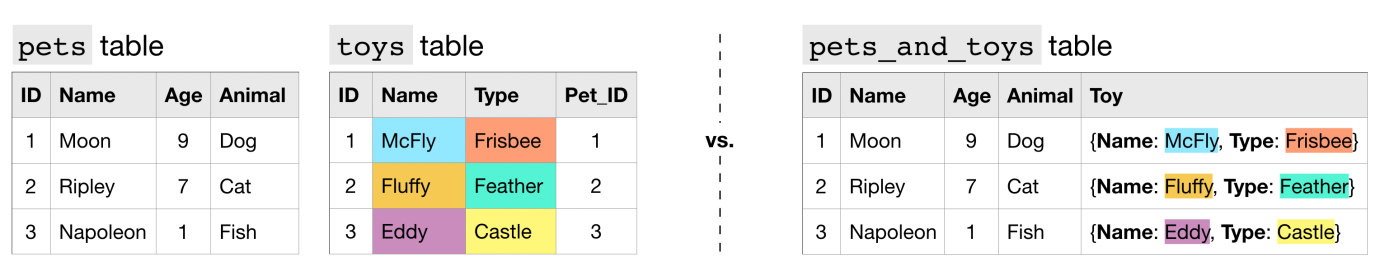
Advanced SQL: 3rd lesson – Nested & Repeated Data

So far, you've worked with many types of data, including numeric types (integers, floating point values), strings, and the DATETIME type. In this tutorial, you'll learn how to query nested and repeated data. These are the most complex data types that you can find in BigQuery datasets!

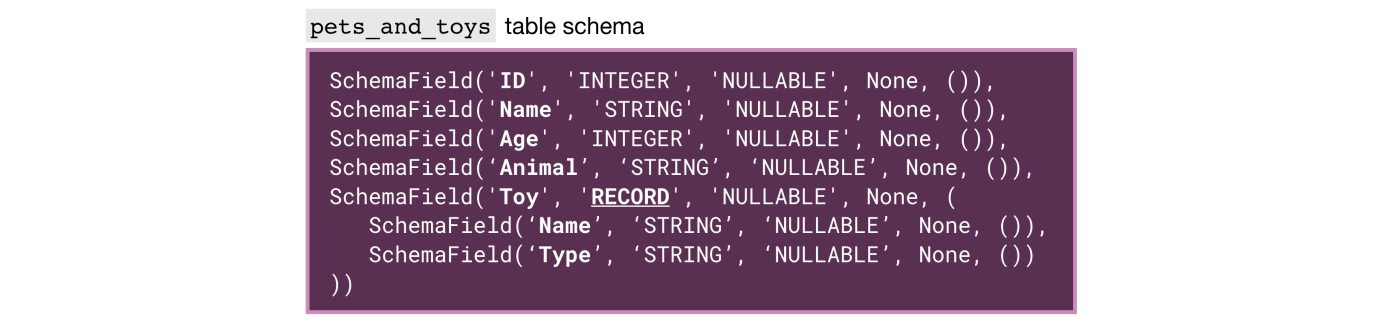
Nested data:

Consider a hypothetical dataset containing information about pets and their toys. We could organize this information in two different tables (a pets table and a toys table). The toys table could contain a "Pet\_ID" column that could be used to match each toy to the pet that owns it.

Another option in BigQuery is to organize all of the information in a single table, similar to the pets\_and\_toys table below.



Nested columns have type STRUCT (or type RECORD). This is reflected in the table schema below. Recall that we refer to the structure of a table as its schema. If you need to review how to interpret table schema, feel free to check out this lesson from the Intro to SQL micro-course.



To query a column with nested data, we need to identify each field in the context of the column that contains it:

* Toy.Name refers to the "Name" field in the "Toy" column, and
* Toy.Type refers to the "Type" field in the "Toy" column.

query = “””

SELECT Name AS Pet\_Name,

Toy.Name AS Toy\_Name,

Toy.Type AS Toy\_Type

FROM `bigquery-public-data.pet\_records.pets\_and\_toys`

“””

------------ ------------- ------------

Pet\_Name Toy\_Name Toy\_Type

------------ ------------- ------------

Moon McFly Frisbee

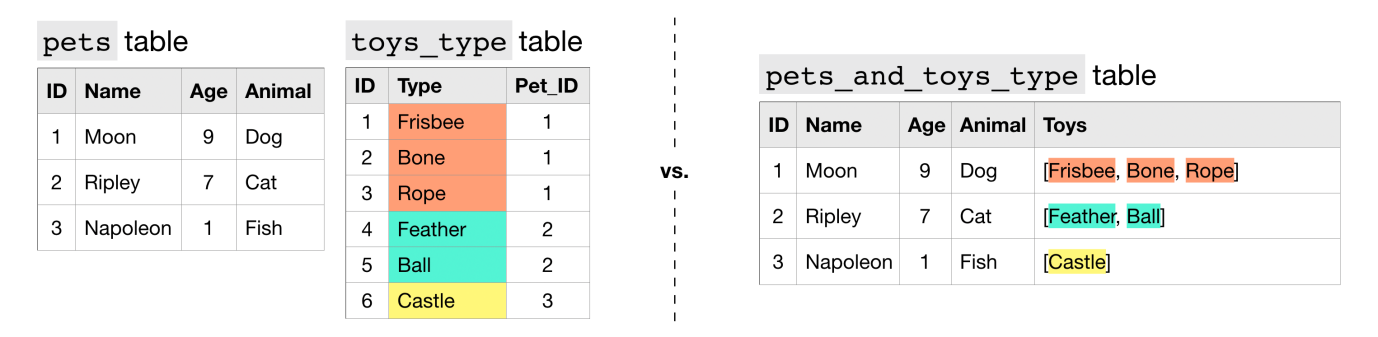
Ripley Fluffy Feather

Napoleon Eddy Castle

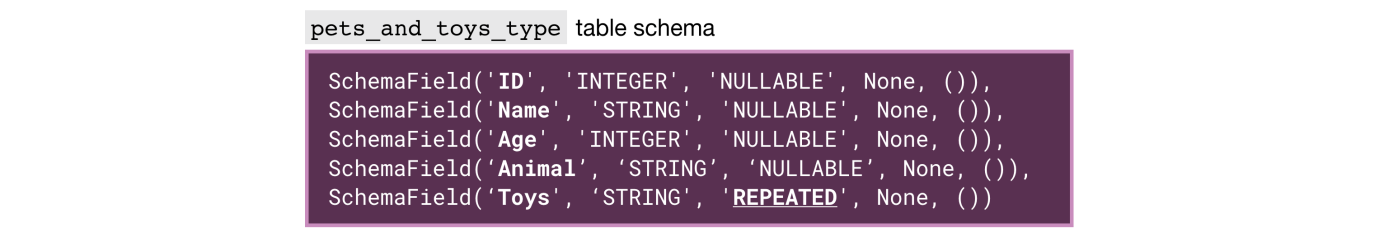
Otherwise, our usual rules remain the same - we need not change anything else about our queries.

Repeated data:

Now consider the (more realistic!) case where each pet can have multiple toys. In this case, to collapse this information into a single table, we need to leverage a different datatype.



We say that the "Toys" column contains repeated data, because it permits more than one value for each row. This is reflected in the table schema below, where the mode of the "Toys" column appears as 'REPEATED'.



Each entry in a repeated field is an ARRAY, or an ordered list of (zero or more) values with the same datatype. For instance, the entry in the "Toys" column for Moon the Dog is [Frisbee, Bone, Rope], which is an ARRAY with three values.

When querying repeated data, we need to put the name of the column containing the repeated data inside an UNNEST() function.

query = “””

SELECT Name AS Pet\_Name,

Toy\_Type

FROM `bigquery-public-data.pet\_records.pets\_and\_toys\_type`,

UNNEST(Toys) AS Toy\_Type

“””

------------ ------------

Pet\_Name Toy\_Type

------------ ------------

Moon Frisbee

Moon Bone

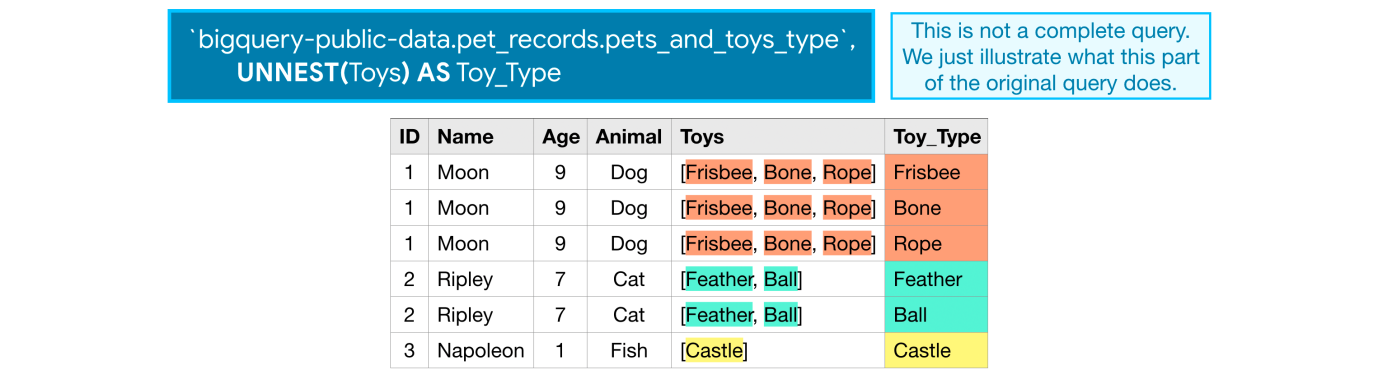
Moon Bone

Ripley Feather

Ripley Ball

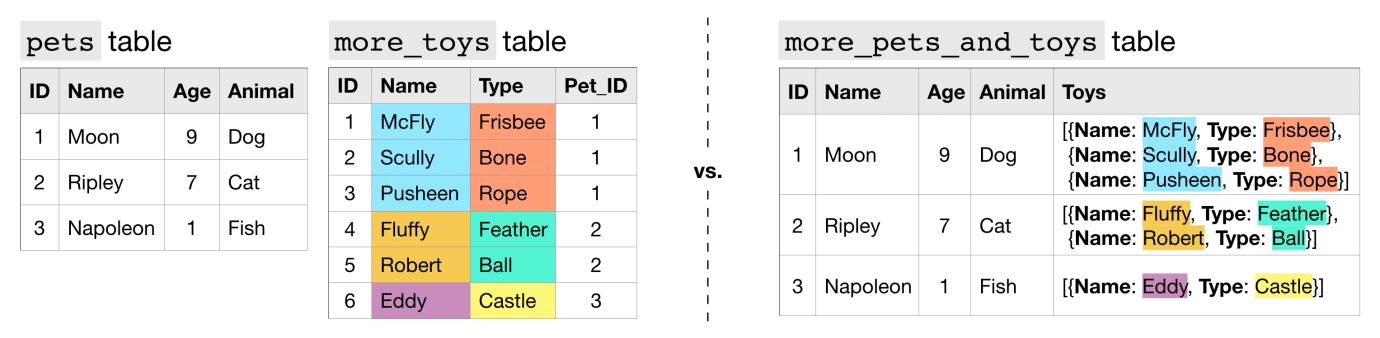
Napoleon Castle

This essentially flattens the repeated data (which is then appended to the right side of the table) so that we have one element on each row. For an illustration of this, check out the image below.

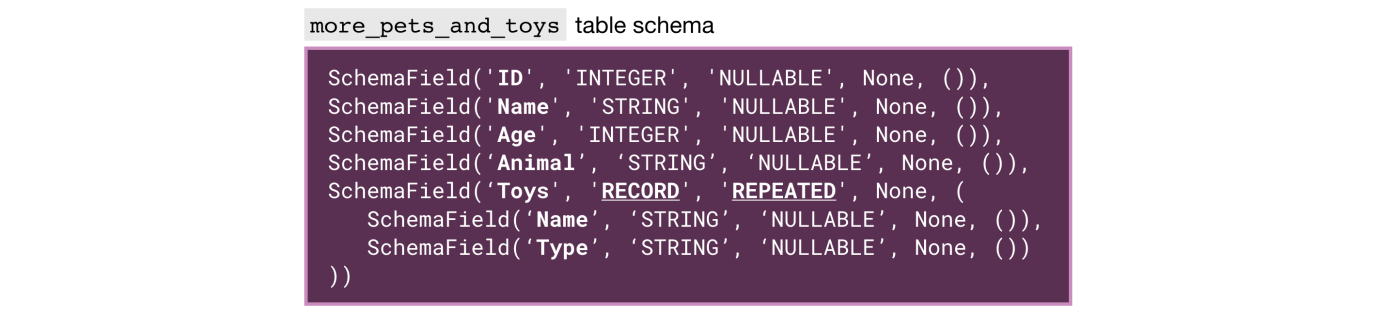


Nested and repeated data:

Now, what if pets can have multiple toys, and we'd like to keep track of both the name and type of each toy? In this case, we can make the "Toys" column both nested and repeated.



In the more\_pets\_and\_toys table above, "Name" and "Type" are both fields contained within the "Toys" STRUCT, and each entry in both "Toys.Name" and "Toys.Type" is an ARRAY.



Let’s look at a sample query.

query = “””

SELECT Name AS Pet\_Name,

t.Name AS Toy\_Name,

t.Type AS Toy\_Type

FROM `bigquery-public-data.pet\_records.more\_pets\_and\_toys`,

UNNEST(Toys) AS t

“””

------------ ------------- ------------

Pet\_Name Toy\_Name Toy\_Type

------------ ------------- ------------

Moon McFly Frisbee

Moon Scully Bone

Moon Pusheen Rope

Ripley Fluffy Feather

Ripley Robert Ball

Napoleon Eddy Castle

Since the "Toys" column is repeated, we flatten it with the UNNEST() function. And, since we give the flattened column an alias of t, we can refer to the "Name" and "Type" fields in the "Toys" column as t.Name and t.Type, respectively.