ETL Project

Presented by Cintia, Berenice and Carlos.

Project Introduction

For this project, we use  2 sources of data, both were found on Kaggle:

<https://www.kaggle.com/ihelon/lego-minifigures-classification?select=index.csv>

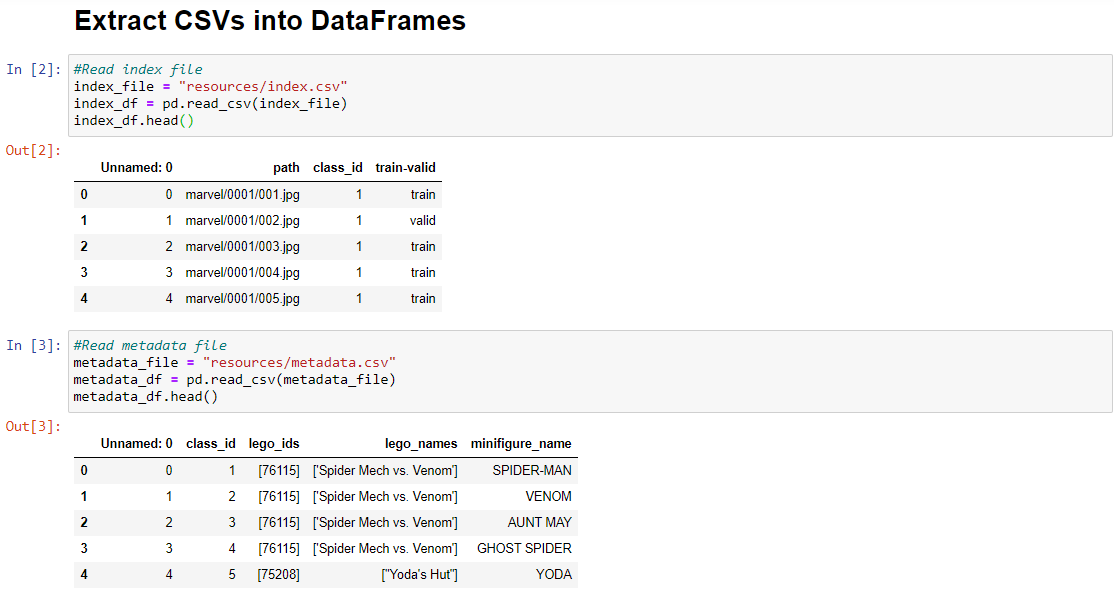
E: Extract sources

Description: The data source “metadata.csv” is a dataset pulled for lego detail, including: id, name and minifigure name from every product and “index.csv” is a dataset for collection path images for every lego figure.

Files format: csv

Columns used in metadata.csv: class\_id, lego\_id, lego\_name, minifigure\_name

Columns used in index.csv: image\_id, image\_path, class\_id



T: Transform

Data was edited and cleaned

Drop index column that was a sequential numbers for every row in the tables

Lambda function (apply) was implemented  to clean special characters like brackets, double quotes in columns: lego\_ids and lego\_names.

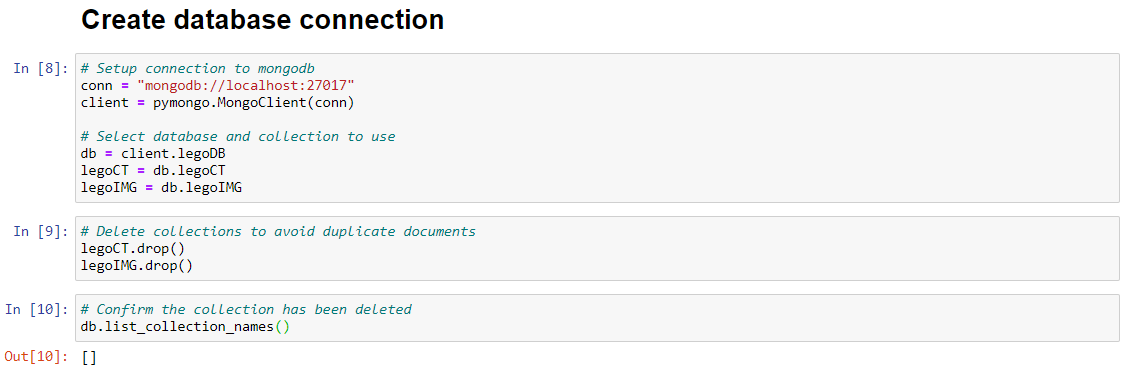
Pandas library was used to format, remove and edit columns.

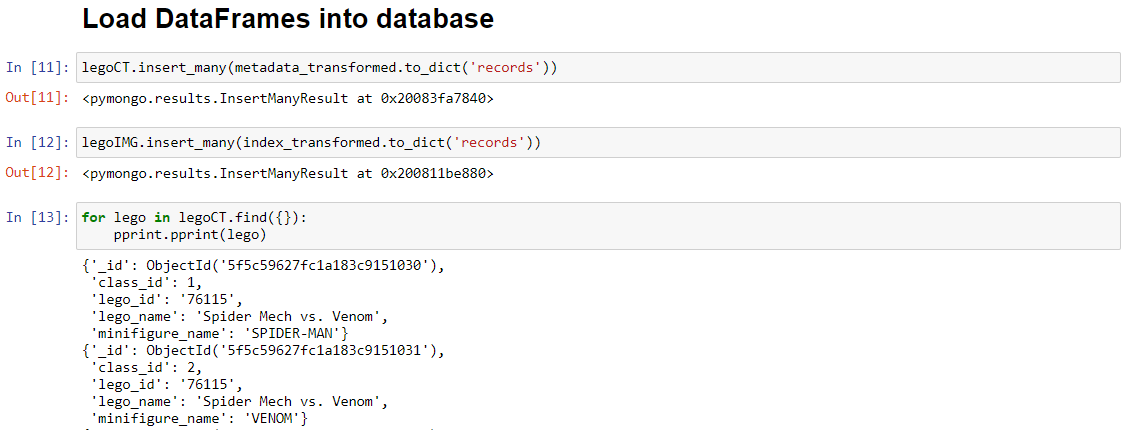
Each csv file was written into mongo collections.

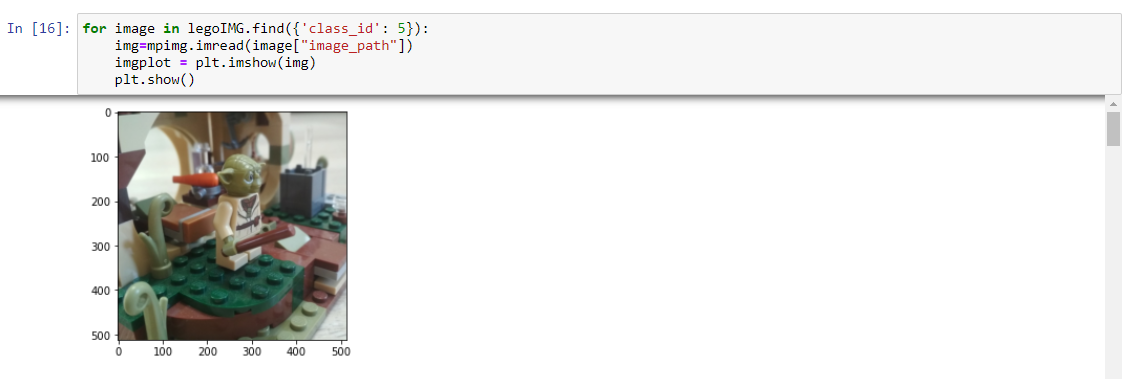


L: Load

* Due to the technical challenge that it represents for the members of the team, it was decided to use mongoDB.
* Jupyter Notebook was used to create the database connection to mongoDB and transfer data to the non-relational database, aside from creating the collections.
* In order to avoid duplicate documents in the database, the collections are dropped each time the process is executed.
* The collections are populated in one go from the dataframes, using the method insert\_many.
* To confirm data load, the find method is used to retrieve all documents for both collections.







The final structure of the database is as follows:

**Database name:** **legoDB**

**Collections:**

* **legoCT.**

**Fields:** \_ID, CLASS\_ID, LEGO\_ID, LEGO\_NAME, MINIFIGURE\_NAME

* **legoIMG**

**Fields:** \_ID, CLASS\_ID, IMAGE\_ID, IMAGE\_PATH