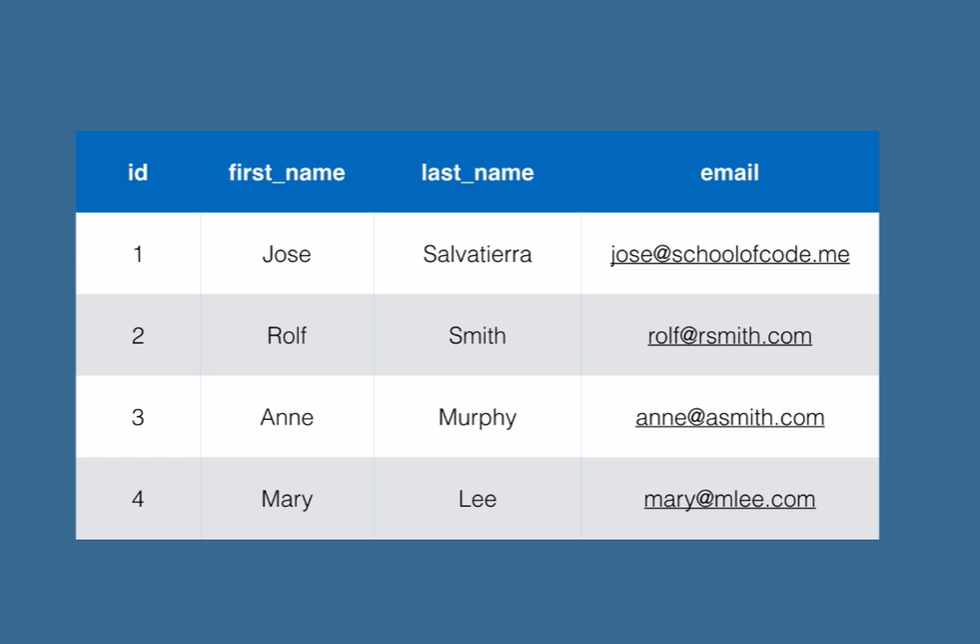
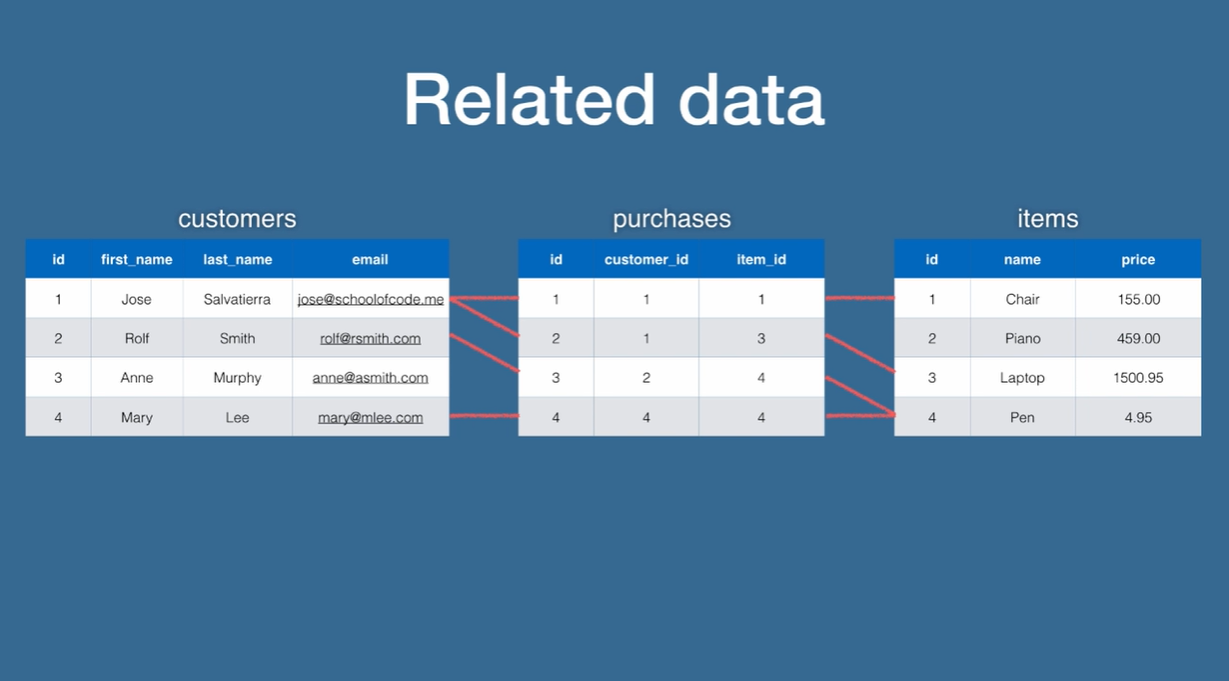
In this lesson we will be looking at Database. When we talk about databases, we tend to think these things,



This is a table and we can see it has got 4 columns, id, first\_name, last\_name, and email. We have got names of 4 different users with their ids, names and email addresses. This is a very normal table to have in a database system. Now a days we are not limited to storing data in a database and retrieving it. There are bunch of things we can do.

*Tables*

* *Tables should store data related to one thing, such as customers*
* *If the database belongs to a larger entity, such as business, then it may store related data in separate tables*
* *For example: customers, items, purchases, stores etc.*



We have three tables of related data here, we have got a customer’s table, it has got 4 columns like before and each of these columns are storing customer information where the customers are defined in rows. We have 4 customers in our customers’ table. Now, let’s move on the items’ table, the items’ table defines information of each specific items in its each specific row. An item has an id, a name and a price in its three columns. Finally, the most interesting table is the purchases’ table, this table has been placed at the middle for a reason. The purchases table has three columns, but now there is no relation of the purchase itself, for example how much a customer paid in the purchase? But rather an id for the purchase, then the id of the customer who made the purchase, and the item id of the item the customer bought in the purchase.

This is assuming each customer can only buy one item per purchase. For example, in our first purchase customer with id number 1, bought the item with id number 1. The red line links up the first customer with the first item. In the second purchase we have the first customer buying the third item, item number 3. In the third purchase we have customer with id number 2, buying item number 4, we have customer number 4 buying item number 4.

As you can see there are relations between these tables, these data are related to one another, and we call this relational data. The purchases’ table performs relationships between customers and items, but in a more basic way the customers’ table is related to purchases’ table and the purchases’ table is related to the items’ table, so there’s no relationship between customers and items. There are only relationships between customers and purchases; and between purchases and items. That is a key distinction.

*Related data*

* *We can model these relationships between tables and columns in database systems designed for it*
* *There are other database systems which do not allow for relational data*
* *They have other advantages, such as increased speed or simpler scalability*

*Scalability*

* *A Database Management System is installed and runs on a computer*
* *Applications (such as our Python apps) connect to that computer and ask it to retrieve data for them*
* *If there are too many requests for data, then the computer can end up unable to cope with the load*
* *Relational Database Management Systems (RDMS) tend to not scale as well as non-relational Database Management Systems*
* *Scalability is important because it enables the servers to serve more clients (your applications)*
* *However, Relational Database Management Systems (RDMS) allow us to interact and retrieve data easily*
* *For example, using the tables shown earlier, it is easy to get “revenue generated per customer” using a Relational Database Management System than a Non-relational Database Management System*

*Relational and Non-relational Database Systems*

* *If our data is simple and there are few to no relationships between them then we do not have to use a relational system*
* *If our data is related then expect that as the applications grow and more data is collected, there will be more relationships*
* *With related data: use a relational system*
* *If you are not sure about the data being relational or not then use a relational system, but if you are sure that your data is non-relational then use a non-relational system*

*Other elements of RDBMS’s*

* *We will be looking at other elements that are part of RDBMS’s:*
* *Views*
* *Functions*
* *Aggregation*
* *JOINs*
* *And more!*