts a string to a Picalo Date object.*
   - `DateTime()` *- Converts a string to a Picalo DateTime object.*
   - `boolean()` *- Converts anything to a boolean value. The number 1, string '1', string 'True' are all true. Anything else is false.*

   For more information about these conversions and data types, see the Picalo advanced manual.

7.4 Lists and Loops

Lists are a basic data structure of Python; just about everything revolves around them. A list is simply a group of ordered items. For example, a Picalo table is a list of rows, a column is a list of cell values, a row is a list of cell values, and even strings (text) can be used as a list of characters (see the previous task).

To create a list, simply set a variable equal to a comma-delimited set of items surrounded by square brackets. The following are examples of lists:

```
1 # create a list of numbers
2 nums = [ 1, 2, 3, 4, 5 ]
3
4 # create a list of strings
5 names = [ 'Bart', 'Maggie', 'Lisa', 'Marge', 'Homer' ]
```

1. Create a list containing the days of the week. Call your list `days`.

2. Using a for loop, print the numbers 0-9 to the shell.

3. Using a for loop, print the days of the week you entered earlier.

4. Create a list of the numbers 0-999 (hint: don't do this by hand, use the range() function).

5. Open the ValidNAICS table. Using a for loop, add a zero to the end of each code in the table. Then close the table without saving.

Answer Key: Lists and Loops

1. Create a list containing the days of the week. Call your list `days`.

   In the shell, enter: `days = [ 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat' ]`

2. Using a for loop, print the numbers 0-9 to the shell.

   A for loop repeats one or more lines of code for each item in a list. So, to go through the requested numbers, enter:

   ```
   for n in [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]:
       print n
   ```

   Press enter twice at the end to run the loop. Note the following about this code:

   - *The '`n`' variable can be any name. This is what will be set to each item of the list as the loop repeats. The first time it runs, n=0; then n=1; then n=2; and so forth.*
   - *The list must be followed by a colon, then an indented set of statements below it.*
   - *The `print n` line is indented with four spaces. The shell will even help you with this part. This tells Python which of the lines following the for loop are part of the repeated section. If you don't indent (or don't indent consistently with the same number of spaces), Python will get confused and will give you an error.*
   - *The `print` statement prints things to the Shell. You can simply enter a variable name when it's all you are typing, but you need to use `print` when you are inside of a loop or larger script.*

3. Using a for loop, print the days of the week you entered earlier.

   In the shell, enter:

   ```
   for day in days:
       print day
   ```

The only difference between this task and the last one is we used a variable name for our list rather than specifying the list right in the for loop line.

4. Create a list of the numbers 0-999 (hint: don't do this by hand, use the range() function).

   Sometimes you need an ordered list of numbers. The range function makes it easy to create a huge list of numbers. In the shell, enter: `range(1000)`. *You'll get a list of 1000 numbers, starting at 0 and ending with 999. If you want to set this list equal to a variable, use:* `nums = range(1000)`.

5. Open the ValidNAICS table. Using a for loop, add a zero to the end of each code in the table. Then close the table without saving.

   Since tables are simply lists of records, you can go through them with a for loop just like the above examples. In fact, this pattern is probably the most useful and common things done in Picalo scripts. To go through the table, first open the table by double-clicking on it. If you look in the shell, you'll see that Picalo has simply set a variable called `ValidNAICS` *to the table object. Tables are variables too! To go through the table, enter:*

```
for record in ValidNAICS:
    record.NAICSCode = record.NAICSCode * 10
```

   Press enter twice at the end to run the loop. The variable '`record`*' will be set repeatedly to each record in the table. With this record object, use the dot notation to access its cells. Watch the table closely as you hit enter–the loop runs quickly!*

7.5 Sorting a list

You already know that Picalo can sort tables in several ways. It can actually sort any list, even if the list is not a table.

1. Recreate the list of days of the week. Call your list '`days`'.

2. Sort this list alphabetically using the list.sort() command.

3. Print the sorted list.

Answer Key: Sorting a list

1. Recreate the list of days of the week. Call your list '`days`'.

   In the shell, enter: `days = [ 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat' ]`

2. Sort this list alphabetically using the list.sort() command.

   In the shell, enter: `days.sort()`. *When you type* `days` *again, you'll see the list has been sorted alphabetically. The sort method is very powerful; search the web for the 'python sorting howto' if you want to dive deeper.*

3. Print the sorted list.

   In the shell, enter `days`.

7.6 Create a table of random numbers

This task is the first time we'll use a saved script to execute code rather than the shell. The shell is useful for little commands, but scripts are normally the way to run a sequence of steps.

To start a script, select *File — New Script...*. You'll get a code file with two import statements at the top. Leave these statements there and start your code on line 4. Save this file before continuing (Picalo won't let you run a non-saved file).

When you are ready to run your code, click the blue triangle on the toolbar. You'll see the results of your script in the Script Output tab below.

1. Start a script in Picalo. Create a table called 'data' with the following fields (using the script):

   - ID: Integer int
   - RandomNum: int

2. Using the random library and a for loop, generate 100 random numbers between 1 and 1000. Hint: use random.randint(1, 1000) to create a random number. Add these numbers to the data table created above.

Answer Key: Create a table of random numbers

1. Start a script in Picalo. Create a table called 'data' with the following fields (using the script):

   - ID: Integer int
   - RandomNum: int

   The easiest way to see the code it takes to create a table is to create one using the menus in Picalo (File — New Table...). Then copy the code from the shell into your script. The following is the generated code (I've added hard returns to make it prettier, which you can put in your script as well):

   ```
   data = Table([
         ( "ID", int ),
         ( "RandomNum", int )
   ])
   ```

   When you run this script (little blue triangle on the toolbar), you'll see timestamps for the start and end times in the Script Output below. You should also have an empty 'data' table in the notebook tabs. Note that Picalo assumes you know what you are doing with scripts. If you add data to the table and then rerun the script, Picalo will replace the table without asking any questions.

2. Using the random library and a for loop, generate 100 random numbers between 1 and 1000. Hint: use random.randint(1, 1000) to create a random number. Add these numbers to the data table created above.

   The range command makes this easy:

   ```
   for i in range(100):
         num = random.randint(1, 1000)
         data.append([i, num])
   ```

Add the above code to your program just after the table gets created. The first line sends the variable `i` *through a list of numbers 0-99 (which makes the loop run 100 times). The second line generates a random number between 1 and 1000 using the random library. The third line appends the index and random number to the data table.*

7.7 Modify a field inline

As you saw in the above tasks, for loops are used to go through tables and analyze or modify each record. Let's explore this a little further using the Employee table. Open this table so you can use it in your script.

1. Using a for loop, increment the EmployeeID field by 1.

2. Using a second for loop, remove any of the following terms from the Address field: North, South, East, West, Avenue, Street, Circle, Road.

Answer Key: Modify a field inline

1. Using a for loop, increment the EmployeeID field by 1.

   Add the following to your script:

   ```
   for rec in Employee:
       rec.EmployeeID = rec.Employee + 1
   ```

   When you run the script, it will adjust each employee record up one.

2. Using a second for loop, remove any of the following terms from the Address field: North, South, East, West, Avenue, Street, Circle, Road.

   This one is a little harder because you need to search the Address field for any of the given terms. The string replace method is one trick that makes this easy. Since the address field contains a string, we can use any of the string methods on it. In the code below, the Address field is searched for each term. If it is found, it is replaced with the empty string. Note that I'm not just looking for the word itself, but also a space before each word. Since the field has a space before and after the word, just removing the word would leave two spaces. Taking out the first space with the word prevents the double space.

   ```
   for rec in Employee:
       rec.Address = rec.Address.replace(' North', '')
       rec.Address = rec.Address.replace(' South', '')
       rec.Address = rec.Address.replace(' East', '')
       rec.Address = rec.Address.replace(' West', '')
       rec.Address = rec.Address.replace(' Avenue', '')
       rec.Address = rec.Address.replace(' Street', '')
       rec.Address = rec.Address.replace(' Circle', '')
       rec.Address = rec.Address.replace(' Road', '')
   ```

   Since the return value of the replace method is also a string, we can even chain the replaces together.

   ```
   for rec in Employee:
   ```

```
     rec.Address = rec.Address.replace(' North', '').replace('
South', '').replace(' East', '').Address.replace(' West',
'').Address.replace(' Avenue', '').Address.replace(' Street',
'').Address.replace(' Circle', '').Address.replace(' Road',
'')
```

The following is a third option, which is the most complex because of two embedded for loops. For each record in the table, it runs the w variable through all the words and replaces each one in turn. To an experienced reader, it is the prettiest.

```
words = [ 'North', 'South', 'East', 'West', 'Avenue', 'Street',
'Circle', 'Road' ]

for rec in Employee:

     for w in words:

          rec.Address = rec.Address.replace(' ' + w, '')
```

7.8 Split a string into a list

Strings can be treated as lists of characters. They can also be split into a list of words using the .split() method. This allows you to work with the words one by one.

For this task, open the Contractor table.

1. Using a for-loop, switch the contractor name to be just the first word (usually the first three characters). Use the string.split() command to split it into a list.

Answer Key: Split a string into a list

1. Using a for-loop, switch the contractor name to be just the first word (usually the first three characters). Use the string.split() command to split it into a list.

   Since the contractor names always seem to start with three letters, we could simply us the bracket notation to get the first three characters of the string: `rec.Name = rec.Name[0:3]`. *However, this method is less effective because it breaks as soon as we get a contractor that doesn't follow the pattern. A more robust method is to split the contractor name by spaces, then take the first item in the list. The string split method splits on spaces by default, but you can even make it split on other characters. We'll just use the default.*

   ```
   for rec in Contractor:
       parts = rec.Name.split()
       rec.Name = parts[0]
   ```

   The first line sends the `rec` *variable through the table, record by record. The second line splits the name by spaces. The first record (RSD Partnership) results in a list of two items: ['RSD', 'Partnership']. The third line grabs the first item, 'RSD', and sets it to the record's Name field.*