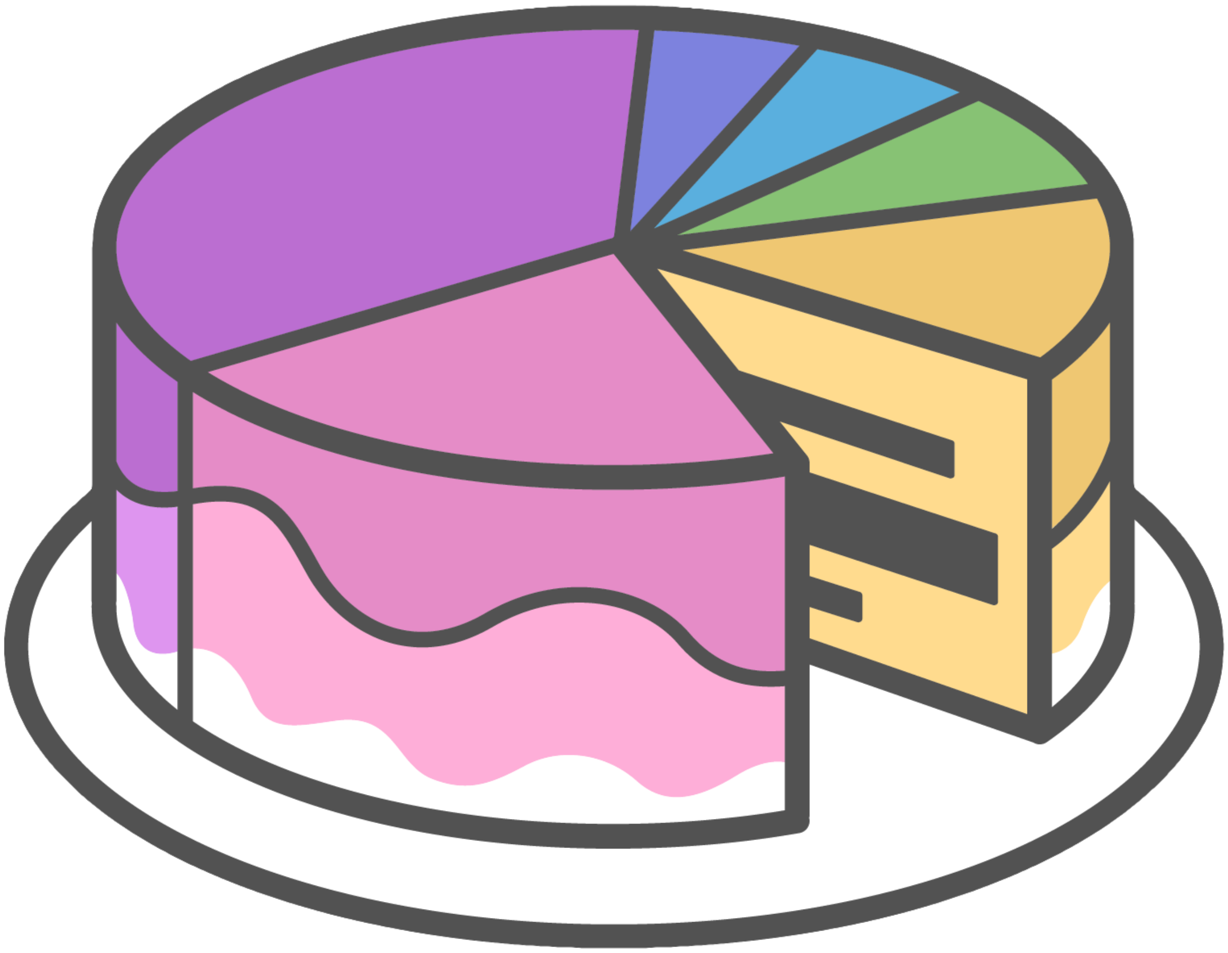


DataBake Documentation

Release 0.1

John Children

Aug 30, 2017

Contents

CHAPTER 1

Quickstart

This quickstart tutorial will take you through the very basic process behind creating a dataset with DataBake. You will learn how to specify columns and their relationships, preview your dataset and finally export it.

While there are a number of template projects available to view, for this tutorial we will create a blank project and build our dataset from there. To get started, enter a project name under the 'Create a new project' heading and click 'Let's go'. This will enable you to track your projects and change the datasets later.

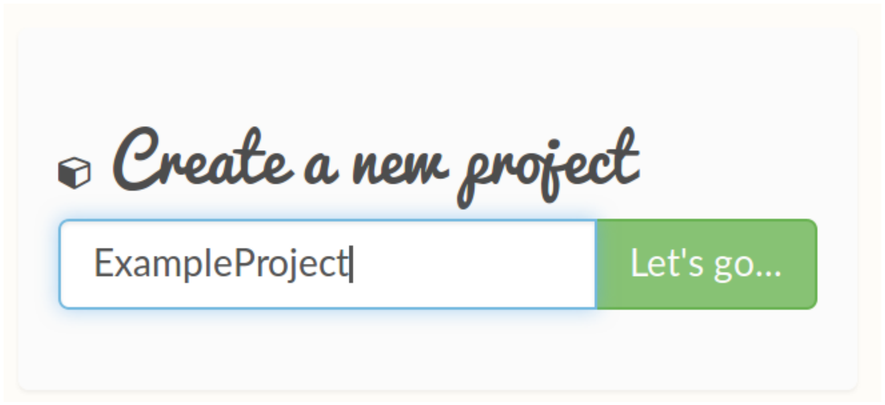


Fig. 1.1: The create project field.

Creating a new project will take you through to the main page - where you can define the columns and the insights in your personalised generated dataset. The title of your new project will appear on the top left-hand side and below it a list of tables in the project. Interacting with this list, you can re-name the table you are about to generate as it will become the name for the output file when exporting your dataset.

You will notice that a help icon on the top right of the main body and in various other places throughout the site. This will lead to the documentation for that section allowing you to quickly gain more insight into the process. If the documentation is not able to help you can also send a support ticket through the tab on the right hand side of the page.

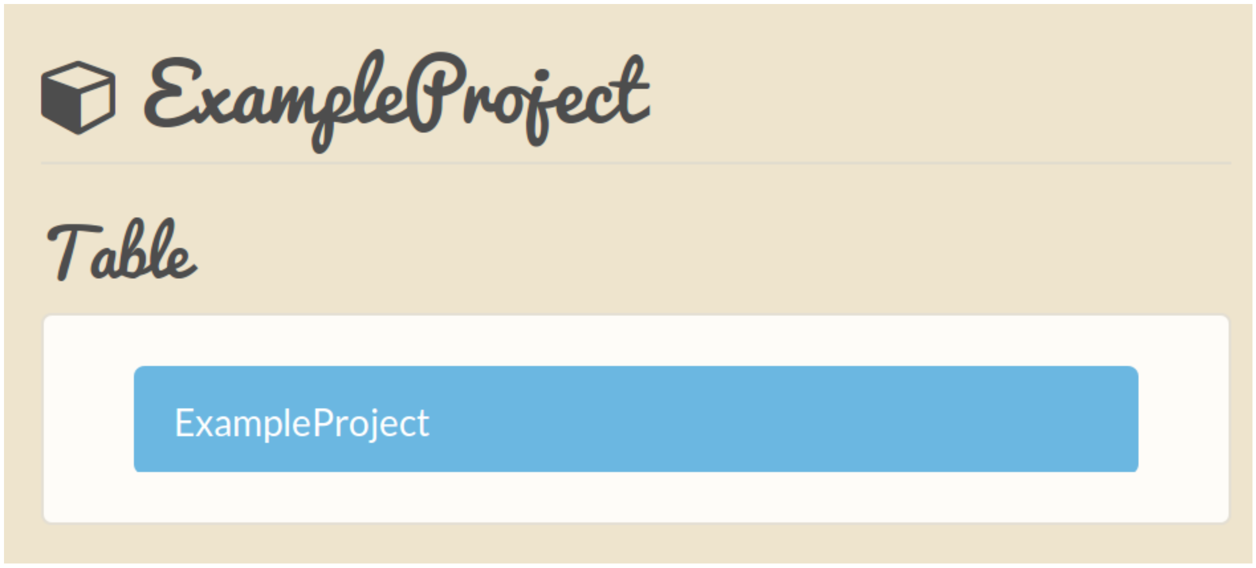


Fig. 1.2: The project view.

To the right of the project name, you will see four tabs titled 'Columns', 'Insights', 'Preview' and 'Export' respectively. These tabs will lead you through the process of generating a dataset using DataBake. Selecting the column tab, we will begin by defining the columns of our table.

Firstly, define the column variables by clicking the 'Add' button underneath the 'Columns' tab. A dialog box will appear and, as you can see, there is a drop down menu containing a range of pre-defined columns arranged by group. There is also the option to generate your own columns based on a custom specification.

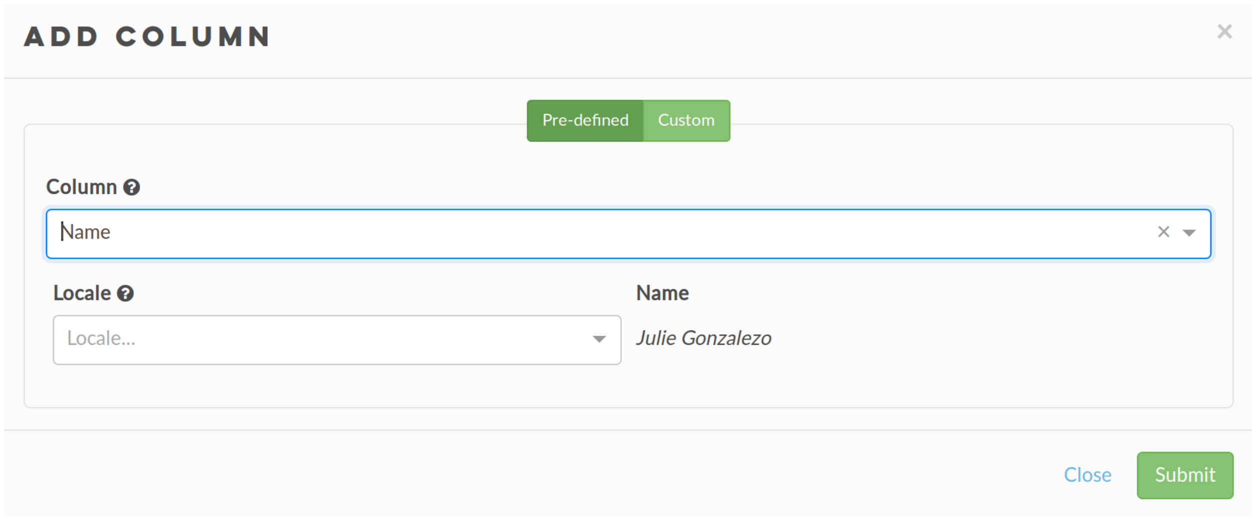
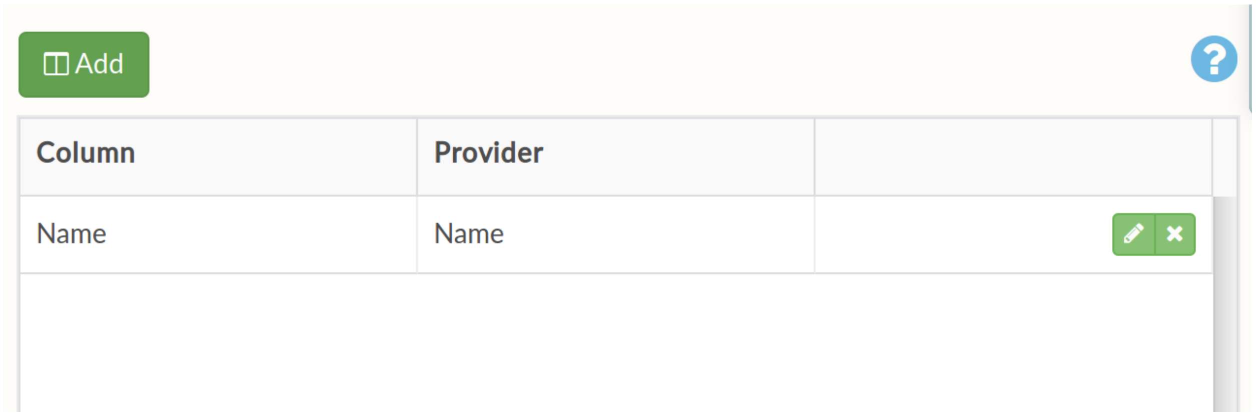


Fig. 1.3: The column creation dialog.

For this example, the first column we will be adding is a pre-defined one called **Name** - which creates a column filled with randomly generated full names. Click 'Submit' and a table with 'Name' in the column field will appear. You will notice that the option to pick a localisation setting for your generated names, ignore this for now but for more information see the profile providers sub-section of the *Columns* documentation.



Column	Provider	
Name	Name	

Fig. 1.4: The columns tab.

Next, click 'Add' again, scroll down the drop down menu to demographic data and select **Age**. Here you will notice that column values are calculated from on a numerical distribution and a histogram of values will be helpfully displayed under the heading of *Sample Data*. This histogram is dynamically generated by the browser and forms a rough approximation of the eventual output. Click 'Submit' and the 'Age' column will appear in the same table as 'Name'.

While pre-defined columns are easy to use, custom columns allow a great deal more flexibility for data generation. To add a custom column follow the same procedure as before by select *custom* instead of pre-defined in the column creation dialog. Type an example column name, such as **MagicNumber** and click on the 'Provider' drop down menu, and select 'Normal'. This will add a third column to the table as another normal distribution.

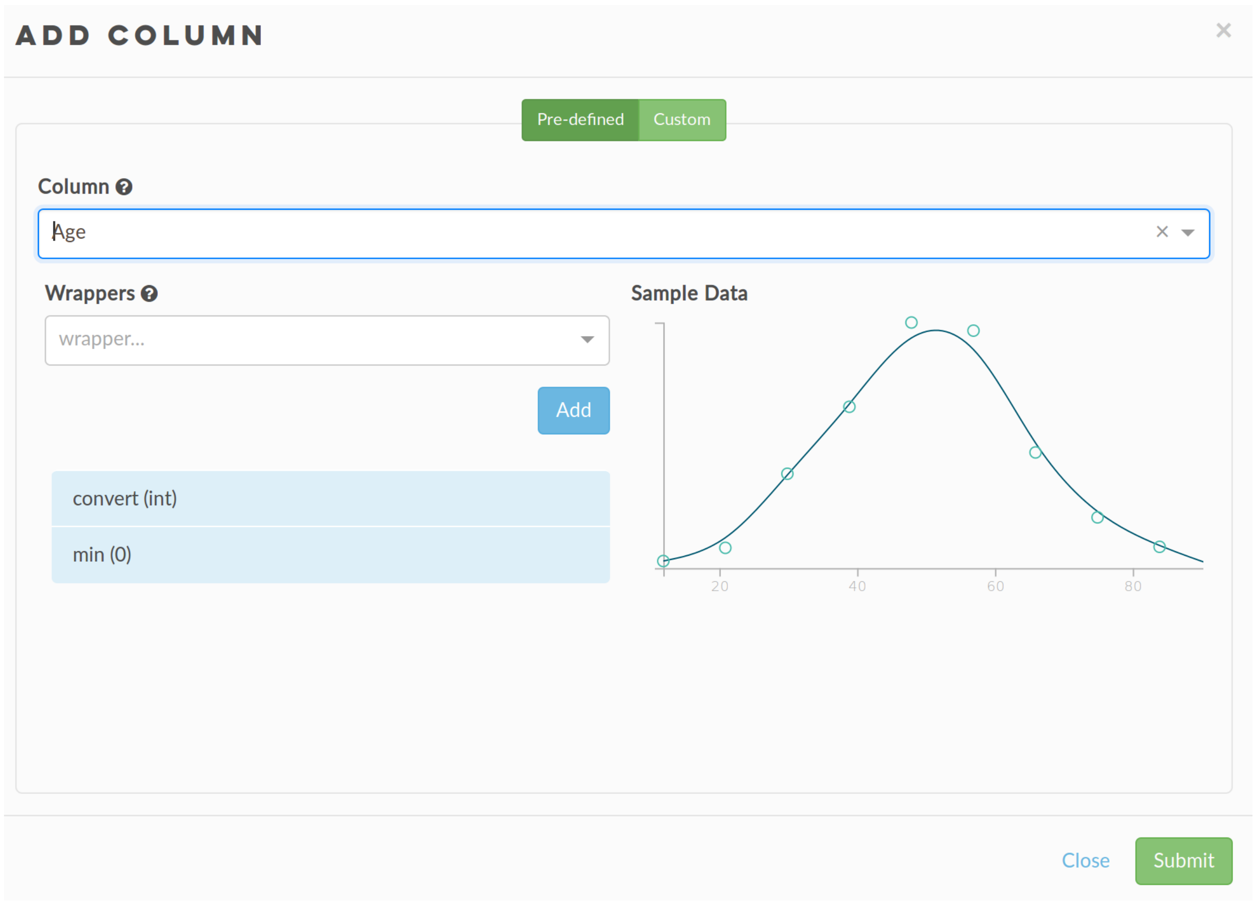
Now that we have defined our columns we can define the relationships between them; click on the 'Insights' tab to begin adding the relationships between the variables.

Click 'Add' and another dialog box will appear which will allow you to define how a column's values are calculated. Select 'MagicNumber' from the columns dropdown, which will be the dependent column in our new relationship. In the 'Mean' box begin to type `Age ** 2` and as you type 'Age', a blue box will appear below – click on this and the text will automatically update to `Age ** 2`. Clicking 'Submit' will return you to the Insights tab and you can see the connection between 'Age' and 'number' in the column relationships box on the left hand side.

You can now click on 'Preview' to view your dataset and insights in a table and charts. The default preview tab is the charts, which display histograms and bar charts built from a sample of the dataset. For more information see the *Preview* section of the documentation.

If you are interested in seeing a sample of raw data, a preview is available in the data tab where you can see a preview of 100 rows.

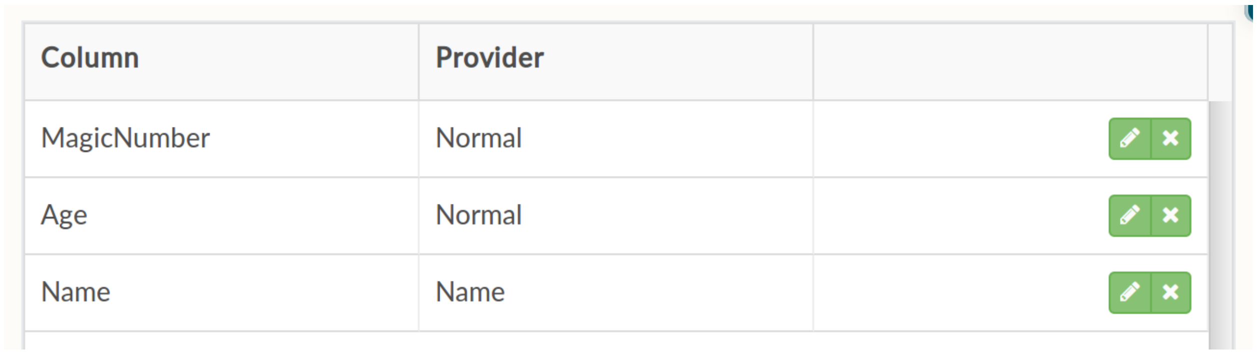
As we can from the table, the MagicNumber column contains decimal values, and ideally we would want it to be an integer instead. To achieve this, return to the columns tab and add a an int wrapper to the the magic numbre column which will convert all generated values to integers. Alternatively we could have rounded all the results, or added a constant to each of them.



convert (int)

min (0)

Fig. 1.5: A numerical columns preview



Column	Provider	
MagicNumber	Normal	
Age	Normal	
Name	Name	

Fig. 1.6: Three columns.

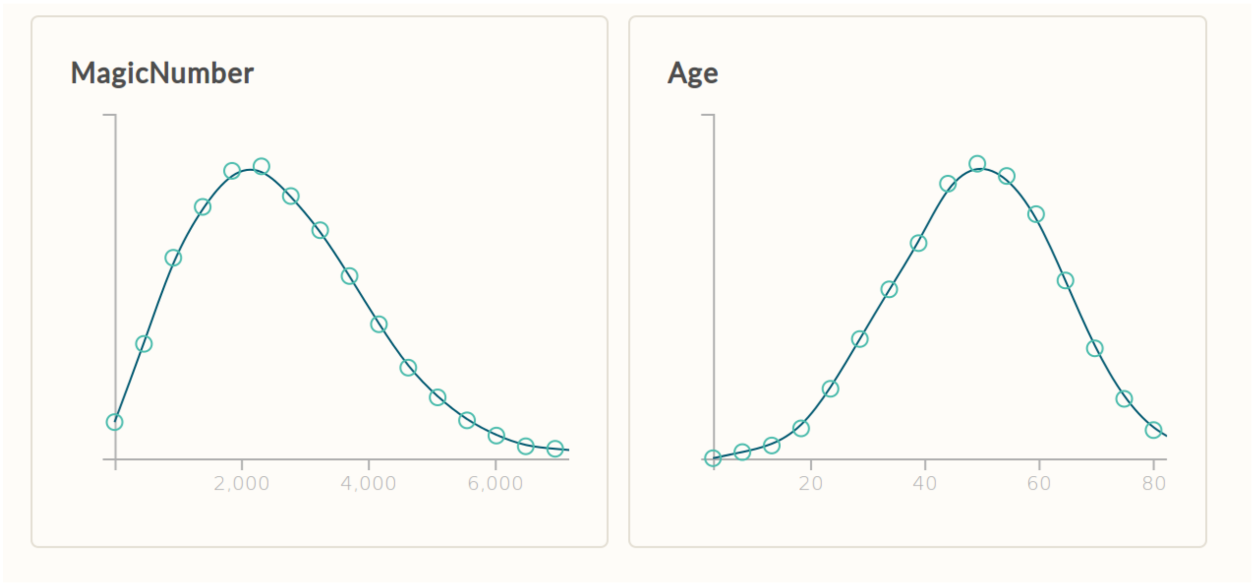
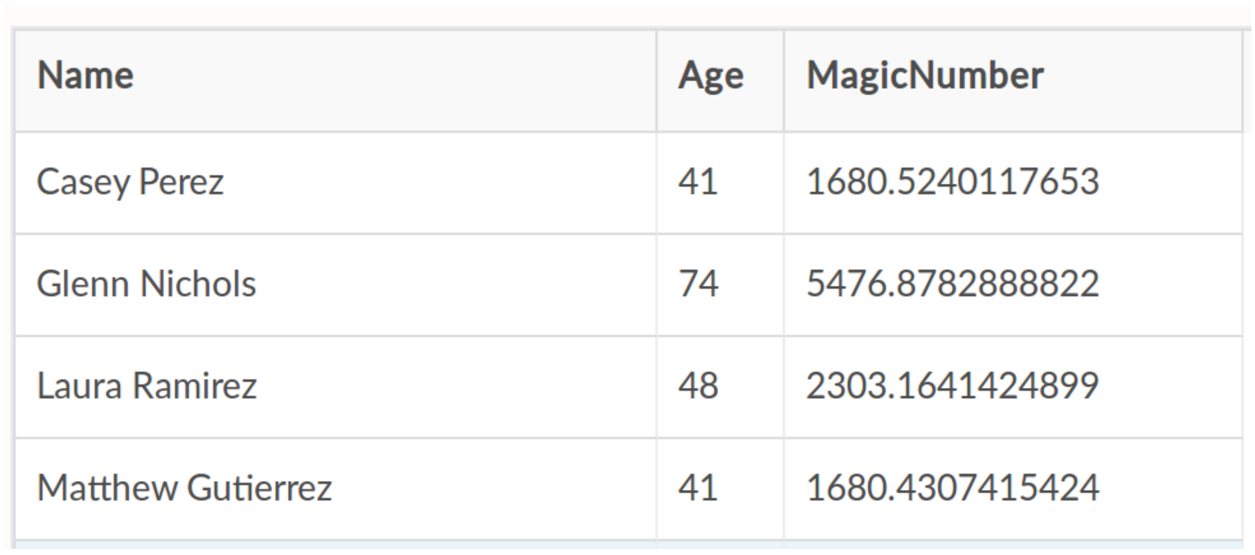


Fig. 1.7: Preview of charts.



Name	Age	MagicNumber
Casey Perez	41	1680.5240117653
Glenn Nichols	74	5476.8782888822
Laura Ramirez	48	2303.1641424899
Matthew Gutierrez	41	1680.4307415424

Fig. 1.8: Preview of the table.

When we are finally happy with out dataset, select the 'Export' tab to *Export* your dataset by choosing the number of rows you wish to generate. A download button will appear after the generation is complete.

CHAPTER 2

Projects

Projects let you keep track of your settings so that you can modify or regenerate your dataset. These projects will persist in your browser through local storage, allowing you to easily save and retrieve your work or switch between multiple projects.

Templates

Templates are a number of pre-built projects that can be used as examples datasets for reference or to be extended and modified for your own use. To use a template, simply select one from the drop-down at the initial DataBake screen or after selecting a new project. From there you can proceed as you would for a regular project.

Some examples of the kinds of relationships in these example projects can be found on the main site in the graphs gallery.

Tables

On the project page there is the option to configure the tables of your dataset. Currently DataBake only supports a single flat table as the intention is to add the ability to generate relational data in a future version.

By default the table will take the name of the project, but it can be changed as the table name will be used to name the output file during the *Export* process. Once you have your table you can continue by adding *Columns*.

CHAPTER 3

Columns

A key ingredient of DataBake is the columns tab, which allows users to define the columns of their data schema and how they are generated as well as any post-generation operations to apply. Columns can be selected from a pool of *pre-defined* variants or defined explicitly as *custom* columns. Currently there is a limit of 20 columns that can be added to a table.

Pre-defined

A number of pre-defined column types are available that can be easily added to your schema. These column types have been selected based on the types of customer data companies will typically collect. These columns can also be tweaked through the insight tab - in the same way as *custom* columns and can be renamed during creation.

Custom

Alternatively, column types can be defined manually in custom mode. From the custom tab, in addition to specifying a column name, you can choose your *providers* and *wrappers* to define the ways in which the data will be generated. Values for the column can then be tweaked with *Insights*.

Providers

Providers are functions which generate random values for the columns. They allow preformatted standard strings or numerical values to be added to a dataset with minimal configuration.

Basic

Basic providers are simple providers for generating specific kinds of data.

Table 3.1: Basic providers

Function	Effect
Custom	Returns a single value, very useful when combined with Insights.
ISO Date	Generates a timestamp between the 1st of January 1970 and the time of generation.
String	Returns a string specified by the formatting: `? will be replaced a random letter` `# will be replaced a random digit` `E.g. ####-???-ABCD` `=> '7432-YGL-ABCD'`
Location	Generates a random latitude longitude pair.

Profile

Profile providers relate to personal information about individuals commonly used in datasets.

Table 3.2: Profile providers

Provider	Effect
Address	Generates a random address.
Job	Generates a random job title.
Last Name	Generates a random last name.
Email	Generates a random email address.
Name	Genetates a random full name.
Phone Number	Generates a random phone number.
Social Security Number	Generates a random social security number (or national equivalent).

For all of these providers an optional locale can be set which will generate values specific to that location.

Distributions

These providers will sample a distribution to generate random values. Parameters can be tweaked through Insights, but only a single sample can be returned per entry.

Table 3.3: Distribution providers

Distribution	Effect
Uniform	Generates values from a uniform distribution (see `numpy.random.uniform()`).
Normal	Generates values from a normal distribution (see `numpy.random.normal()`).
Log Normal	Generates values from a log normal distribution (see `numpy.random.lognormal()`).
Beta	Generates values from a beta distribution (see `numpy.random.beta()`).
Gamma	Generates values from a gamma distribution (see `numpy.random.gamma()`).

Choice Distributions

Choice distributions are some of the most convoluted but powerful providers which allow relationships to applied from continuous values to discrete ones.

Table 3.4: Choice distribution providers

Distribution	Effect
Choice Distribution	Bisect a list at a point from a uniform distribution.
Beta Choice Distribution	Bisect a list at a point from a beta distribution.
Triangular Choice Distribution	Bisect a list at a point from a triangular distribution.

Wrappers

Wrappers allow for a series of operations to be applied to each value generated for a column. These will be applied in the sequence in which they are added.

Table 3.5: Wrappers

Wrapper	Effect
add	Offset the value by a float or int.
multiply	Multiply the value by a float or int.
max	Filter out results greater than a maximum.
min	Filter out results smaller than a minimum.
round	Reduce a float value to a number of digits after the decimal point.

If the requested operation cannot be applied to the value a None type will be returned instead.

CHAPTER 4

Insights

DataBake can generate insights in your dataset by through a range of methods.

The simplest type of insight is defining the literal arguments - which can give you a constant value for a variable in a generation function. For example, a variable with based on a normal distribution can be changed to show insights by changing the mean and standard deviation of the *Columns*.

The expressions which define the functions's arguments are processed using `ast.literal_eval()`. These are then passed to the function where the new values are generated. This allows for some basic operators to be utilised - while still allwowing your dataset to be generated as quickly as possible. Some examples of the types of literals that can be used in this way are:

```
5
3.14
7.4E-9
[1, 2]
'abc'
3 + 1
```

For more details see the python documentation of `ast.literal_eval()`.

When modifying the variables, a small preview graph will be generated to give a helpful in browser plot of the shape of the expected distribution. Once you add the insight this graph will then be derrived from a preview of the dataset. However, if the resulting expression cannot be used as an argument for the function, the server will return an error and a preview will not be displayed.

For additional information about the types of functions provided and their arguments, see *Columns*.

Choices

Another way of generating insights is by using the choice providers.

Before any bias is applied, each item will have an equal chance by default. When using a choice provider to determine a columns' insight, you will be able to change the ratios at which the items are likely to be generated as a value. This can be in addition to any other providers.

Relationships

The most powerful aspect of insights is the ability to add relationships between columns to build dependent variables into your datasets. Using this feature, the arguments passed to the generation function will be updated for each value. This has a substantial perfomance tradeoff compared to using constant expressions, but is necessary to create the most interesting datasets.

Supposing that you have already defined three columns, *age*, *average_product_price*, and *favourite_product*, you can now define a simple relationship between the age of the customers in our dataset and the price of the products they are buying.

age
average_product_price
favourite_product

Fig. 4.1: Unconnected columns.

This relationship is added by creating an insight for *average_product_price* and setting one of the arguments in the provider to be derived from another column. The arguments for each provider will appear as forms inside the insights dialog window once a column is selected, where you can type in the column name you wish to derrive values from as a variable. For example, if we were to add an expression containing the column name 'age' as an argument, this will mean that each time a value for *average_product_price* is generated for the dataset, the value in the age column will be used to calculate each value for the column to which the insight has been added.

More specifically, if *average_product_price* uses a normal distribution to generate values, a relationship could be added to make average product price a multiple of age by entering 'age * 30' into the *Mean* box. Once the dataset is generated this would have the effect of simulating a larger range of average purchase prices as people grow older.

This insight could be visualised as a connection between the two variables in a directed graph.

Extending this idea further, more relationships can be added between *favourite_product* and the other two columns. Not only can this be done through having a source for each variable, but the two variables can even be combined with one another.

These relationships are resolved before generation and used to work out the order in which columns should be generated, but it means there is an important condition that the relationships between columns must form a directed acyclic graph - such that no variable is dependent upon itself.

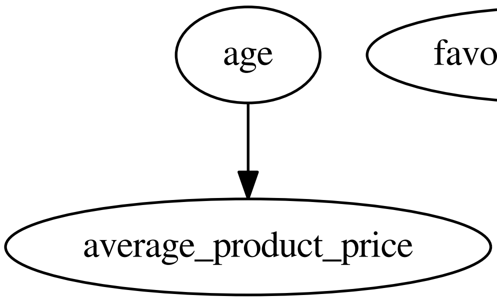


Fig. 4.2: A single relationship.

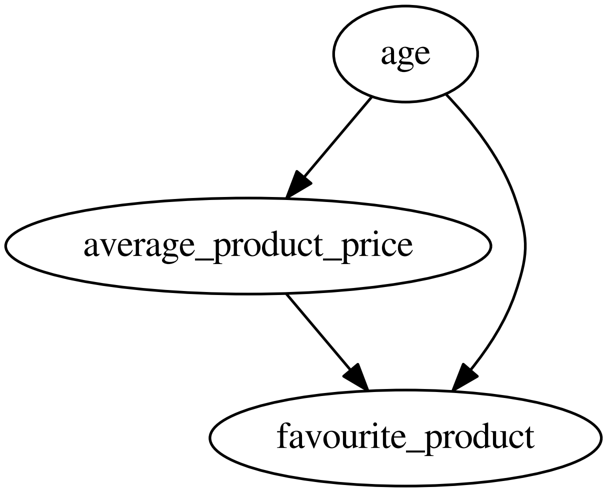


Fig. 4.3: The complete relationships

DataBake will display a simplified version of these relationships in a graph on the left hand side of the page under the heading of **Column Relationships**. Here you will be able to see how each column connects to one another, including the expression that dictates their relationship.

Editor

When editing column relationships in the insights dialog window, a few additional features are available. Arithmetic operators can be used in your expressions including modulo and indices, though it is important to note that there is an execution time limit for each cell so you will not be able to use generate datasets containing expressions like `9**9**9**9**9**9**9` as they will fail to preview. The == comparison operator is also available, primarily for string comparisons in column relatiponships.

Functions

In order to provide flexibility in dataset creation a number of extra functions are available for use in insights.

Builtins

Builtins are basic python functions for commonly used operations.

Table 4.1: Common builtin functions.

Function	Effect
`abs()`	Returns the absolute value of a numerical object.
`all()`	Returns true if all conditions in an iterable are met.
`any()`	Returns true if any conditions in an iterable are met.
`chr()`	Returns the string character of an integer.
`dir()`	Returns the list of names in the current scope.
`hash()`	Returns the hash value of the object.
`len()`	Returns the number of items of an object.
`max`	Returns the largest value in an object.
`min`	Returns the smallest value in an object.
`ord()`	Returns integer representation of a unicode character.
`pow()`	Returns the first argument to the power of the second.
`round()`	Returns the first value rounded to digits specified by the second.
`sorted()`	Returns a sorted version of an object.
`sum()`	Returns the sum of all objects.

Type Conversion

Type conversions let you manipulate the kinds of data being passed into functions.

Table 4.2: Type functions.

Function	Effect
`bin()`	Converts an integer number to a binary string.
`bool`	Converts a value to boolean.
`complex`	Converts a value to a complex number.
`float`	Converts a value to a floating point number.
`hex()`	Converts a value to its hex representation.
`int`	Converts a value to an integer.
`oct()`	Converts a value to its oct representation.
`str`	Converts a value to a string.

Statistical

Statisical functions let you draw values from distributions to be used as function arguments.

Table 4.3: Statistical functions.

Function	Effect
`normal`	Draws random samples from a normal distribution.
`triangular`	Draws random samples from a triangular distribution.
`uniform`	Draws random samples from a uniform distribution.
`poisson`	Draws random samples from a poisson distribution.
`beta`	Draws random samples from a beta distribution.

CHAPTER 5

Preview

The preview tab lets you see an example of a small dataset of 100 rows generated from your project settings which allows you to get an idea of a sample of what your dataset might look like without committing to generating it in full. This is very useful when checking for data nullified by wrappers and catching errors.

Charts

A few charts are available for most columns that will display the preview data on histograms or fitted curve plots, these charts are from a larger set of 1000 rows and normalised in order to reduce the effect of noise on the shape of the distribution. These charts will give you a good idea as to what the shape of the distributions in your final dataset will be so that it is easy to get quick feedback on your distributions so that it is adjustable. In addition to the histogram, a gaussian kernel density estimator (KDE) will be plotted on the graph that will give the distribution a smoother shape for visualisation.

Data

Alternatively, you can see a sample of the kind of data created from the data tab. This will show you a table of the 100 preview rows so that you can see the underlying data.

Scatter plots

When you have multiple distributions in your dataset, scatter plots of each column will become available, allowing you to easily visualise the correlations between columns in the generated data. To allow flexibility in this view, you can deselect columns you are not interested in seeing as well as colour coding the points by categorical data.

CHAPTER 6

Export

Export will allow you to generate and obtain an instance of your specified dataset by running a generation task in the cloud.

After pressing the *Export* button in the tab, you will be able to choose the number of rows you wish to generate. Selecting your desired amount and then submitting your request you will take you back to the export tab displaying a time estimate for the expected duration of the process.

Once the job is complete you will receive a link to a zip compressed file containing a copy of your dataset stored remotely in an S3 bucket, which will be available for a week.

Currently only comma seperated value (csv) outputs are supported but more export types are planned for the near future.