**Introduction to relational databases**

You have already mastered the art of importing all types of single files in Python: congratulations! However, to earn your daily bread and butter as a Data Scientist, you'll be required to interact with more complex data structures, such as relational databases.

**What is a relational database?**

What is a relational database? It's a type of database that is based upon the Relational model of data, first described by Ted Codd in the late 1960s. Before getting too theoretical, however, let's check out at an illuminating example,

**Example: Northwind database**

the Northwind Traders database, a synthetic database that contains sales data for a fictitious company. Firstly, a database consists of tables. Here you can see 3 tables from the Northwind database:

**Example: Northwind database**

'Orders',

**Example: Northwind database**

'Customers' and 'Employees'. So what's a table? A table generally represents one entity type,**.**

**The Orders table**

such as 'Order' . Notice that this table looks a great deal like a dataframe. That's the point. In a relational database table,each row or record represents an instance of the entity type: in this case, each row is an Order.

Each column represents an attribute of each instance, such as 'OrderDate' in the case of 'Orders'. In this sense, a table is entirely analogous to a dataframe. It is essential that each row contain a unique identifier, known as a primary key, that we can use to explicitly access the row in question. In our 'Orders' table, you can see that

**The Orders table**

the key is 'OrderID' the first column. But recall that a database consists of many tables! The really cool thing about relational databases is not merely that you have a bunch of tables,

**Tables are linked**

but that the tables are linked. How this linking works is ultra-intuitive: see that the 'Orders' table has both a column called 'CustomerID' and one called 'EmployeeID'. These columns correspond precisely to the primary keys in the

**Tables are linked**

'Customers' and 'Employees' tables, respectively. So, given an Order, you can immediately look up the details of the relevant Customer or Employee. This is cool because it means that you don't need to store all the Customer details, such as first name, last name, company with every order that they place: you merely need to look it up in the 'Customers' table. This saves an incredible amount of space!

**Relational model**

As stated earlier, the relational database model was originally proposed by "Ted" Codd and has been widely adopted. There is a great deal to theory but it is most neatly summarized in Codd's 12 Rules, also known as Codd's 12 Commandments, which he developed in the early 1980s to combat what he viewed as a dilution of his original relational database vision. Codd's 12 Rules actually consist of 13 rules but they are zero-indexed, that is, the first rule is zero-indexed. If that makes you laugh, you're definitely a geek like me! These 13 rules were defined to describe what a Relational Database Management System should adhere to in order to be considered relational.

**Relational Database Management Systems**

Among the most popular of such systems are PostreSQL (commonly called Postgres),

**Relational Database Management Systems**

MySQL and SQLite, all of which use the SQL query language. In fact, SQL itself is actually an acronym for Structured Query Language, which describes how you communicate with a database in order to both access and update the information it contains. The term "querying" is really just a fancy way of saying getting data out from the database. Next up, you'll learn how to connect to databases but before that, let's make sure that you have a solid conceptual grip on the relational model.