**Kaggle Learn**

**1. Intro to Programming**

**1.1 Arithmetic and Variables**

A white table with black text and symbols

Description automatically generated

**PEMDAS (order of operations):**

A close up of a sign

Description automatically generated

A screenshot of a math test

Description automatically generated

A screenshot of a math test

Description automatically generated

**Variables names rules:**

A screenshot of a computer

Description automatically generated

**1.2 Functions**

A close-up of a white background

Description automatically generated

**1.3 Data Types**

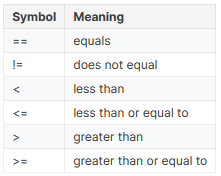
When you multiple an integer or float by a boolean with value True, it just returns that same integer or float (and is equivalent to multiplying by 1). If you multiply an integer or float by a boolean with value False, it always returns 0. This is true for both positive and negative numbers. If you multiply a string by a boolean with value True, it just returns that same string. And if you multiply a string by a boolean with value False, it returns an empty string (or a string with length zero).

When you add booleans, adding False is equivalent to adding 0, and adding True is equivalent to adding 1.

A screenshot of a computer program

Description automatically generated

**1.4 Conditions and Conditional Statements**



A screenshot of a white background

Description automatically generated

**1.5 Intro to Lists**

You can also pull a segment of a list (for instance, the first three entries or the last two entries). This is called slicing. For instance:

* to pull the first x entries, you use [:x], and
* to pull the last y entries, you use [-y:].

**2. Python**

**2.1 Hello, Python**

A screenshot of a computer

Description automatically generated

a % b example (the modulo operator):

A screenshot of a computer

Description automatically generated

**2.2 Functions and Getting Help**

Use the help() function to get information on some function:

A screenshot of a computer

Description automatically generated

Docstrings or comments in your own functions:

A screenshot of a computer program

Description automatically generated

The print() function has a special feature sep that specifies a separator between values. If you don’t specify a separator your values will be separated by spaces.

A screenshot of a math problem

Description automatically generated

Calling functions on functions:

A screenshot of a computer code

Description automatically generated

**2.3 Booleans and Conditionals**

A screenshot of a computer

Description automatically generated

Function that checks if a number is odd:

A white rectangular object with text

Description automatically generated with medium confidence

**2.4 Lists**

Swapping two variables:

A screenshot of a math problem

Description automatically generated

**2.6 Strings and Dictionaries**

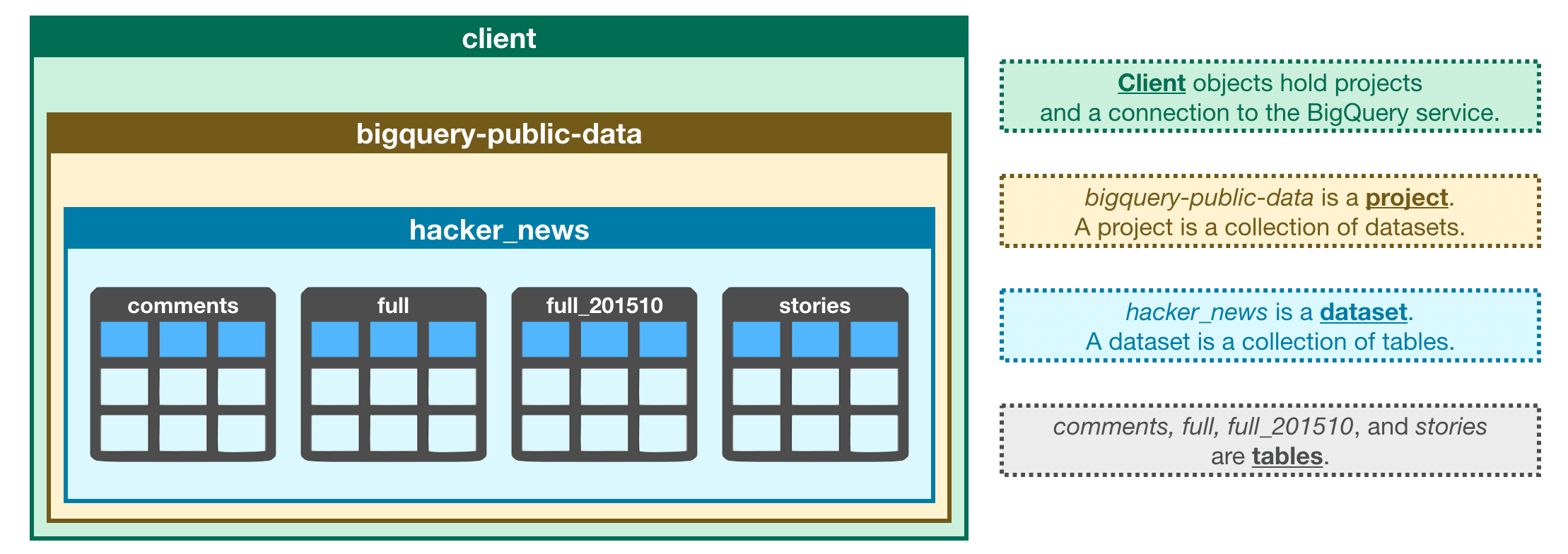
Important uses of the backslash character:

A screenshot of a chat

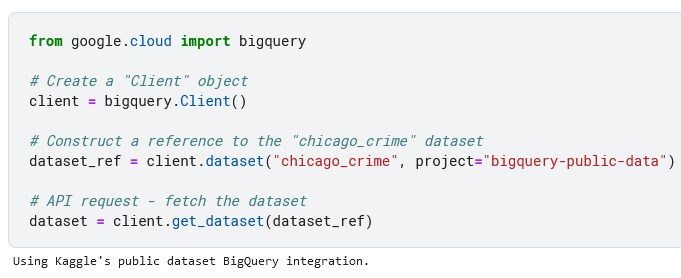
Description automatically generated

**3. Intro to SQL**

**3.1 Getting Started With SQL and BigQuery**



Use this code to fetch a dataset from BigQuery:

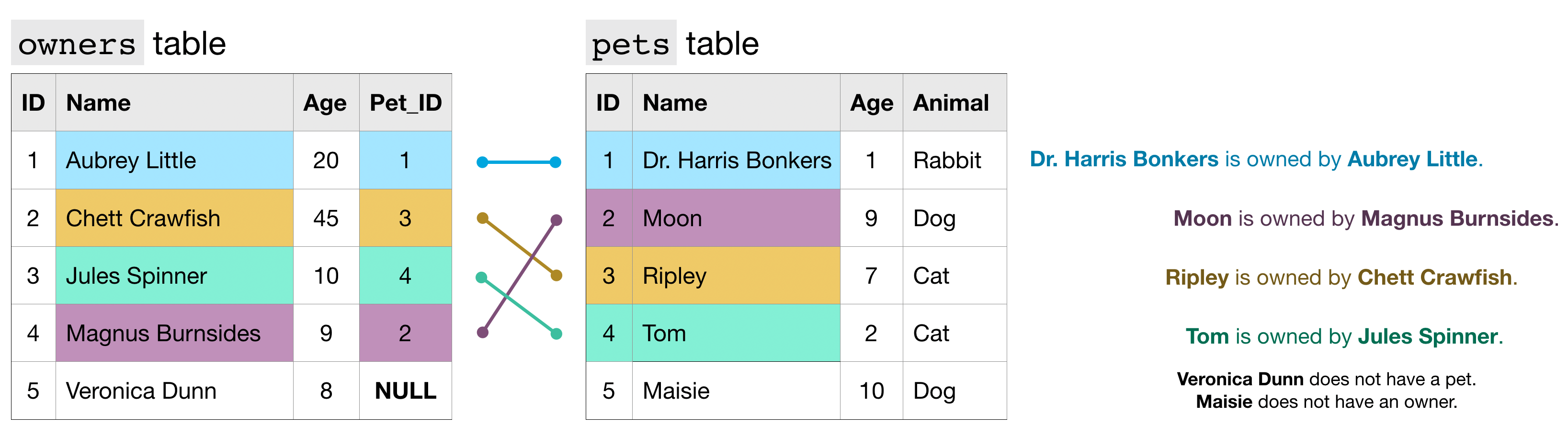


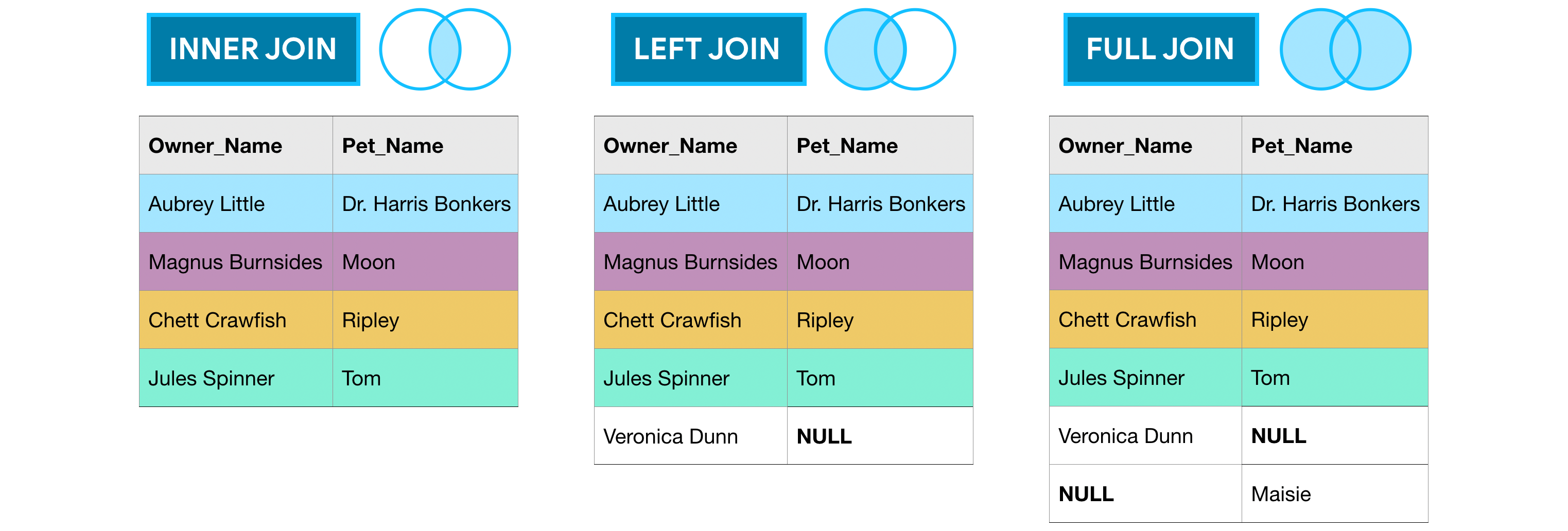
**3.3 GROUP BY, HAVING & COUNT**

When using GROUP BY with COUNT you can use COUNT(1) to count the rows in each group.

**4. Advanced SQL**

**4.1 JOINs and UNIONs**

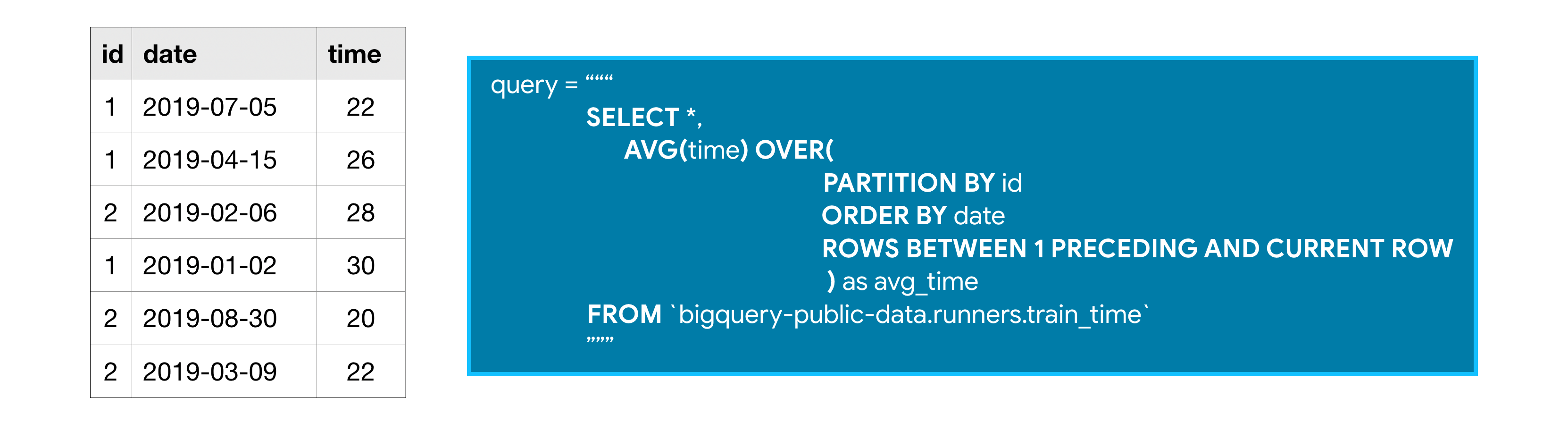




Use UNION ALL to place the second table below the first, use UNION DISTINCT if you want to remove duplicate values.

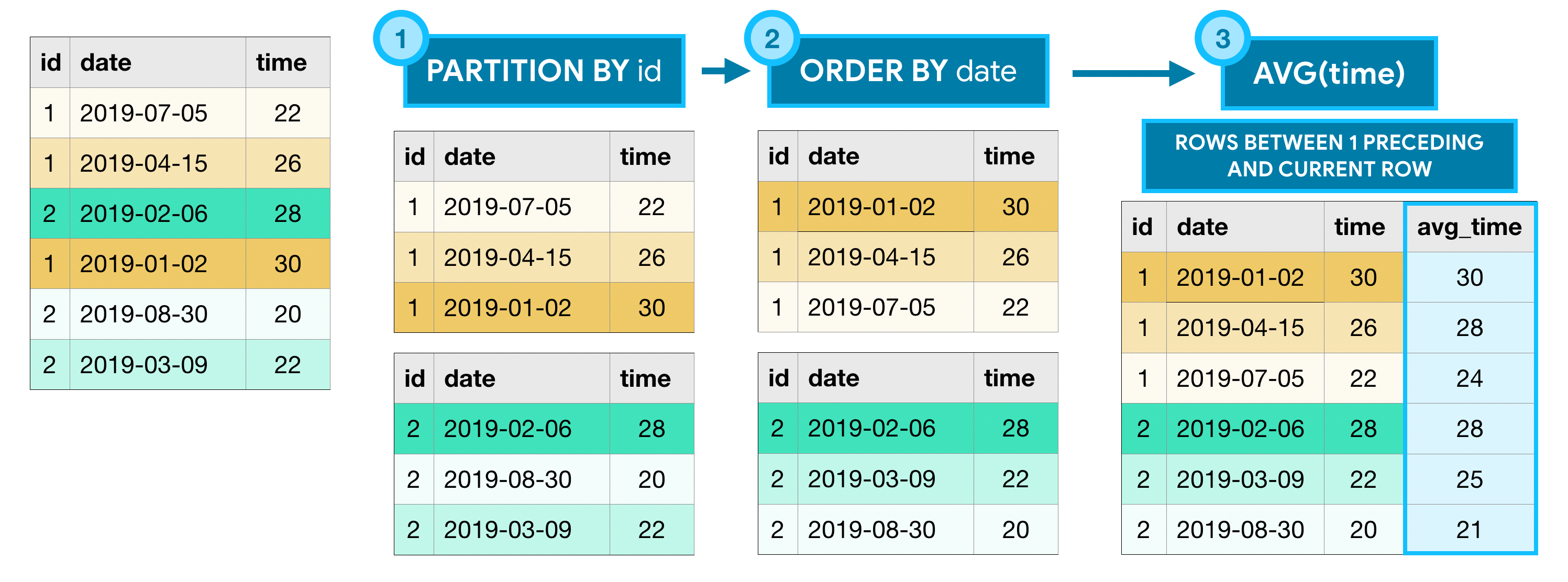
**4.2 Analytic Functions**

Example of a moving average:



A white background with black text

Description automatically generated



A screenshot of a computer

Description automatically generated

**Google Cloud documentation on** [**Window functions**](https://cloud.google.com/bigquery/docs/reference/standard-sql/window-function-calls)**.**

Calculating the cumulative (cumsum) number of trips per day:

A screenshot of a computer

Description automatically generated

Track beginning and ending station on October 25, 2025 for each bike:

A screenshot of a computer program

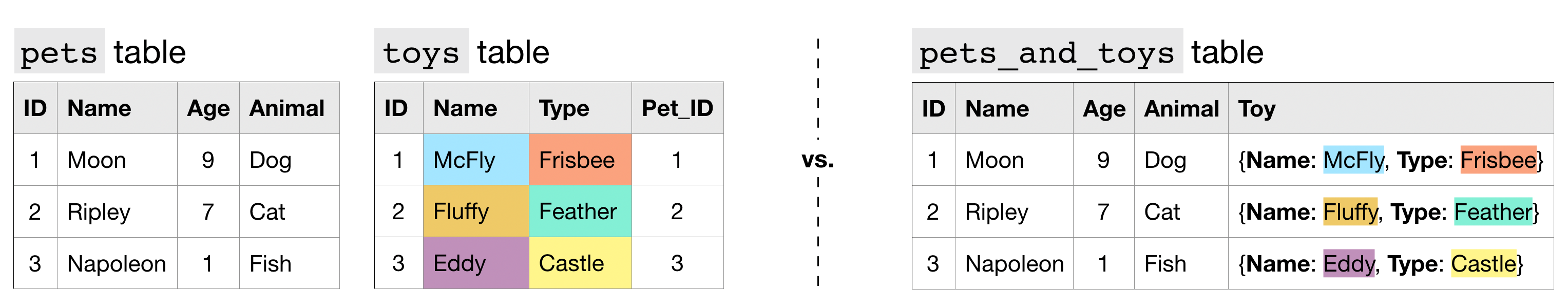
Description automatically generated

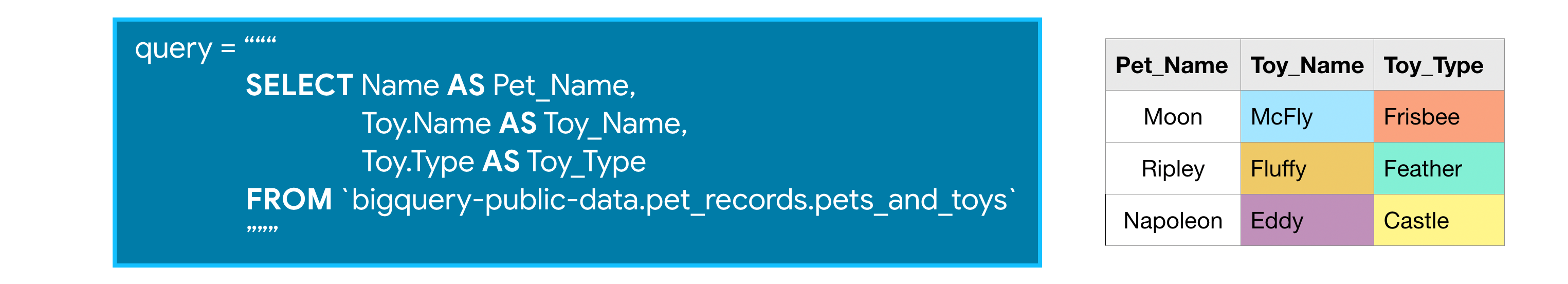
A screenshot of a computer

Description automatically generated

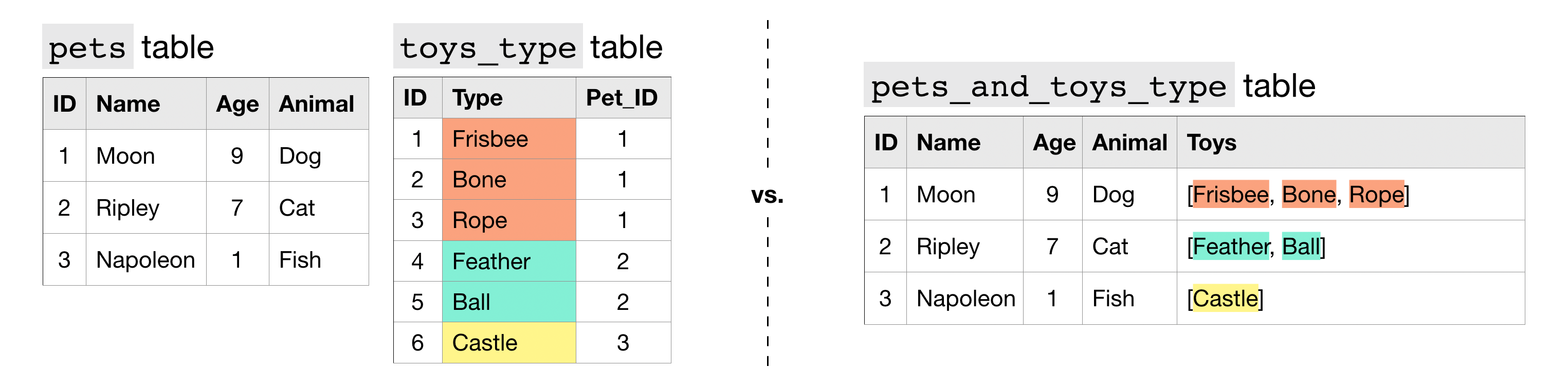
**4.3 Nested and Repeated Data**

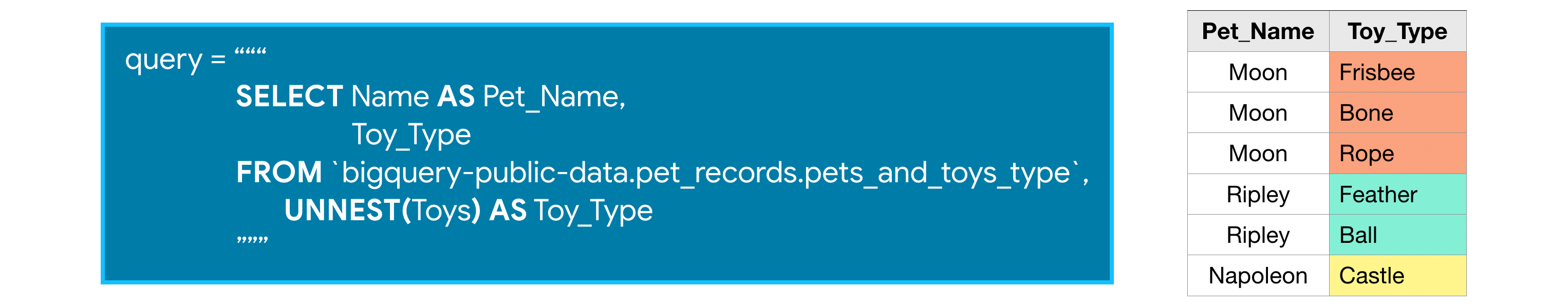
**Nested data (datatype STRUCT or RECORD):**



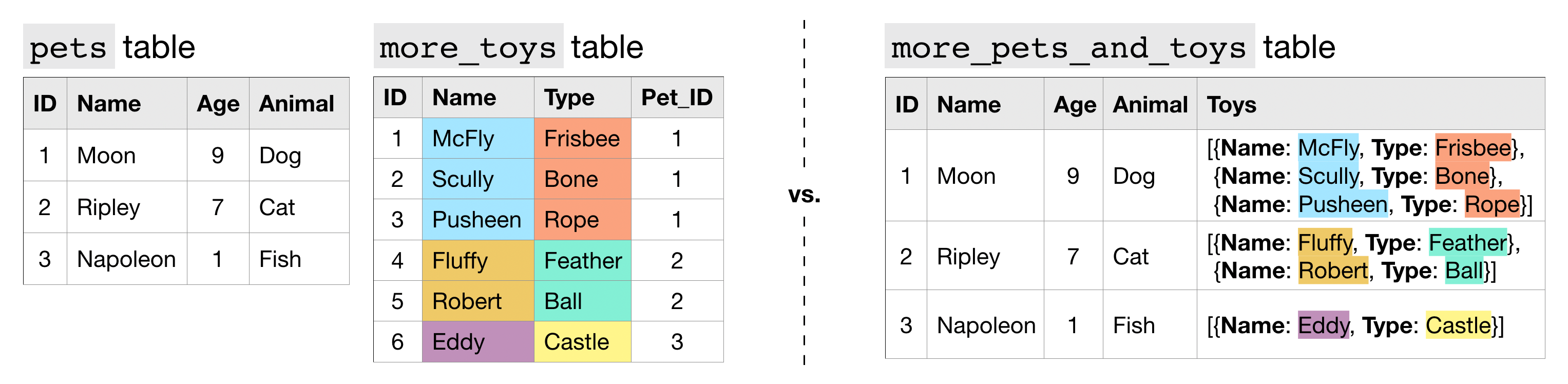


**Repeated data (datatype REPEATED):**





**Nested and repeated data:**





Unnesting example:

A screenshot of a computer code

Description automatically generated

**4.4 Writing Efficient Queries**

1. Only select the columns you want;
2. Read less data;
3. Avoid N:N JOINs (many to many connections).

**5. Pandas**

**5.3 Summary Functions and maps**

To subtract a mean value from a column you can use the .map() function:

A screenshot of a computer

Description automatically generated

Or .apply():

A screenshot of a computer

Description automatically generated

Or the pandas functional:

A screenshot of a computer

Description automatically generated

You can also combine columns this way:

A screenshot of a computer

Description automatically generated

**5.5 Data Types and Missing Values**

Return the datatype of every column in a DataFrame using df.dtypes:

A screenshot of a computer

Description automatically generated

**6. Data Cleaning**

**6.1 Handling Missing Values**

Calculate the number of missing values in the first 10 columns of a DataFrame using df.isnull().sum():

A screenshot of a computer

Description automatically generated

To calculate the percentage of missing data you can use np.product(df.shape):

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

Remove all rows/columns that contain missing values using df.dropna():

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Use df.fillna() to specify the value you want to fill the null/NA values with:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**6.2 Scaling and Normalization**

**Scaling**

Changing the range of your data. This means that you're transforming your data so that it fits within a specific scale, like 0-100 or 0-1. You want to scale data when you're using methods based on measures of how far apart data points are, like support vector machines (SVM) or k-nearest neighbors (KNN). With these algorithms, a change of "1" in any numeric feature is given the same importance.

For example, you might be looking at the prices of some products in both Yen and US Dollars. One US Dollar is worth about 100 Yen, but if you don't scale your prices, methods like SVM or KNN will consider a difference in price of 1 Yen as important as a difference of 1 US Dollar! This clearly doesn't fit with our intuitions of the world. With currency, you can convert between currencies. But what about if you're looking at something like height and weight? It's not entirely clear how many pounds should equal one inch (or how many kilograms should equal one meter).

Scaling example:

A screenshot of a computer screen

Description automatically generated

**Normalization**

Changing the shape of the distribution of your data. The point of normalization is to change your observations so that they can be described as a normal distribution (the “bell curve”).

In general, you'll normalize your data if you're going to be using a machine learning or statistics technique that assumes your data is normally distributed. Some examples of these include linear discriminant analysis (LDA) and Gaussian naive Bayes. (Pro tip: any method with “Gaussian” in the name probably assumes normality.).

Normalization example:

A screenshot of a computer screen

Description automatically generated

**6.3 Parsing Dates**



To change a date like *m/d/yy* to *d-m-yyyy* use pd.to\_datetime(). You can add infer\_datetime\_format = True to parse dates automatically.

A screenshot of a computer program

Description automatically generated

You can check how values in a date column are different from each other using str.len() and .value\_counts():

A screenshot of a computer code

Description automatically generated

**6.4 Character Encodings**

UTF-8 is the standard text encoding. All Python code is in UTF-8 and, ideally, all your data should be as well. It's when things aren't in UTF-8 that you run into trouble.

Use .encode(“utf-8”, errors = “replace”) or .decode(“utf-8”):

A screenshot of a computer program

Description automatically generated

Example:

A screenshot of a computer

Description automatically generated

**Opening a file that is not in UTF-8:**

A screenshot of a computer program

Description automatically generated

**6.5 Inconsistent Data Entry**