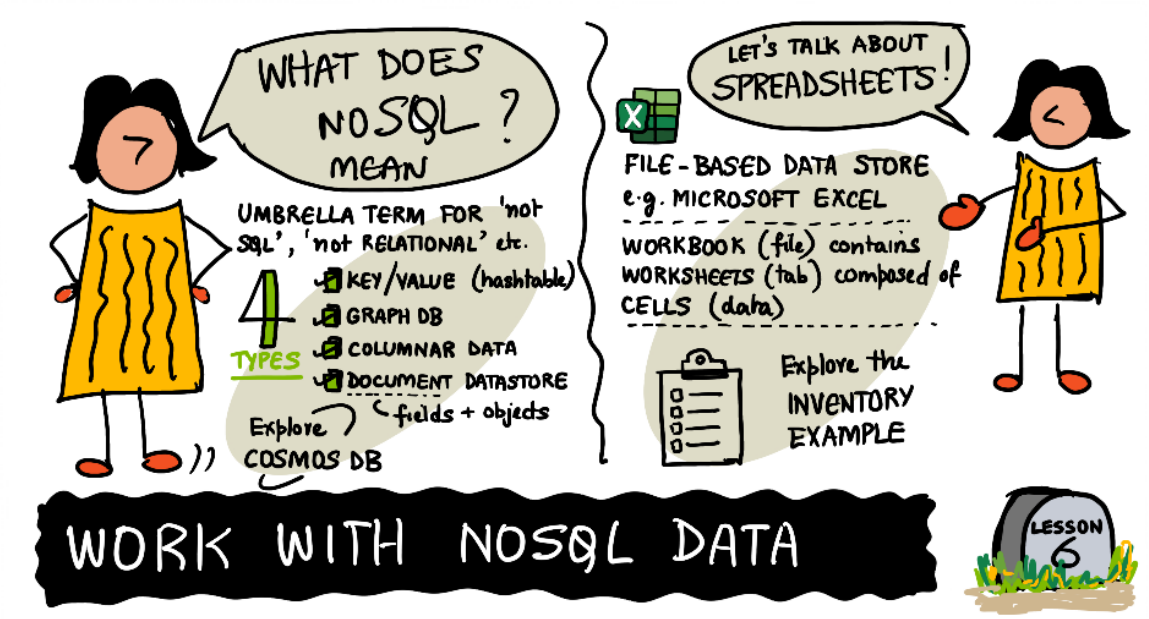
Working with Data: Non-Relational Data



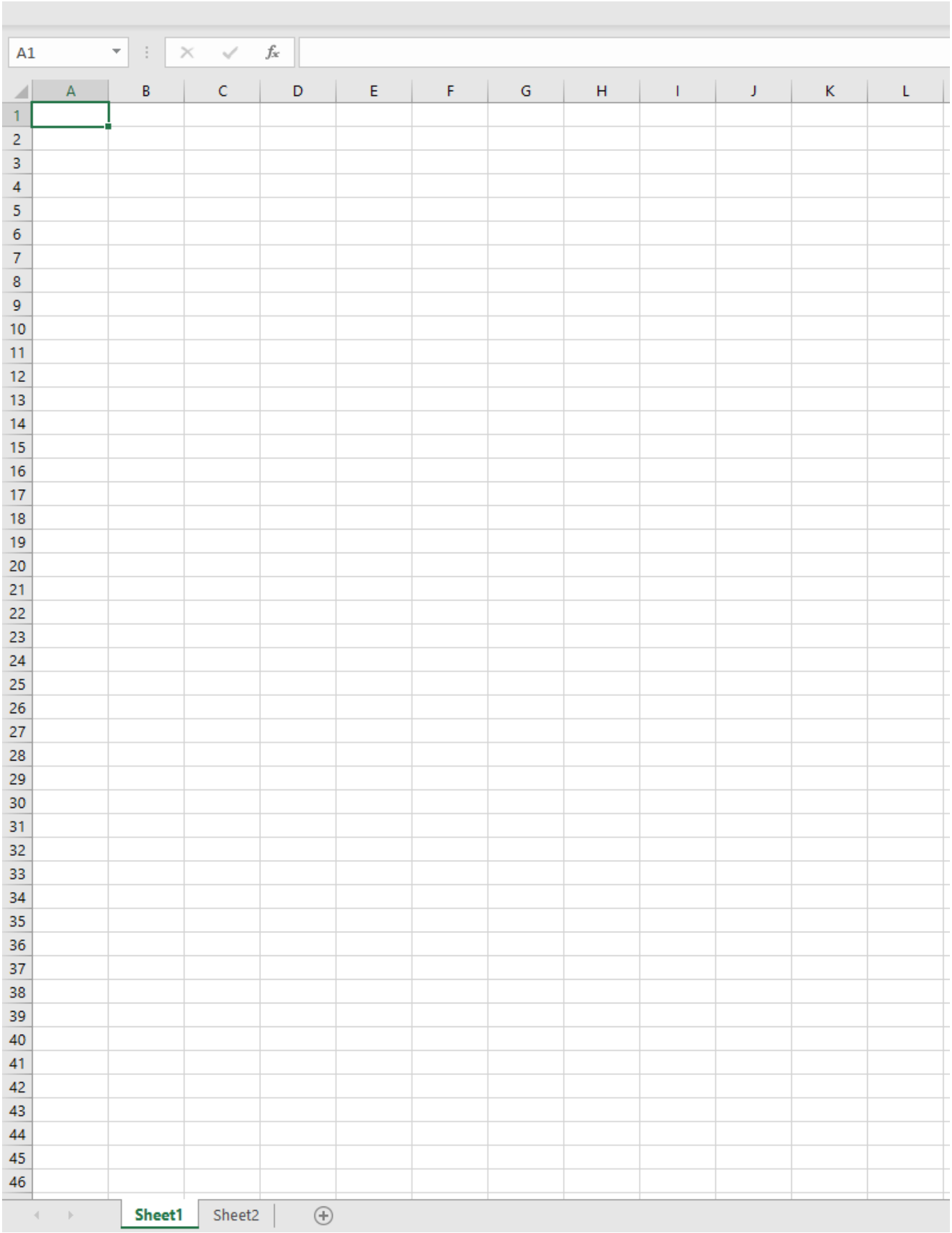
Working with NoSQL Data - *Sketchnote by @nitya*

Pre-Lecture Quiz

Data is not limited to relational databases. This lesson focuses on non-relational data and will cover the basics of spreadsheets and NoSQL.

Spreadsheets

Spreadsheets are a popular way to store and explore data because it requires less work to setup and get started. In this lesson you'll learn the basic components of a spreadsheet, as well as formulas and functions. The examples will be illustrated with Microsoft Excel, but most of the parts and topics will have similar names and steps in comparison to other spreadsheet software.



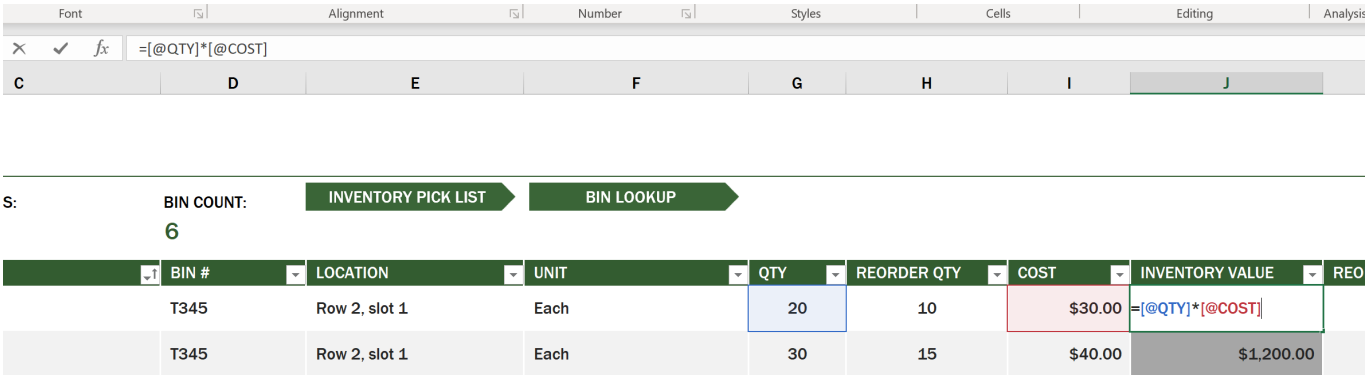
A spreadsheet is a file and will be accessible in the file system of a computer, device, or cloud based file system. The software itself may be browser based or an application that must be installed on a computer or downloaded as an app. In Excel these files are also defined as **workbooks** and this terminology will be used the remainder of this lesson.

A workbook contains one or more **worksheets**, where each worksheet are labeled by tabs. Within a worksheet are rectangles called **cells**, which will contain the actual data. A cell is the intersection of a row and column, where the columns are labeled with alphabetical characters and rows labeled numerically. Some spreadsheets will contain headers in the first few rows to describe the data in a cell.

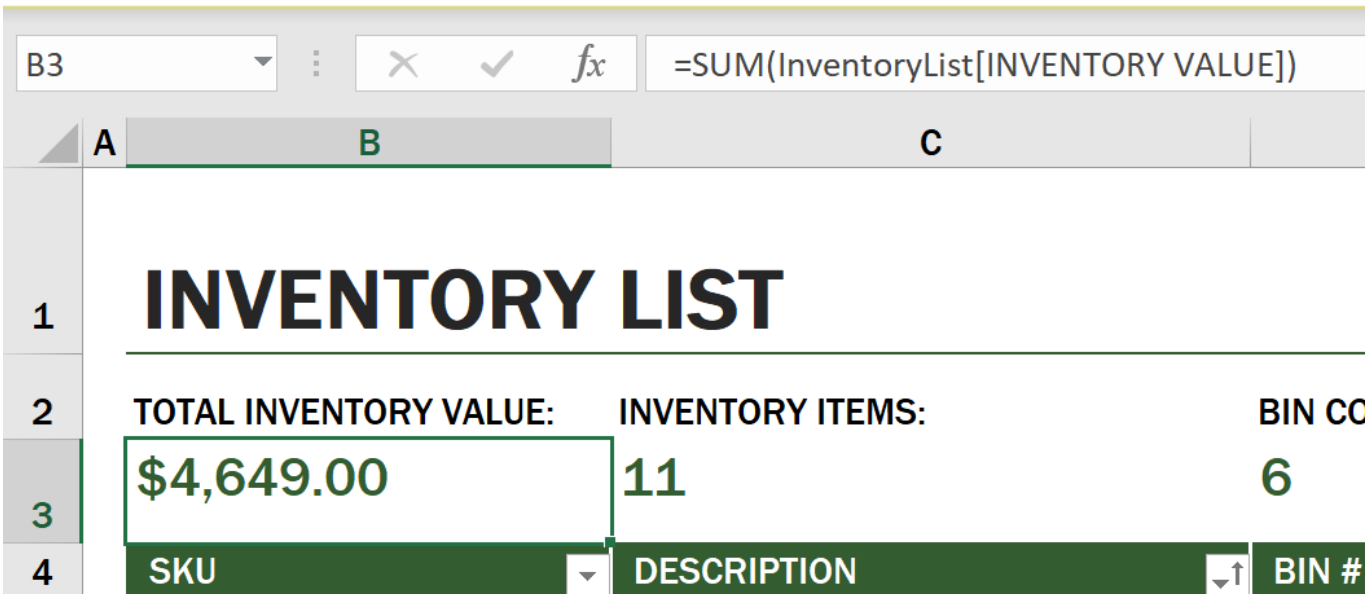
With these basic elements of an Excel workbook, we'll use and an example from Microsoft Templates focused on an inventory to walk through some additional parts of a spreadsheet.

Managing an Inventory

The spreadsheet file named "InventoryExample" is a formatted spreadsheet of items within an inventory that contains three worksheets, where the tabs are labeled "Inventory List", "Inventory Pick List" and "Bin Lookup". Row 4 of the Inventory List worksheet is the header, which describes the value of each cell in the header column.



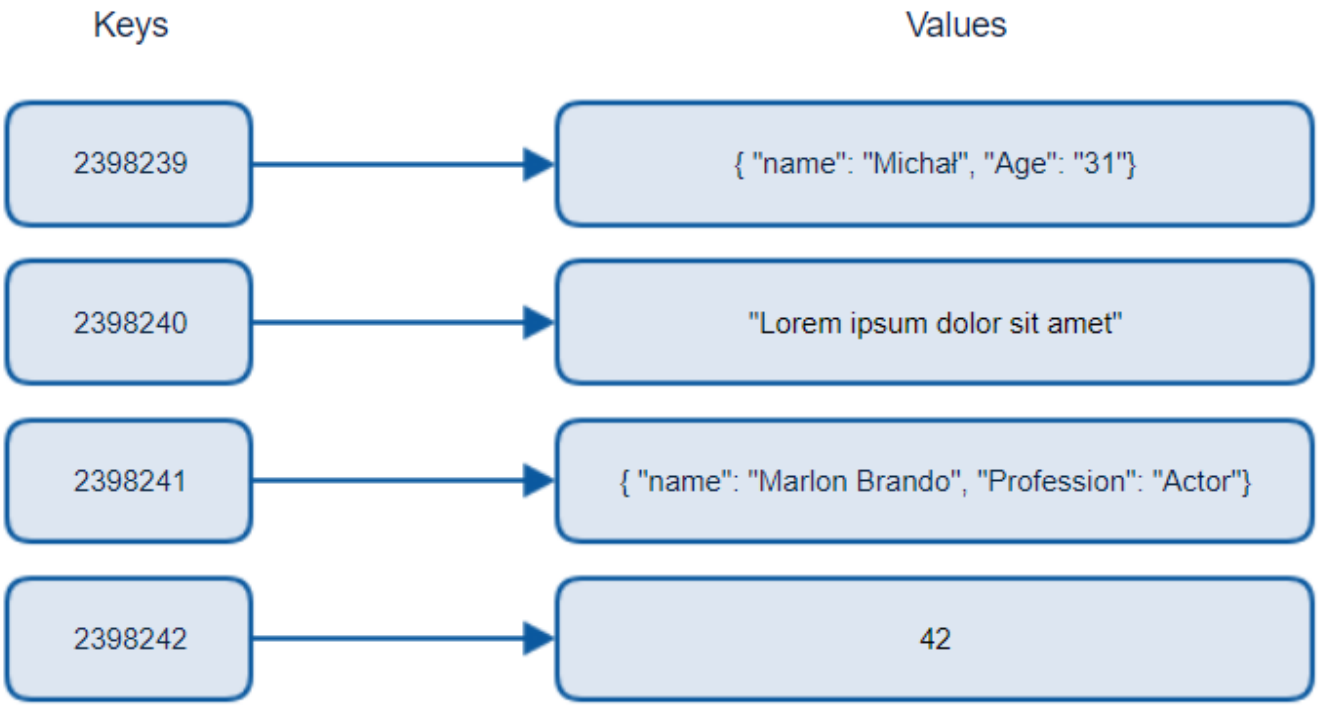
There are instances where a cell is dependent on the values of other cells to generate its value. The Inventory List spreadsheet keeps track of the cost of every item in its inventory, but what if we need to know the value of everything in the inventory? **Formulas** perform actions on cell data and is used to calculate the cost of the inventory in this example. This spreadsheet used a formula in the Inventory Value column to calculate the value of each item by multiplying the quantity under the QTY header and its costs by the cells under the COST header. Double clicking or highlighting a cell will show the formula. You'll notice that formulas start with an equals sign, followed by the calculation or operation.



We can use another formula to add all the values of Inventory Value together to get its total value. This could be calculated by adding each cell to generate the sum, but that can be a tedious task. Excel has **functions**, or predefined formulas to perform calculations on cell values. Functions require arguments, which are the required values used to perform these calculations. When functions require more than one argument, they will need to be listed in a particular order or the function may not calculate the correct value. This example uses the SUM function, and uses the values of on Inventory Value as the argument to add generate the total listed under row 3, column B (also referred to as B3).

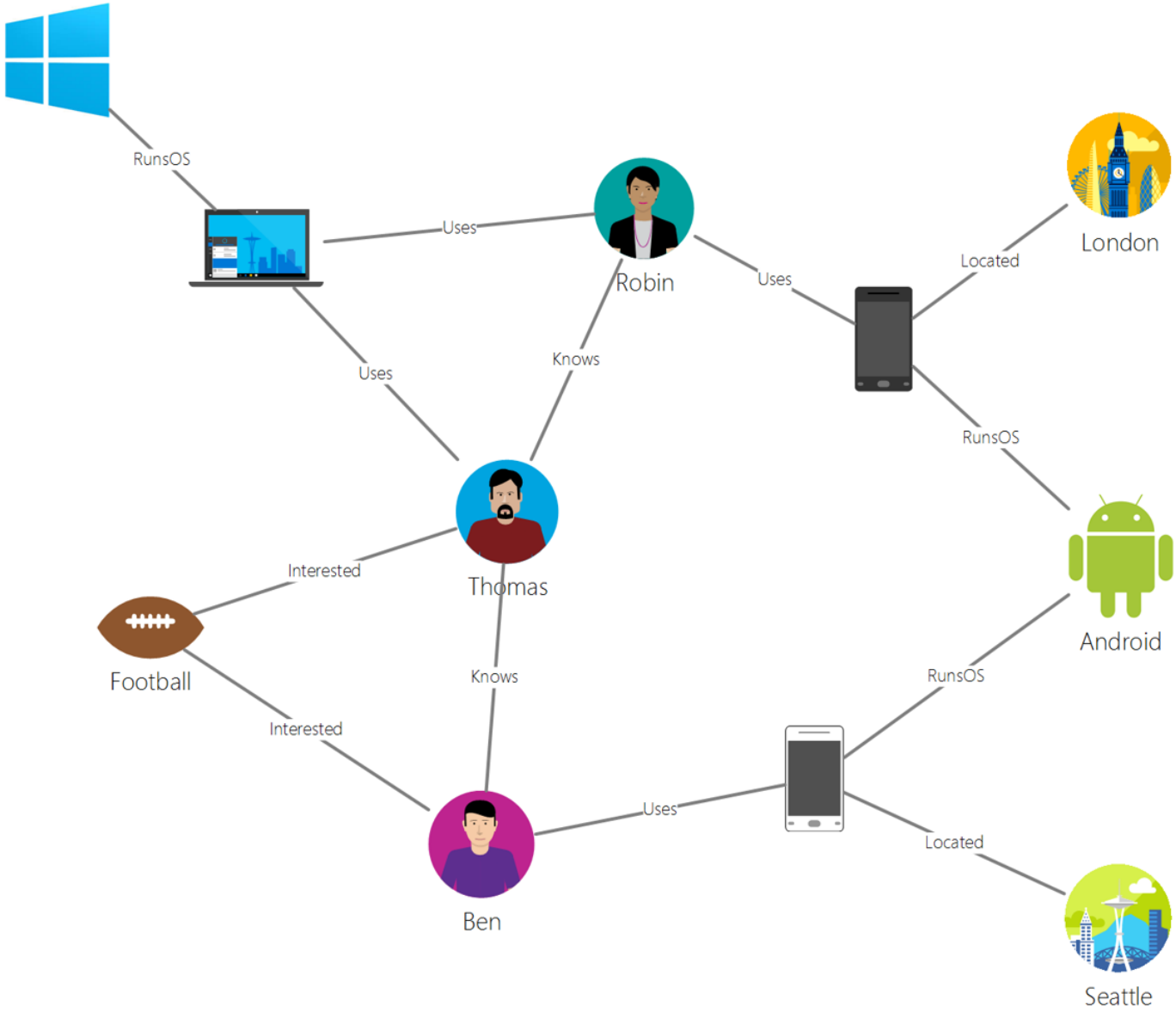
NoSQL

NoSQL is an umbrella term for the different ways to store non-relational data and can be interpreted as "non-SQL", "non-relational" or "not only SQL". These type of database systems can be categorized into 4 types.



> Source from Michał Białecki Blog

Key-value databases pair unique keys, which are a unique identifier associated with a value. These pairs are stored using a hash table with an appropriate hashing function.



> Source from Microsoft

Graph databases describe relationships in data and are represented as a collection of nodes and edges. A node represents an entity, something that exists in the real world such as a student or bank statement. Edges represent the relationship between two entities Each node and edge have properties that provides additional information about each node and edges.

CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

Columnar data stores organizes data into columns and rows like a relational data structure but each column is divided into groups called a column family, where all the data under one column is related and can be retrieved and changed in one unit.

Document Data Stores with the Azure Cosmos DB

Document data stores build on the concept of a key-value data store and is made up of a series of fields and objects. This section will explore document databases with the Cosmos DB emulator.

A Cosmos DB database fits the definition of "Not Only SQL", where Cosmos DB's document database relies on SQL to query the data. The previous lesson on SQL covers the basics of the language, and we'll be able to apply some of the same queries to a document database here. We'll be using the Cosmos DB Emulator, which allows us to create and explore a document database locally on a computer. Read more about the Emulator here.

A document is a collection of fields and object values, where the fields describe what the object value represents. Below is an example of a document.

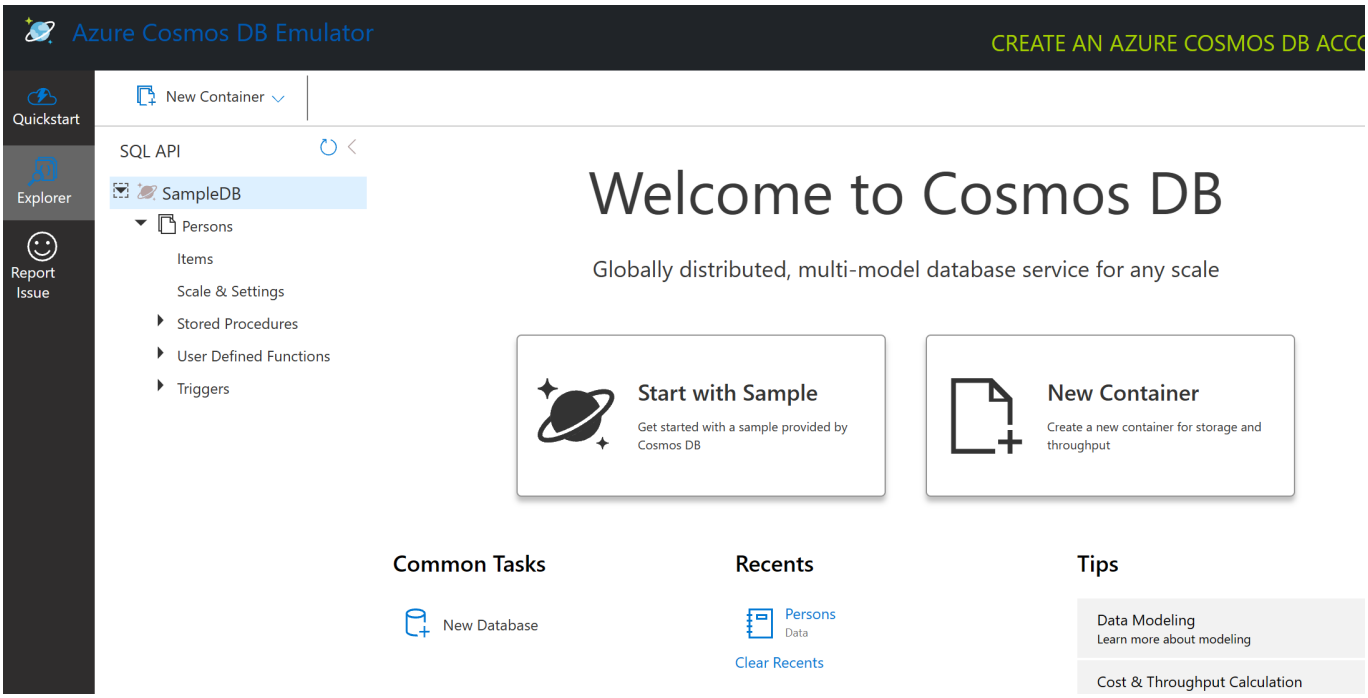
```
{
    "firstname": "Eva",
    "age": 44,
    "id": "8c74a315-aebf-4a16-bb38-2430a9896ce5",
    "_rid": "bHwDAPQz8s0BAAAAAAAAAA==",
    "_self": "dbs/bHwDAA==/colls/bHwDAPQz8s0=/docs/bHwDAPQz8s0BAAAAAAAAAA==/",
    "_etag": "\"00000000-0000-0000-9f95-010a691e01d7\"",
    "_attachments": "attachments/",
    "_ts": 1630544034
}
```

The fields of interest in this document are: `firstname`, `id`, and `age`. The rest of the fields with the underscores were generated by Cosmos DB.

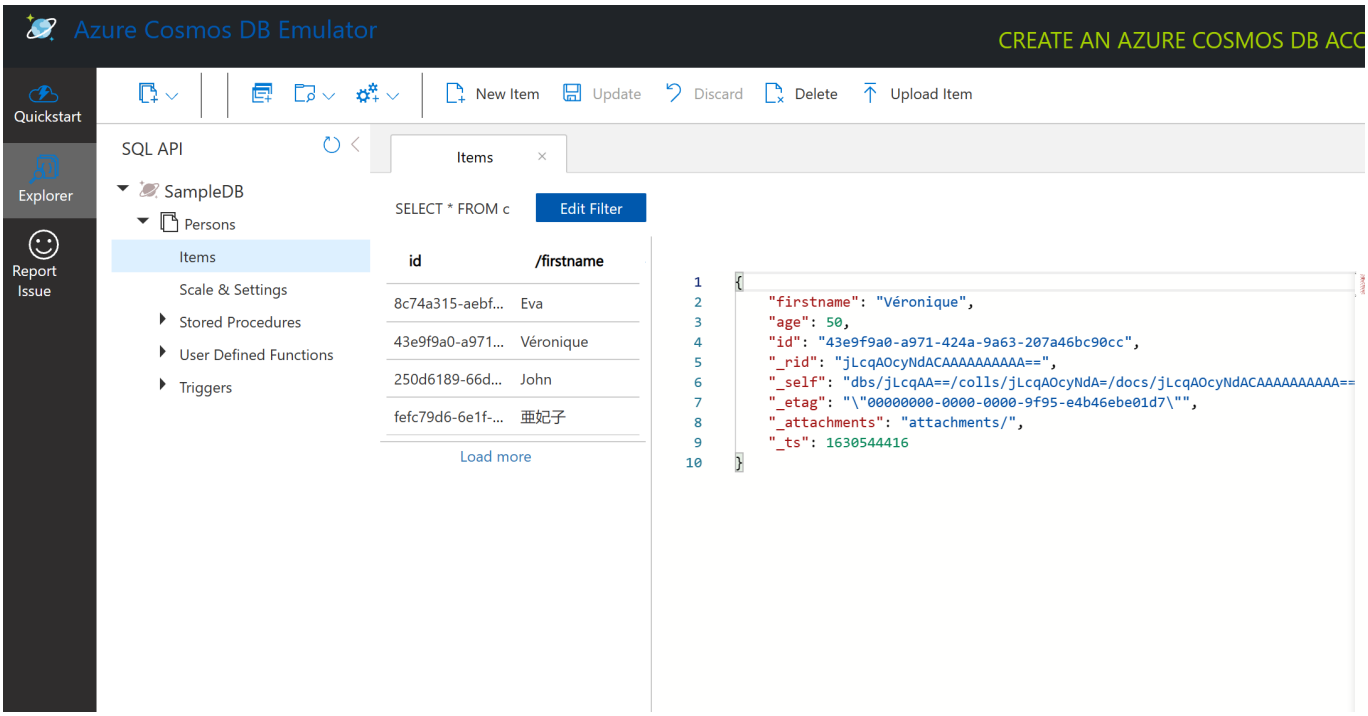
Exploring Data with the Cosmos DB Emulator

You can download and install the emulator for Windows here. Refer to this documentation for options on how to run the Emulator for macOS and Linux.

The Emulator launches a browser window, where the Explorer view allows you to explore documents.



If you're following along, click on "Start with Sample" to generate a sample database called SampleDB. If you expand Sample DB by clicking on the arrow you'll find a container called `Persons`, a container holds a collection of items, which are the documents within the container. You can explore the four individual documents under `Items`.

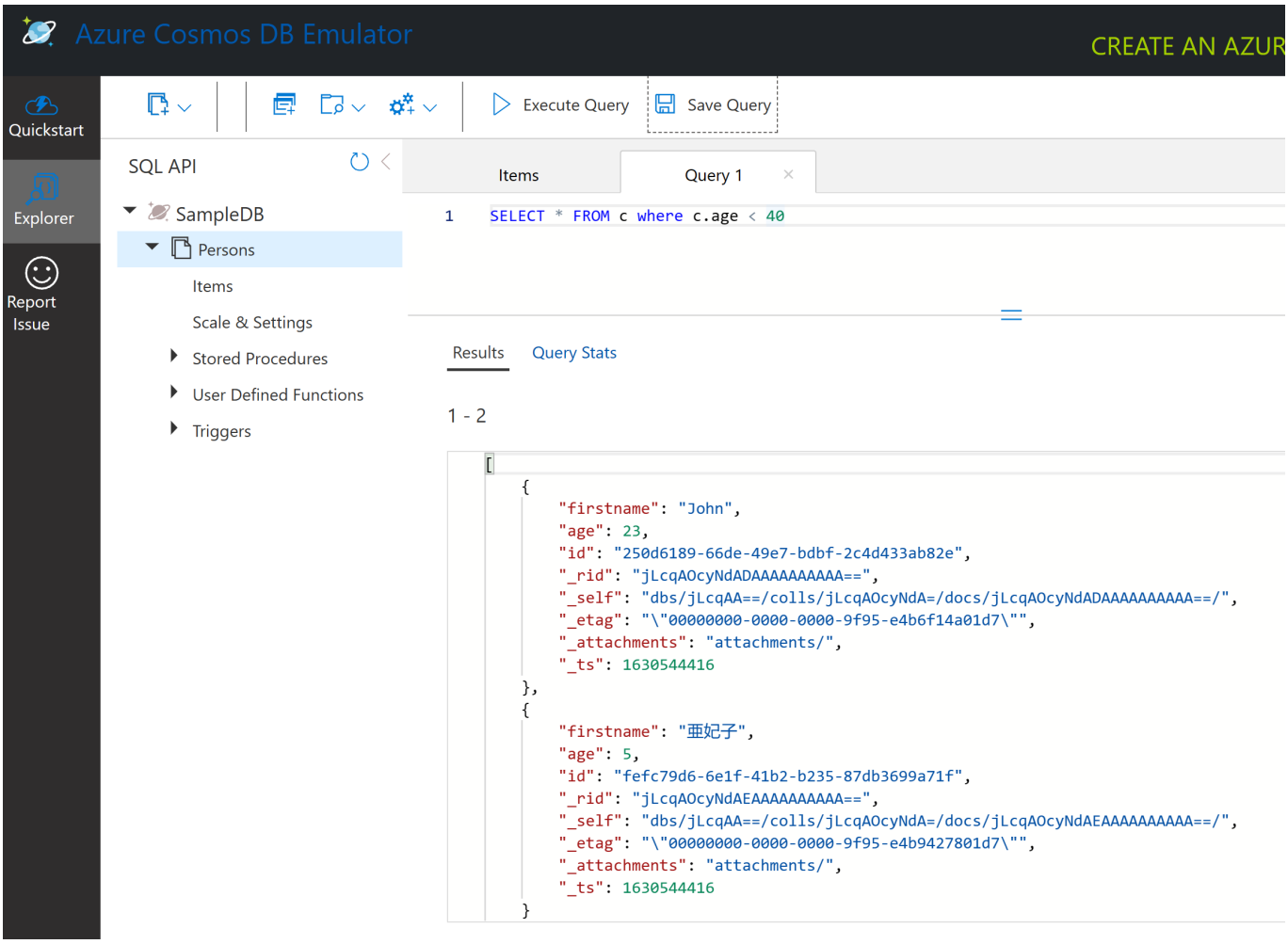


Querying Document Data with the Cosmos DB Emulator

We can also query the sample data by clicking on the new SQL Query button (second button from the left).

`SELECT * FROM c` returns all the documents in the container. Let's add a where clause and find everyone younger than 40.

```
SELECT * FROM c where c.age < 40
```



The query returns two documents, notice the age value for each document is less than 40.

JSON and Documents

If you're familiar with JavaScript Object Notation (JSON) you'll notice that documents look similar to JSON. There is a `PersonsData.json` file in this directory with more data that you may upload to the Persons container in the Emulator via the `Upload Item` button.

In most instances, APIs that return JSON data can be directly transferred and stored in document databases. Below is another document, it represents tweets from the Microsoft Twitter account that was retrieved using the Twitter API, then inserted into Cosmos DB.

```
{
    "created_at": "2021-08-31T19:03:01.000Z",
    "id": "1432780985872142341",
    "text": "Blank slate. Like this tweet if you've ever painted in Microsoft
Paint before. https://t.co/cFeEs8eOPK",
    "_rid": "dhAmAIUsA4oHAAAAAAAAAA==",
    "_self": "dbs/dhAmAA==/colls/dhAmAIUsA4o=/docs/dhAmAIUsA4oHAAAAAAAAAA==/",
    "_etag": "\"00000000-0000-0000-9f84-a0958ad901d7\"",
    "_attachments": "attachments/",
    "_ts": 1630537000
```

The fields of interest in this document are: `created_at`, `id`, and `text`.

🚀 Challenge

There is a `TwitterData.json` file that you can upload to the SampleDB database. It's recommended that you add it to a separate container. This can be done by:

1. Clicking the new container button in the top right
2. Selecting the existing database (SampleDB) creating a container id for the container
3. Setting the partition key to `/id`
4. Clicking OK (you can ignore rest of the information in this view as this is a small dataset running locally on your machine)
5. Open your new container and upload the Twitter Data file with `Upload Item` button

Try to run a few select queries to find the documents that have Microsoft in the text field. Hint: try to use the LIKE keyword

Post-Lecture Quiz

Review & Self Study

- There are some additional formatting and features added to this spreadsheet that this lesson does not cover. Microsoft has a large library of documentation and videos on Excel if you're interested in learning more.
- This architectural documentation details the characteristics in the different types of non-relational data: Non-relational Data and NoSQL
- Cosmos DB is a cloud based non-relational database that can also store the different NoSQL types mentioned in this lesson. Learn more about these types in this Cosmos DB Microsoft Learn Module

Assignment

Soda Profits