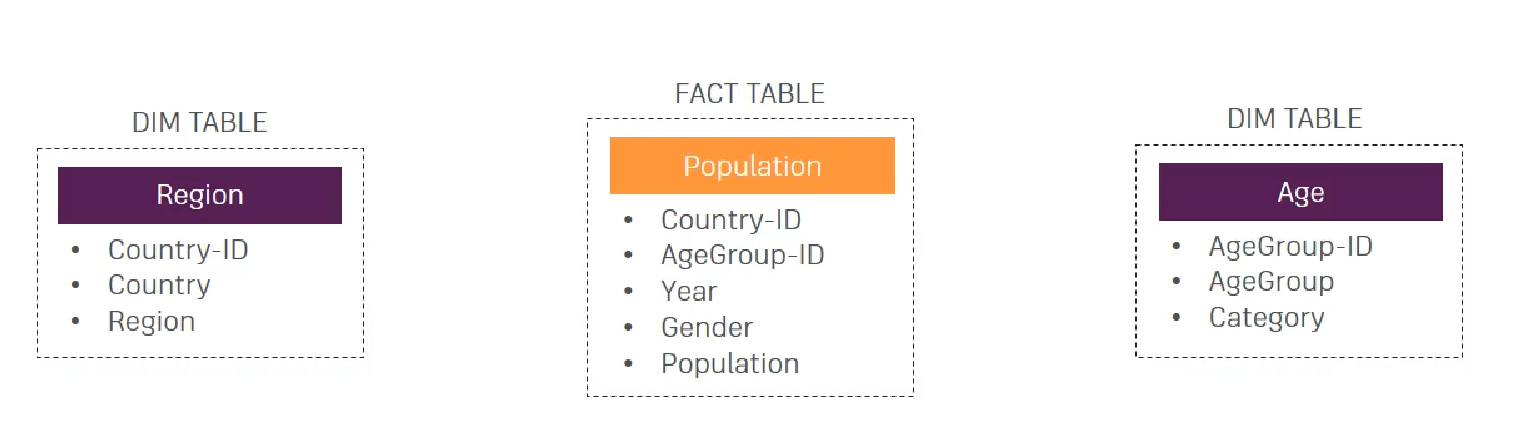
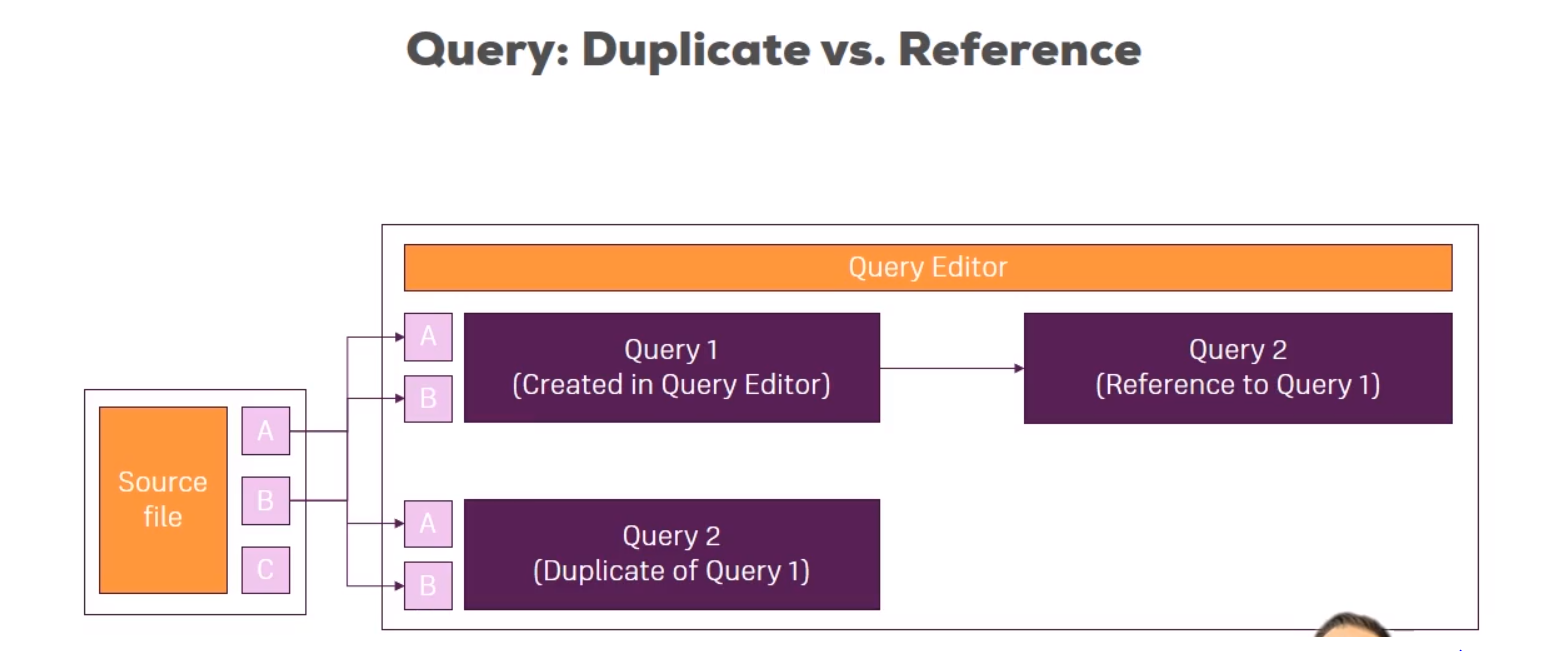
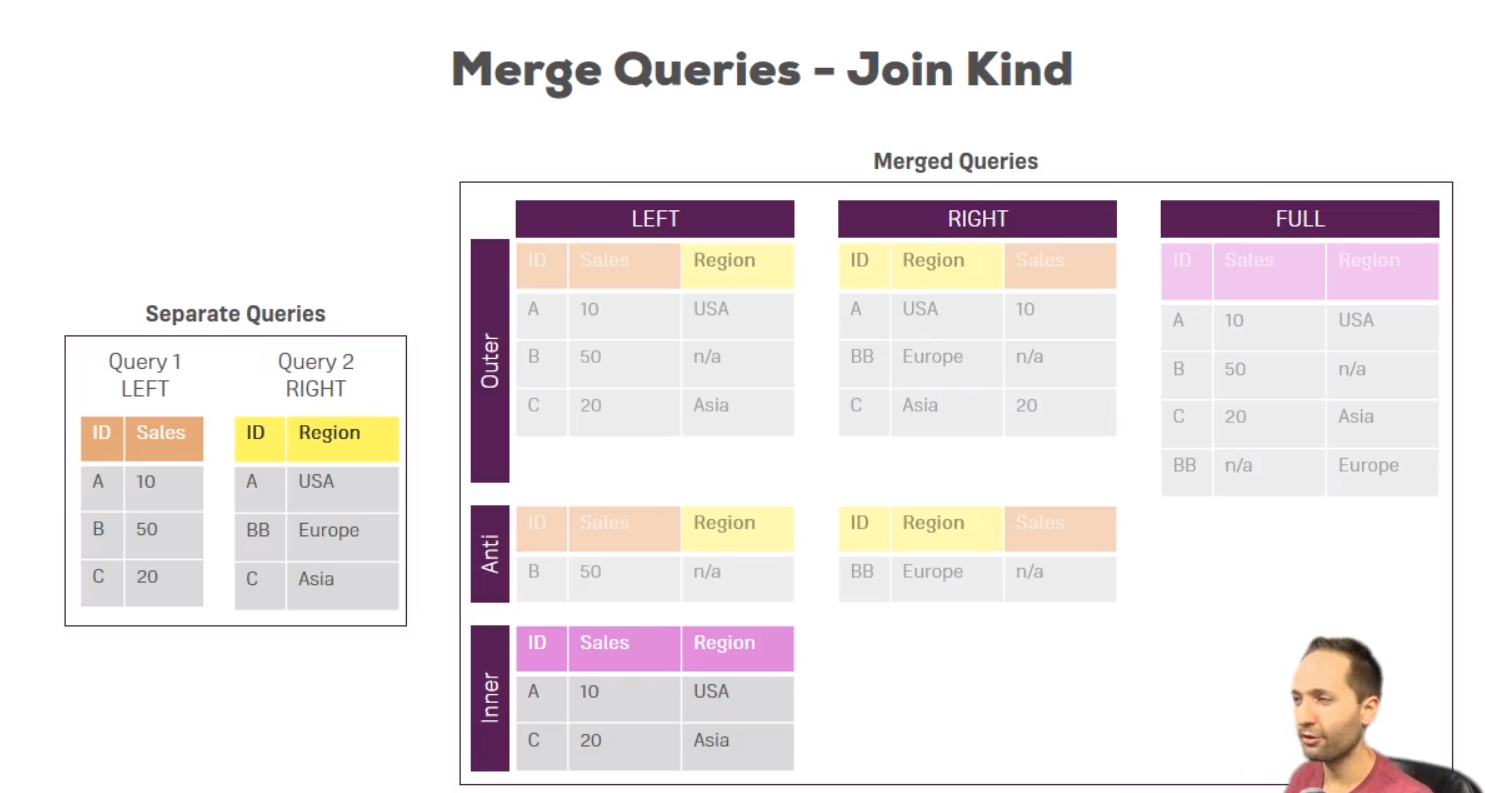
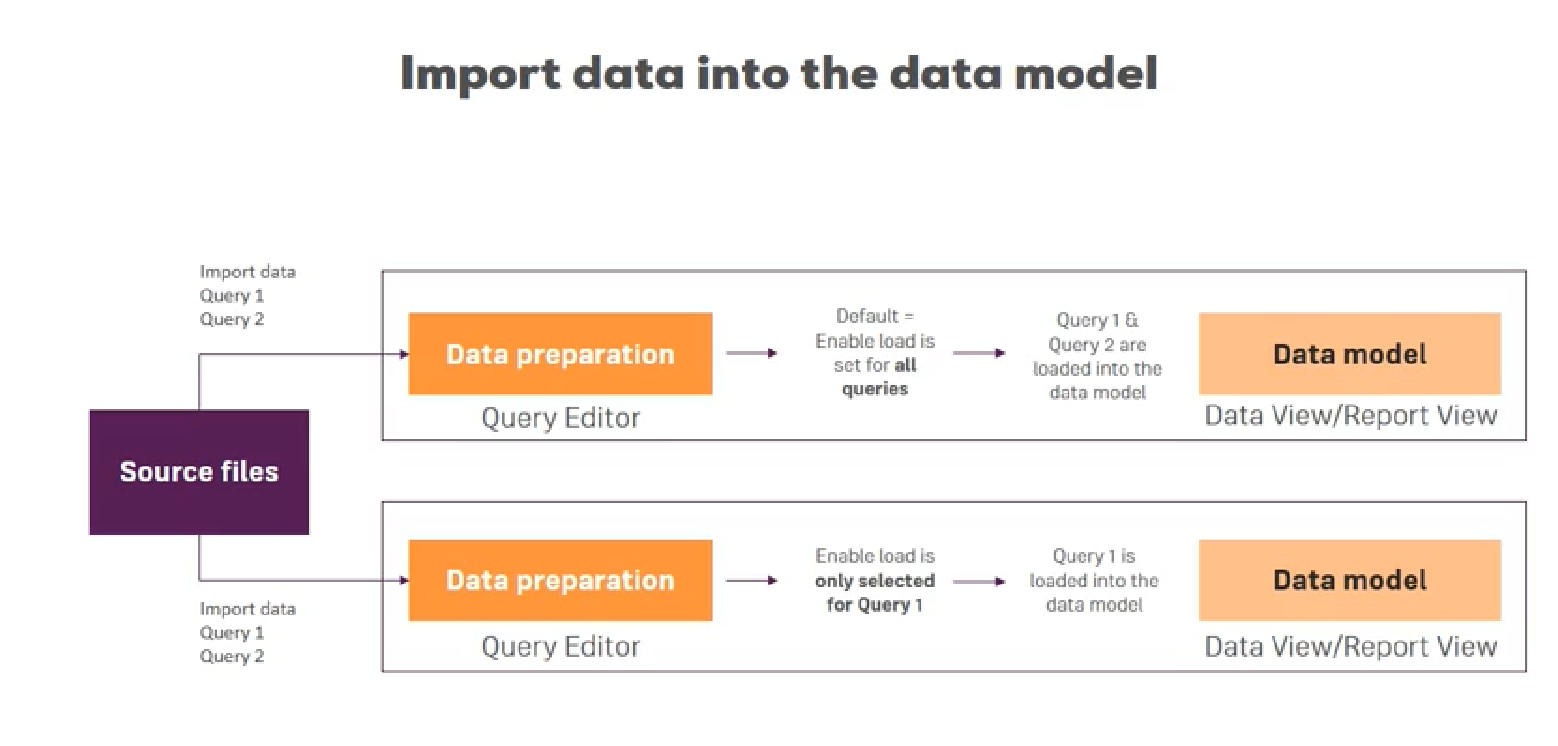
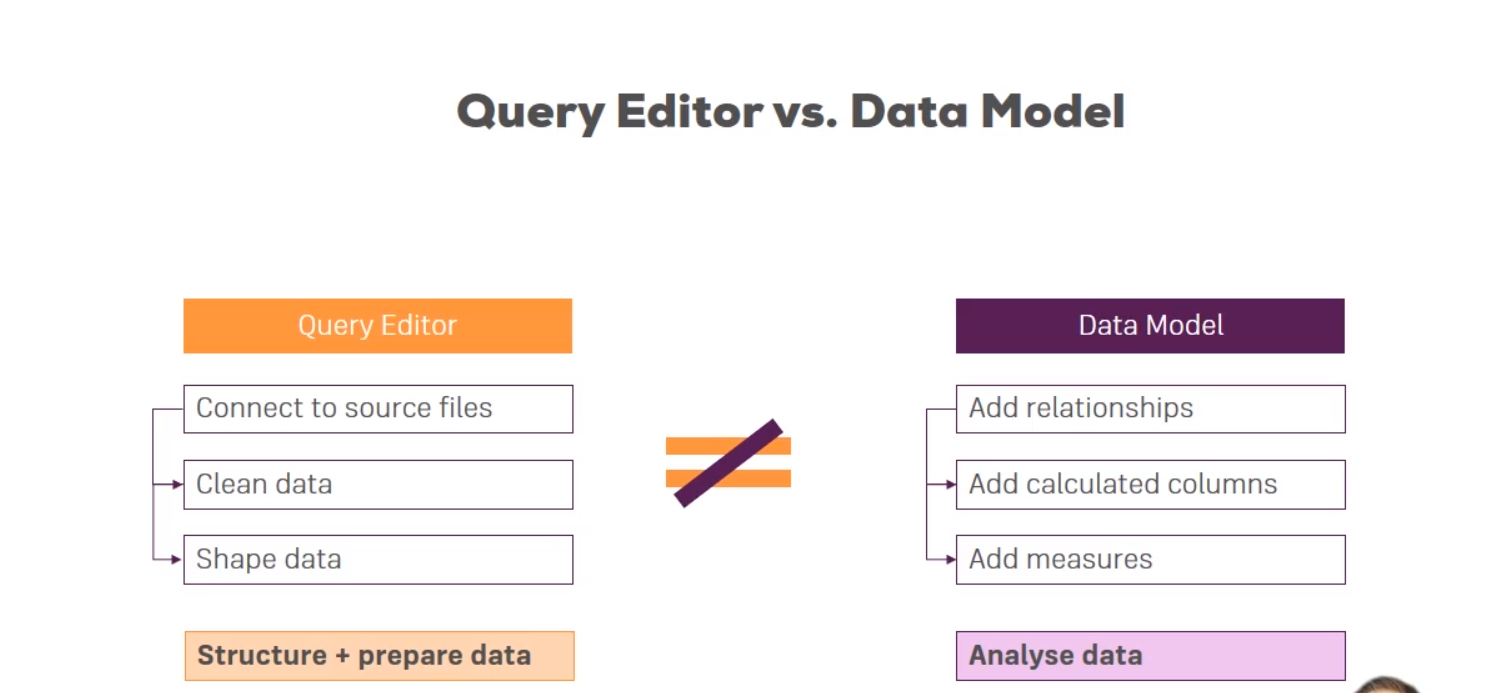
I want to model the data based on star schema with central FACT table and spanning with 2 DIM tables.

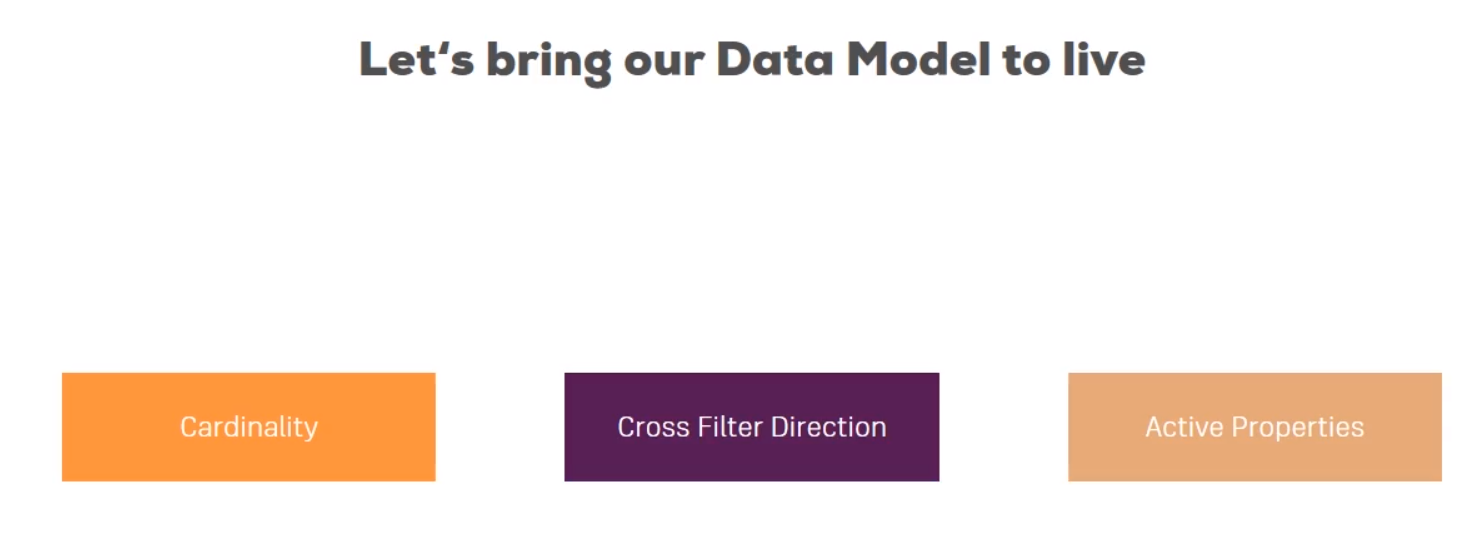


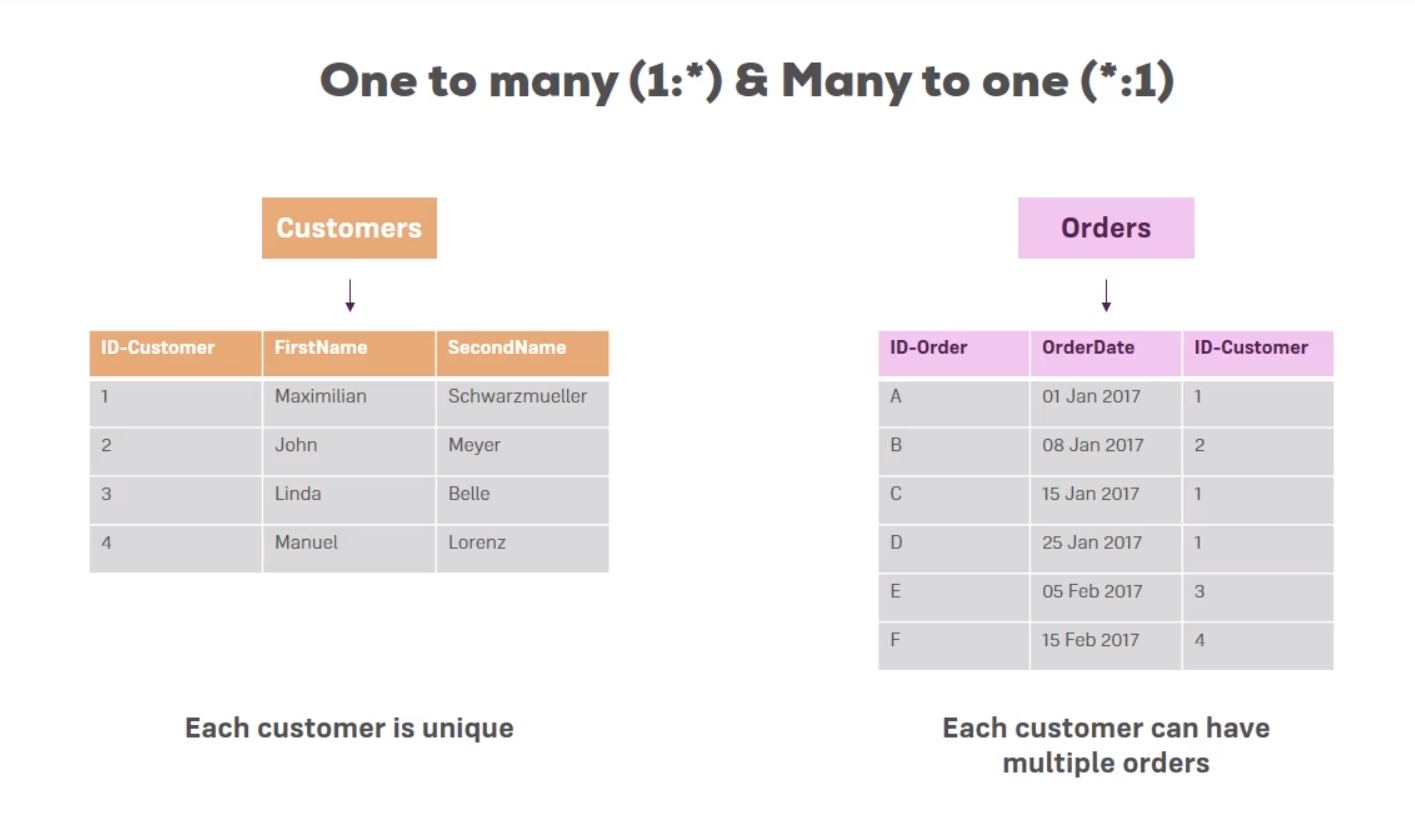


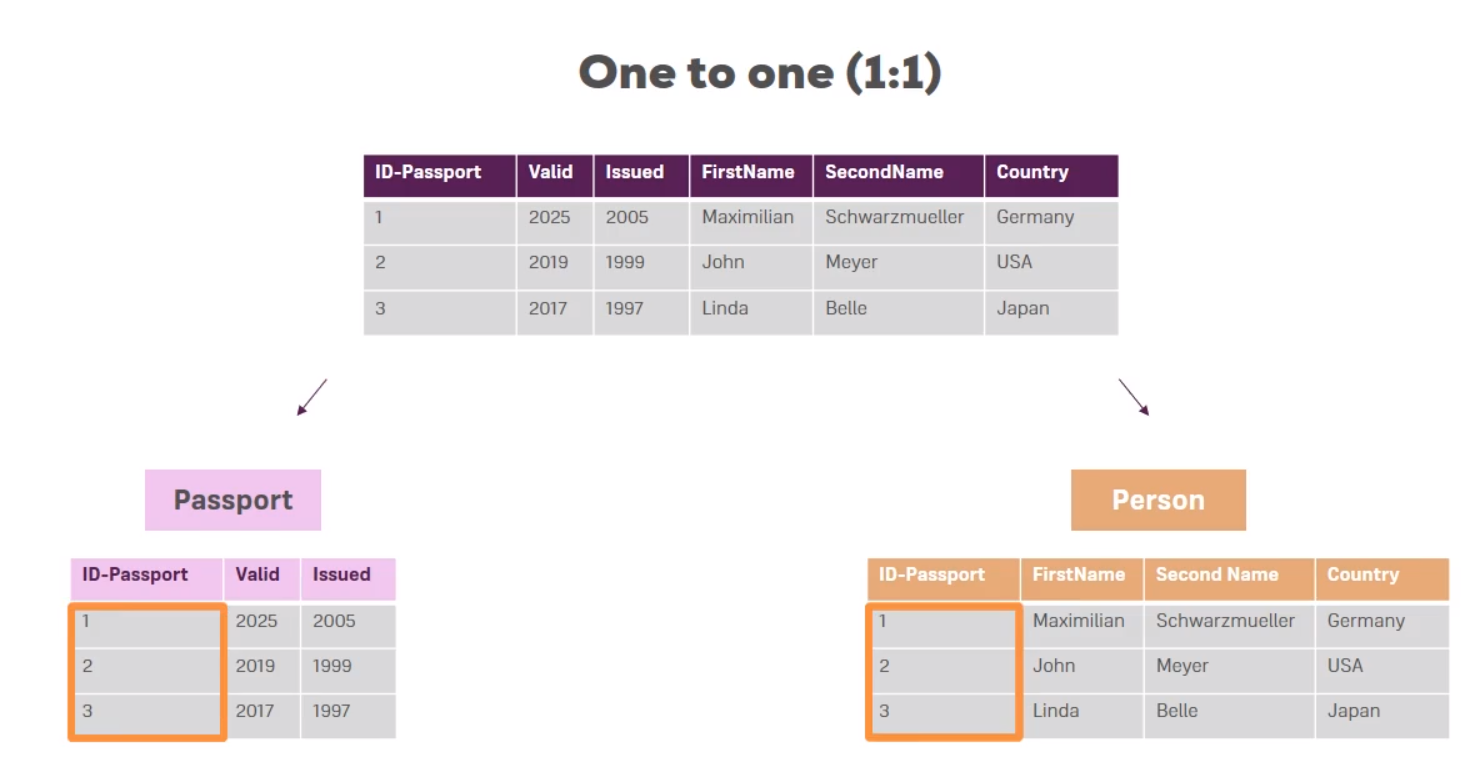


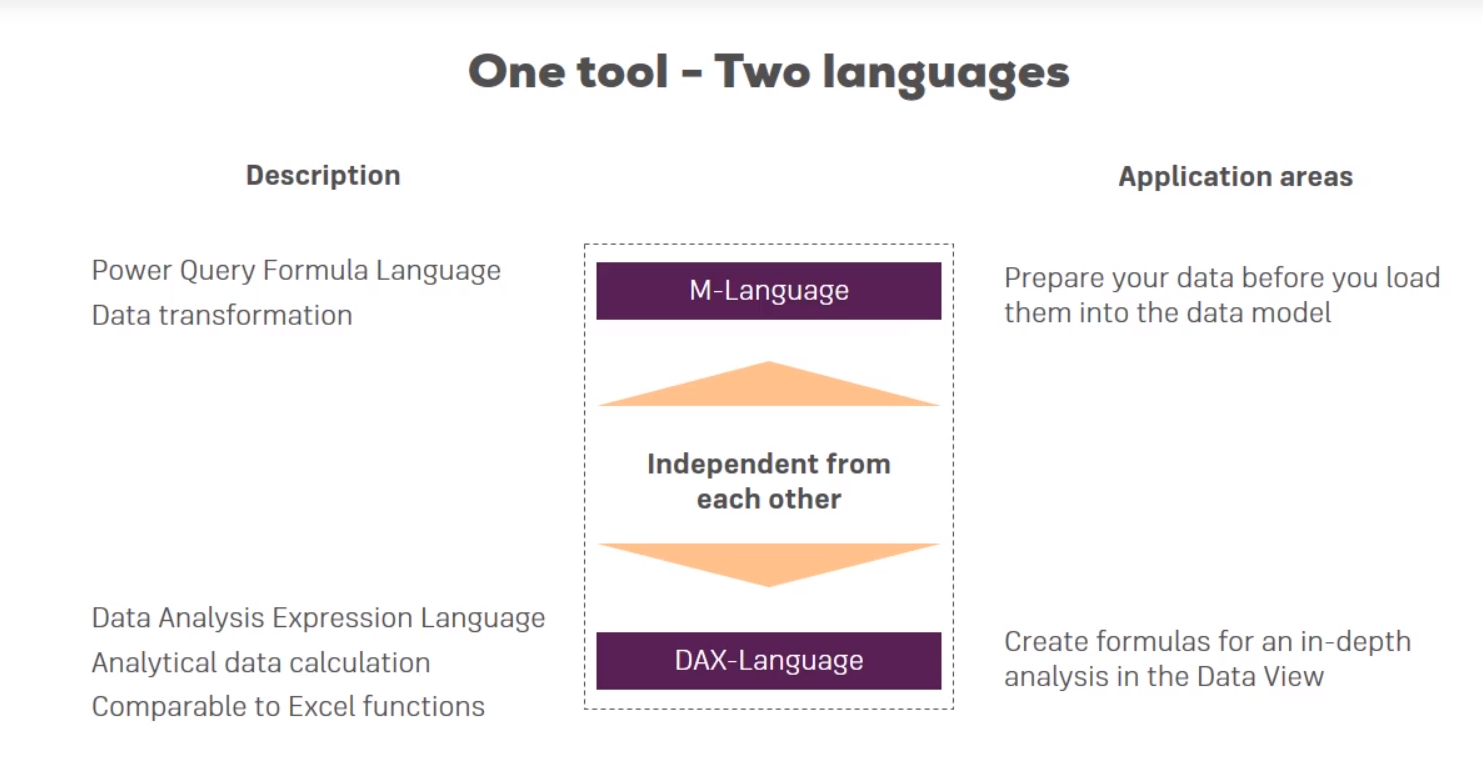


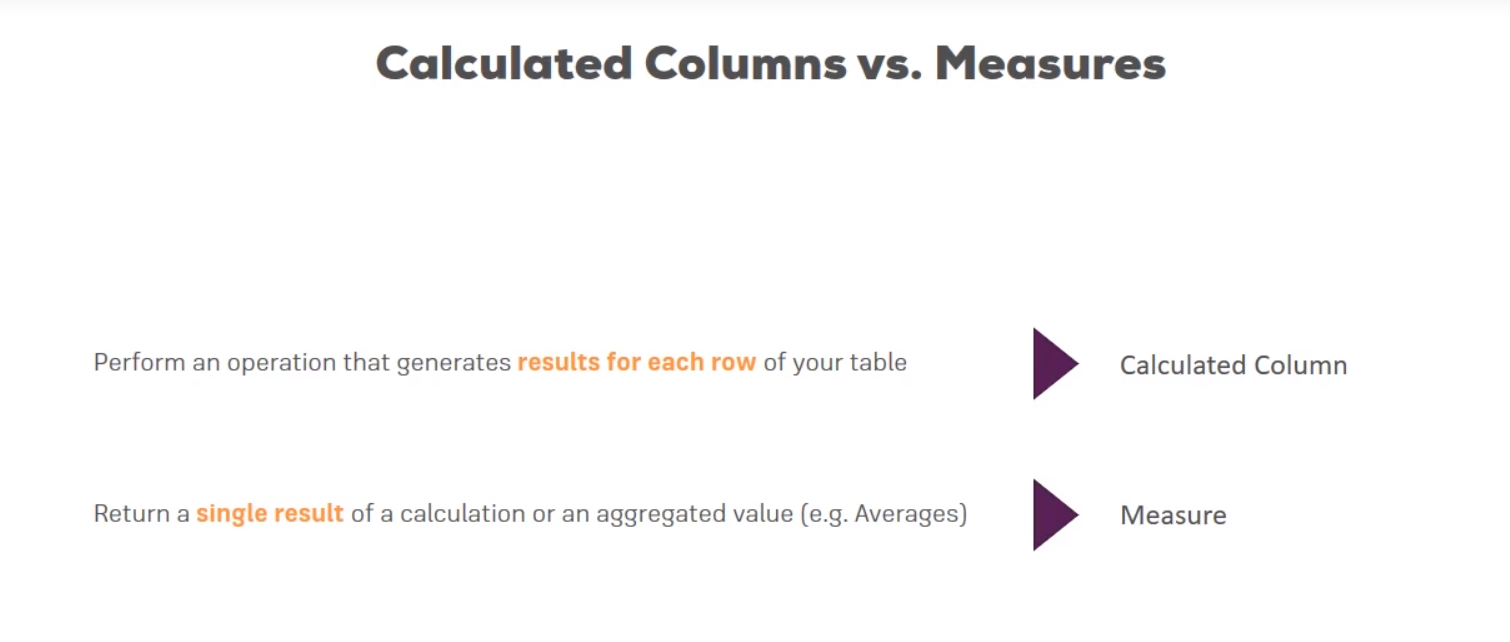


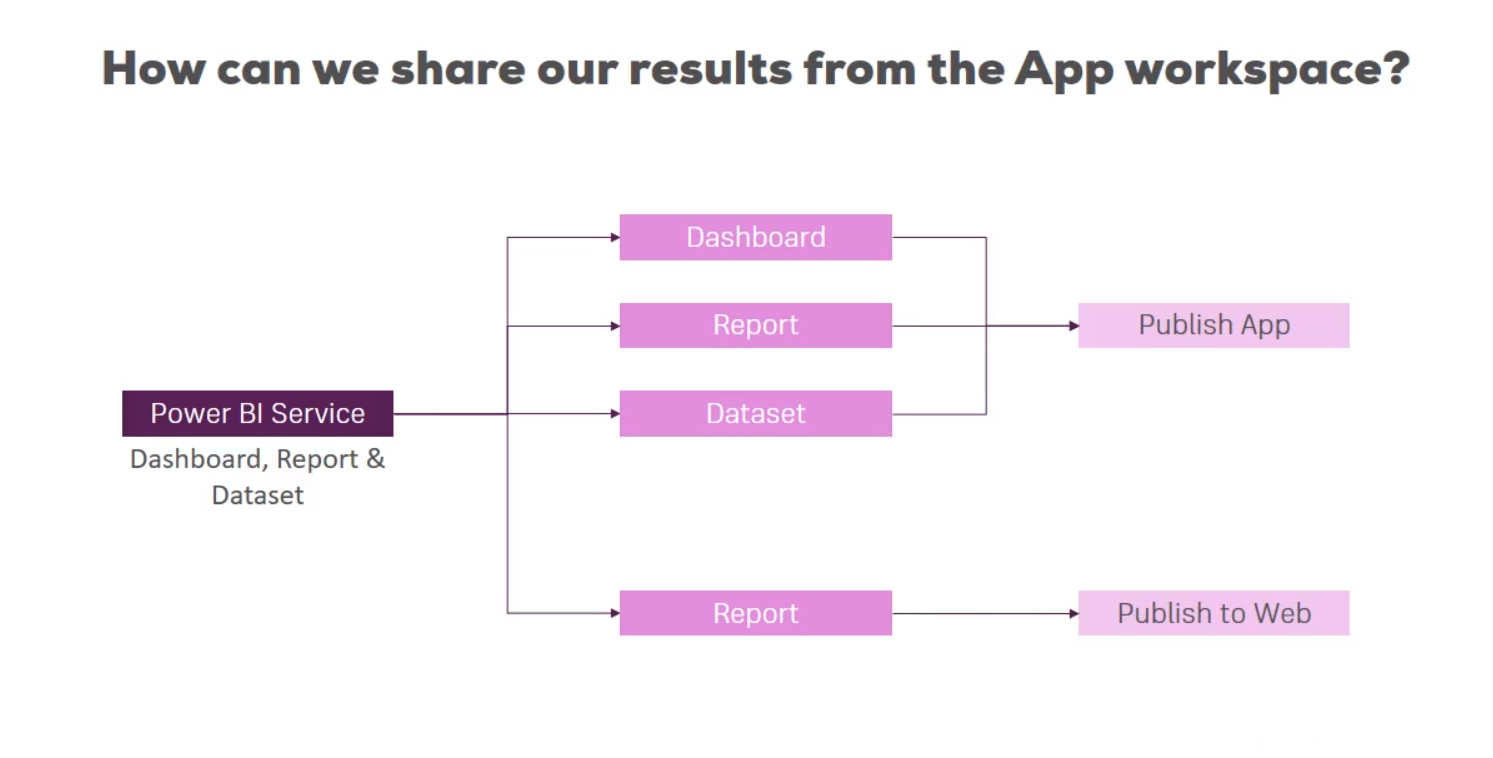




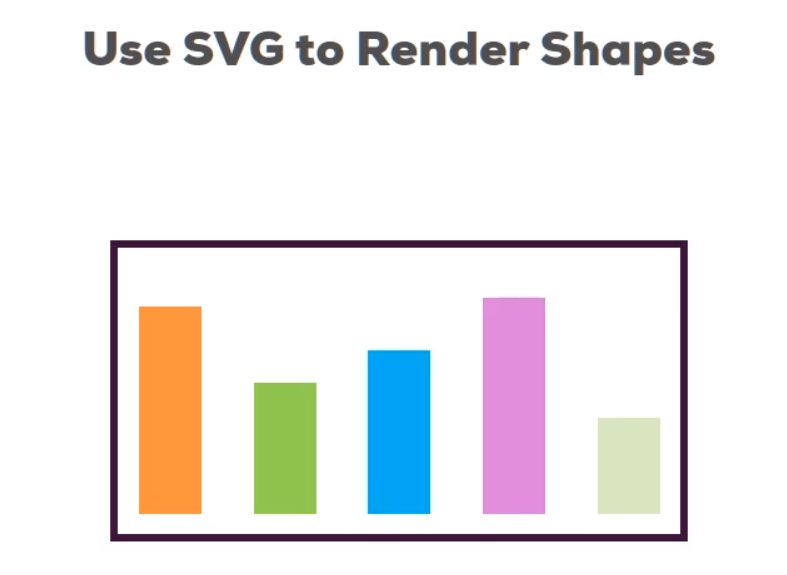




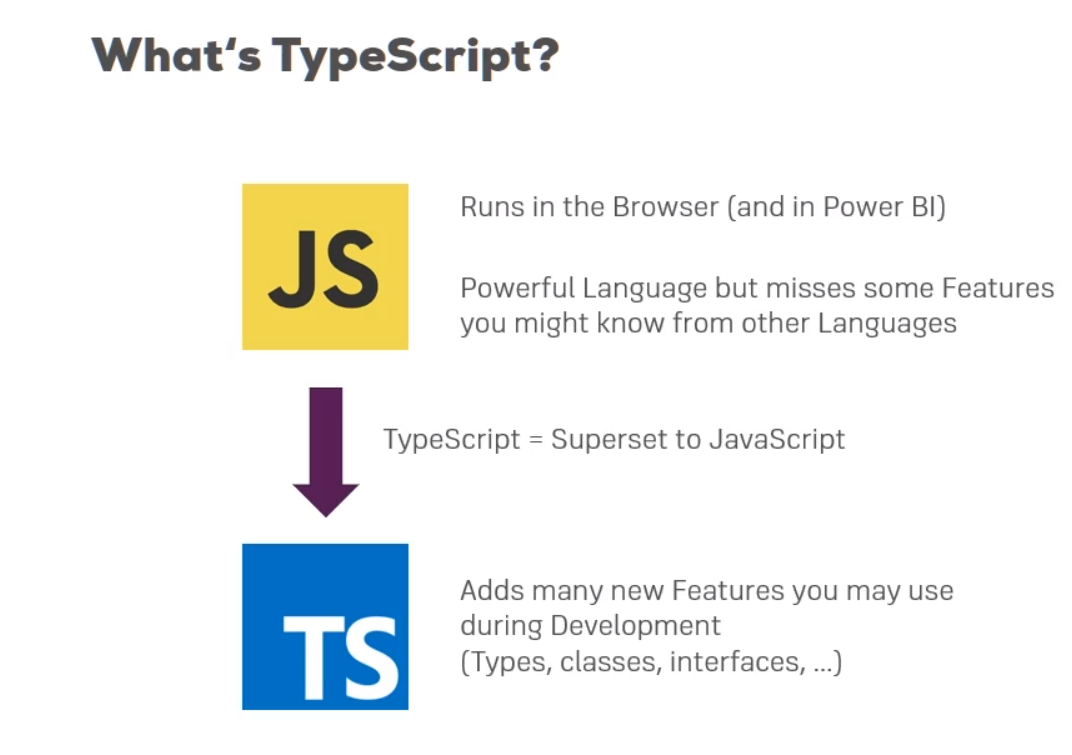




Scaler vector graphics

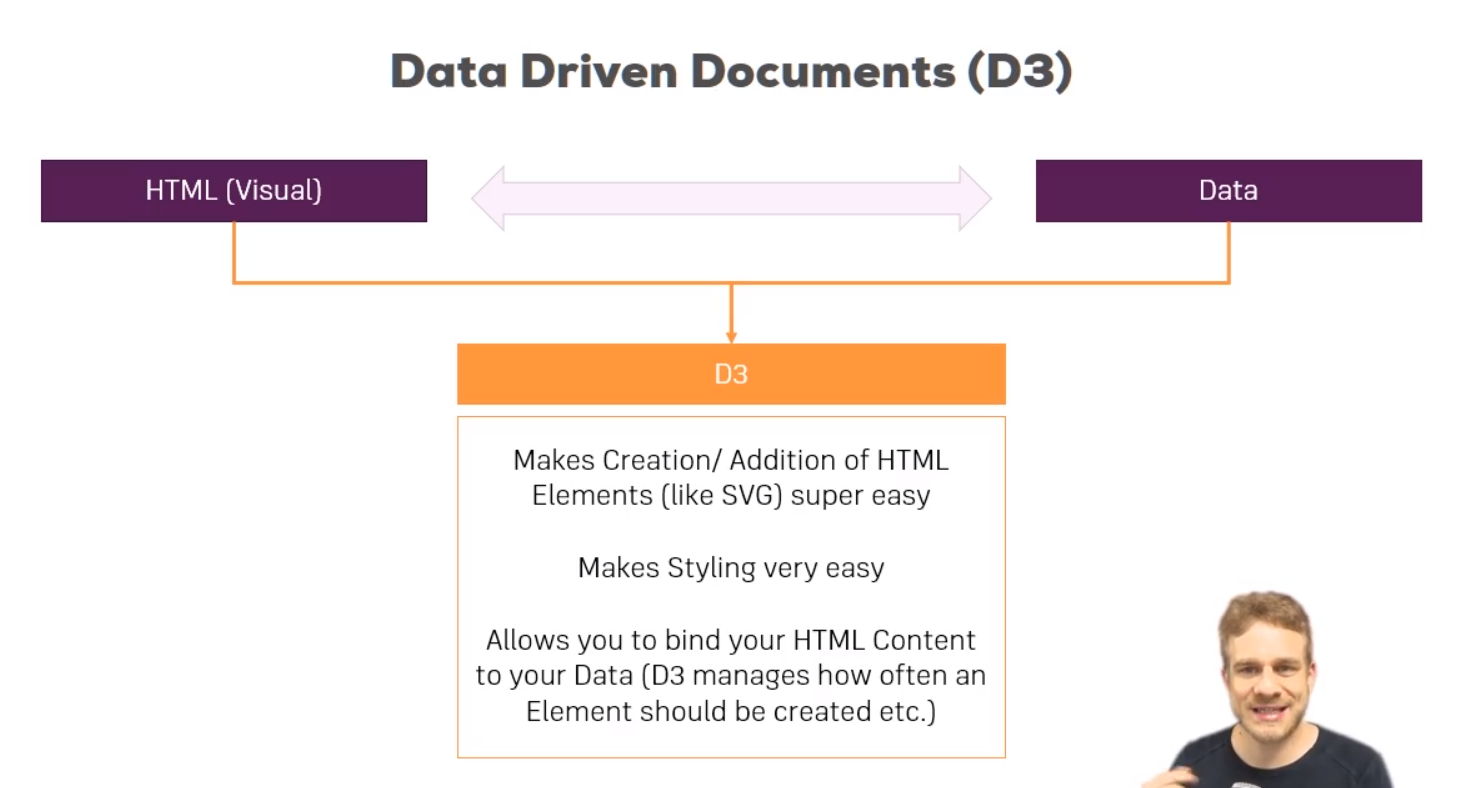






Typescript code is compiled to javascript

We have data to be included in the document or webcode that we are making and we use javascript for it so we can simply use additional javascript package caleed D3 which allows easily to manipulate the drawn html code at runtime but without the need to deep dive into the hard code. Instead you just bind some data to the shapes which changes with every new addition of data element.



NodeJS is a server side language which uses javascript on the server.

We use javascript and typescript to build our visual but it is done on client side together with html and css.

Nodejs needs to be installed for 2 reasons:

1. It has NPM (node package manager) which is a great tool for handling packages as you need for development. The power bi developer tools is such a package which we install for the node package manager and makes installation of package and management of dependencies more easier.
2. We also need it because we can get the behaviour of building our custom visual and then using the developer tools to run a command to create a little development server which will use NodeJS in the background to host our custom visual, so that we can log into our power bi service and get live updating of our visual whilst we are on it.

\_\_\_\_\_\_\_\_.less file in code used as a kind of css where you can add styles for the visual that you create using developer tools.

Capabilities.json file is the file in which we can configure which options we can see on the right side panel in power bi service related with a visual we are creating

Package.json stores a reference to all dependencies we use

Pbiviz.json file configures our custom visual with properties like id and catalog and info about author. Here we can also link the icon we use for the visual, style files we use, and connect to external javascript libraries

Tsconfig.json file configures how typescript gets compiled

To start using d3, we install it in current folder and then in the project, in the pbiviz.json file, in the externalJS section mention "node\_modules/d3/d3.min.js",

Because the installed dependency d3 shows up in node modules.

Since we are using typescript, we need to install a bridge from javascript d3 package to typescript code ecause typescript does not recognize the shape or the schema that d3 uses. These translation files are available for download, to do it, in powershell type npm install @types/de@3.5.35 –save-dev.

Now in tsconfig.json add

"typeRoots":["./node\_modules/@types"],

Y this we tell typescript that whenever you are encountering a code that you don’t recognize, like us calling d3, then have a look into @types folder to see if you can find some translation for it there.

After creating the chart if we want to be able to drag and drop data to right hand side fields like axis and legends we have to make changes to capabilities.json file as it helps in accessing the capabilities of the project which here is interacting with the chart.

In capabilities.json file, dataroles is what we display to user

The other fields here are internal mappings, name simply means how we can use it internally and kind is important. Grouping means this is a discrete value basically, so this is by which our data is grouped whereas measure means this is the value we can actually work with, we want to display so to say.

To limit the input of data, in capabilities.json add the following code to dataviewmappings

"dataViewMappings": [

{

"conditions":[

{

"category":{

"max":1

},

"measure": {

"max":1

}

}

]

It is for adding the limit that category and measures have at most 1 value.