**2020**



**Big Data Architecture & Governance**

**Student Names:**

Bhagyashree Nair

Parth Jani

Smruti Jethwani

**Assignment Name:**

Group 7- Book Depository

1. Contents

[2. Assignment 2](#_Toc121520786)

[2.1. Assignment Goals 2](#_Toc121520787)

[3. Documentation 3](#_Toc121520788)

[3.1. Vision Diagram 3](#_Toc121520789)

[3.2. Data Profiling 4](#_Toc121520790)

[3.3. Data Wrangling and Cleansing 6](#_Toc121520791)

[3.4. Database Installation 9](#_Toc121520792)

[3.5. Data Mapping and Integration 13](#_Toc121520793)

[3.6. Technical and Business Metadata 15](#_Toc121520794)

[3.7. Data Validation and Data Visualization 16](#_Toc121520795)

[3.8. Visualization Interpretations 21](#_Toc121520796)

[3.9. System Integration and User Acceptance Testing 27](#_Toc121520797)

[3.10. Risks and Issues 30](#_Toc121520798)

[3.11. Challenges Encountered 32](#_Toc121520799)

[3.12. Velero Portal 33](#_Toc121520800)

[3.13. End User Instructions 35](#_Toc121520801)

# Assignment

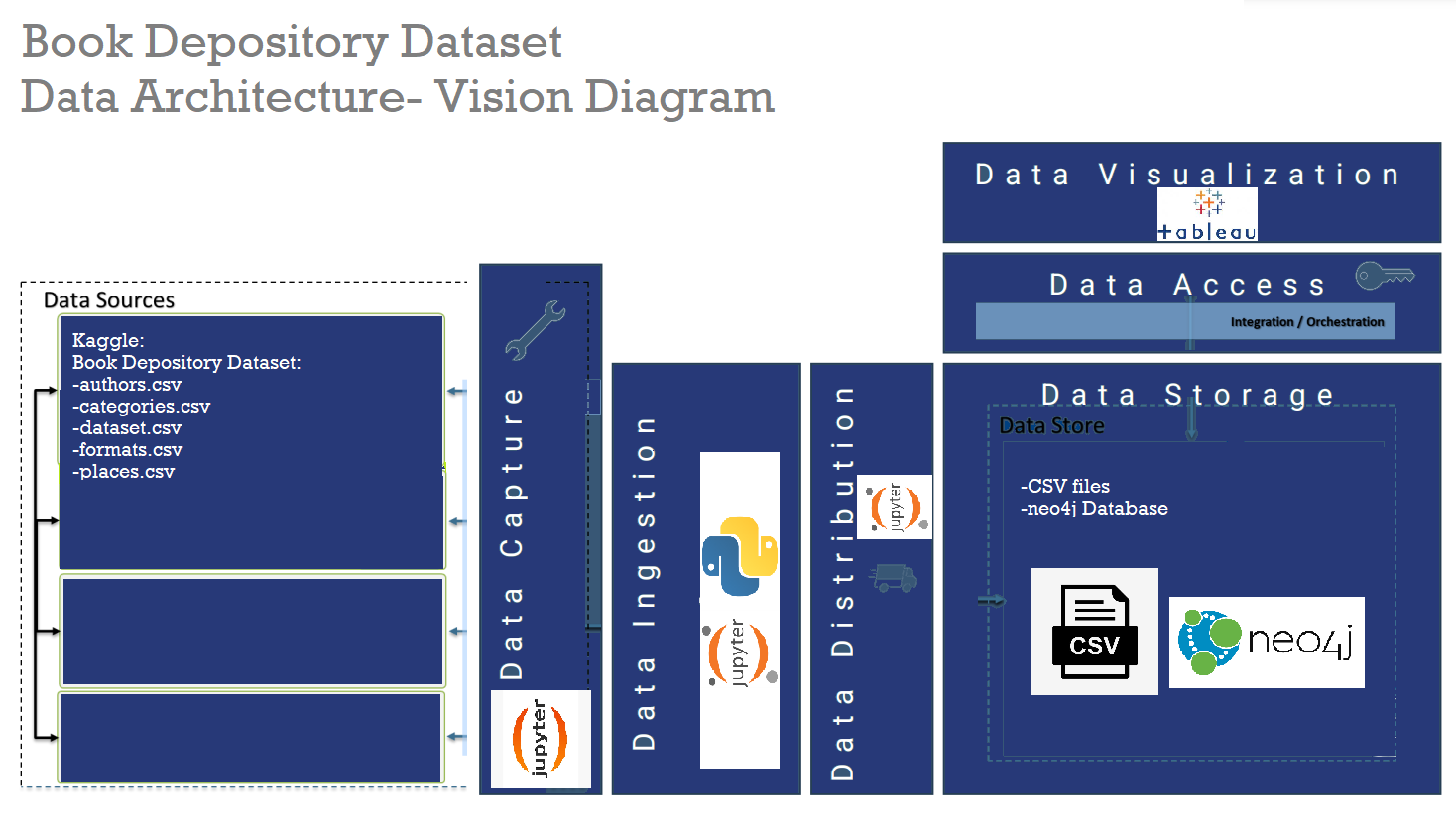
## Assignment Goals

To work with datasets, Perform/Create:

* Create you group assignment project in Velero:
  + Project
  + Project Plan
  + Resource Allocation
  + Timesheet
  + Issues & Risks.
  + You are required to report on your team progress every week
* **Data Profiling** – Using Python profiling library, describe your understanding of the data.
* **Data Wrangling and Cleansing** - Pandas/Alteryx/XSV
  + Filtering and Aggregating if needed.
  + Missing value handling.
  + Deriving additional columns from existing datasets if needed.
  + Cleaning (removing blank spaces, formatting dates, Capitalizing etc.).
* Database Installation: Install NEO4J database.
* Data Mapping and Integration to your Database for the Entire Dataset.
* **Business and Technical Metadata** – develop a business term list describing all the data elements available in the file.
* **Data Validation** – Validate the data using python data libraries.
* **Data Visualization** – Create a presentation dashboard to reflect your understanding of the data, you may use python visualization libraries or Power BI
* **System Integration and User Acceptance Testing** - Test Cases – describe your validation & testing process.
* **Risks/Issues** – identify risks and issues related to your project.
* Describe challenges encountered and how you resolved them.
* **End User Instructions (Steps to run your Dashboard)** – provide a full description how to run your process:
  + Database Creation and load.
  + Visualization interpretation - describe information regarding your findings.

# Documentation

## Vision Diagram



**Data architecture of our project:**

**Data Sources:**

Kaggle link- <https://www.kaggle.com/datasets/sp1thas/book-depository-dataset?select=dataset.csv>

**Dataset files-**

dataset.csv, authors.csv, categories.csv, formats.csv, places.csv

**Data Capture** performed on Jupyter notebook

**Data Ingestion, Integration** **and Distribution** performed on Jupyter Notebook

**Data stored** in csv files then imported to a neo4j database

**Data Access** done by JDBC drivers to connect to Tableau

**Data Visualization** performed on Tableau

## Data Profiling

The dataset consists of 5 files-

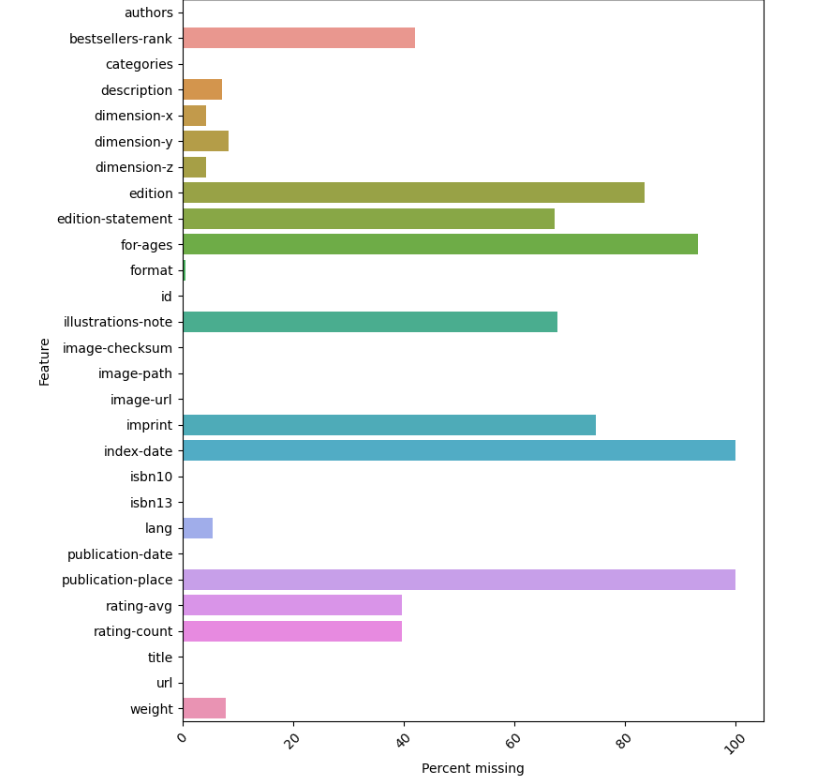
main dataset file

4 lookup files- author, categories, format, places.

Each of them is linked to the main dataset by a unique key.

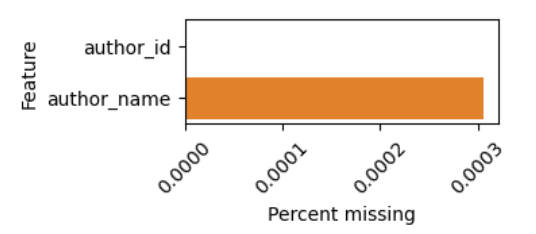
The dataset dimensions are as follows: 1109383 rows and 28 columns

Most of the main dataset features have null values. A visual depiction stating the columns and their null percentage is as follows-

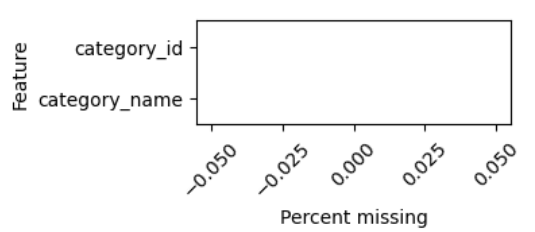


Our lookup tables don’t have as many values as stated here-

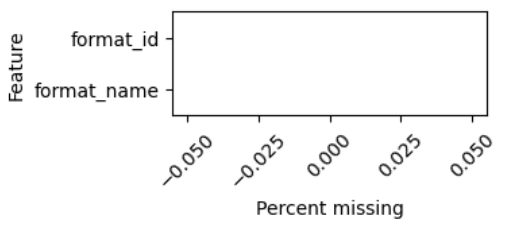
Authors.csv:



Categories.csv



Format.csv



Few points observed during data profiling:

1. Categories and authors are mentioned as a list for a particular book. Lists will not help us analyze it clearly- thus affecting our visualizations. We have to ensure to explode these lists to treat each value as its own, and give it an individual record
2. ‘url’ feature doesn’t contain the actual link to the book depository website. It has to be formatted to create a proper URL which can be later used to check mapping of the data.
3. A lot of null value handling will have to be performed based on the attribute’s datatype. However, 2 attributes- index\_date and publication\_place have no data records and will therefore contribute nothing towards analysis and visualization

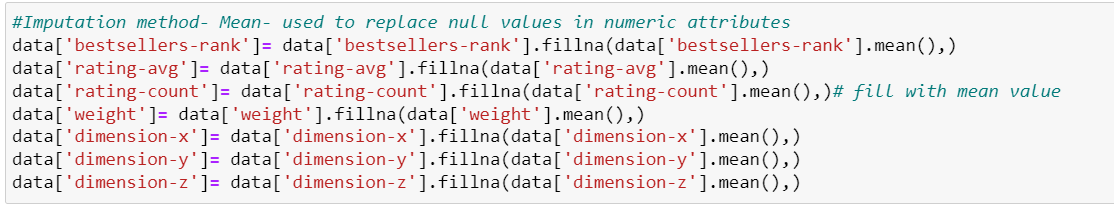
## Data Wrangling and Cleansing

We performed the following Data Cleansing activities:

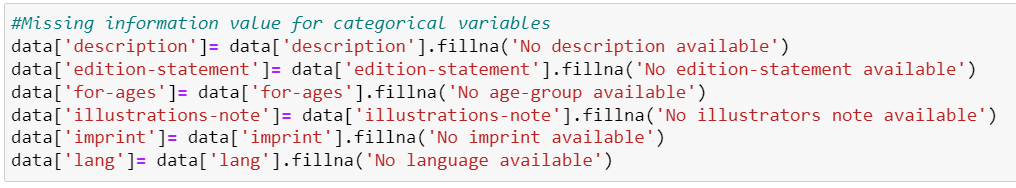
**Missing null value handling:**

Our dataset consisted of a lot of null values. To handle them, we first differentiated between the numeric and categorical data types.

For numeric data types, we used the mean imputation method for each column to replace the null values.



For categorical data types, we provided a Customized 'missing information'



We had 2 attributes with all 1109383 records being NULL-

Index-date, publication-place

Since we will not be able to do any analysis, we dropped these 2 attributes



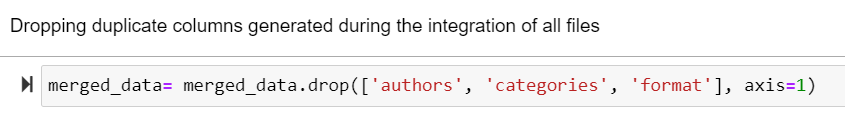
**Cleaning:**

After null-value handling, we moved onto cleaning our data. This was an iterative process, even after loading our final csv file into neo4j as the software was throwing some errors. Activities:

1. Trimmed unwanted blank spaces
2. Removing duplicate column names:

We had 3 csv files to be merged with our main file. While we performed the merging, our unique\_id that was used for the merge was getting duplicated to reflect both the merged files. Since they were the exact same columns, we dropped the duplicate ones.

We merged with 3 other files, hence 3 columns were dropped.

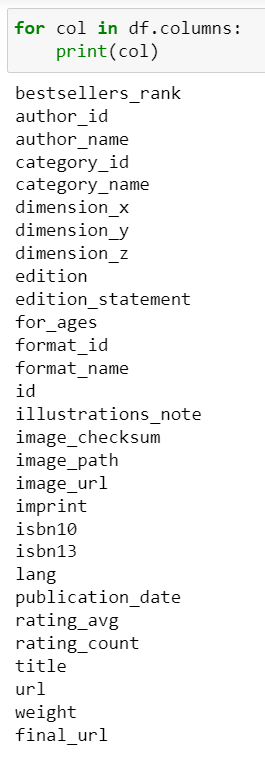


1. Double quotes were used in our ‘description’ attribute in a particular format which was throwing errors when loading to neo4j. This was replaced with single quoted to make it cleaner and solve neo4j loading problems



1. Most of our attribute names had a ‘hyphen’ in them- neo4j failed to recognize and read these names as is- In order to solve this, we cleaned our attribute names and renamed them with an underscore in place of hyphen.

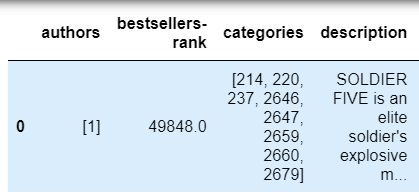


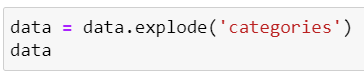
New attribute names: 

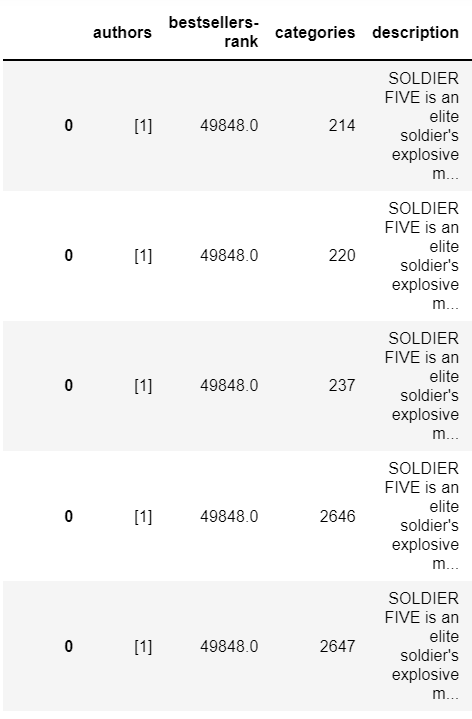
**Filtering and Aggregating:**

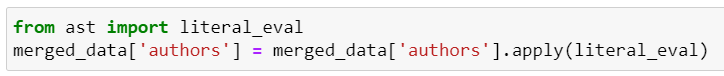
We transform each element of a category list and author list to a row, replicating the index values. This is done in order to incorporate records for every category a book belongs to and every author that published the book. This was implemented by using the data.explode() function.

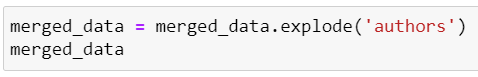








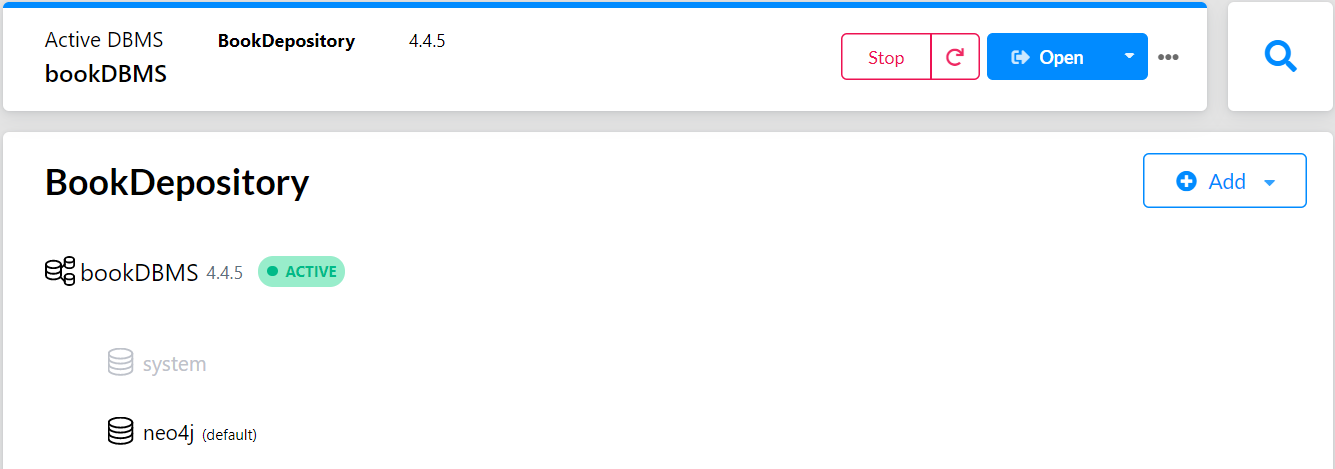




## Database Installation

Neo4j Project: BookDepository

Database: bookDBMS



Cypher statements

//Create constraints for all nodes

CREATE CONSTRAINT ON (book:Book) ASSERT book.book\_id IS UNIQUE;

CREATE CONSTRAINT ON (l:Language) ASSERT l.InLanguage IS UNIQUE;

CREATE CONSTRAINT ON (a:Author) ASSERT a.author\_id IS UNIQUE;

CREATE CONSTRAINT ON (f:Format) ASSERT f.format\_id IS UNIQUE;

CREATE CONSTRAINT ON (c:Category) ASSERT c.category\_id IS UNIQUE;

CREATE CONSTRAINT ON (age: AgeGroup) ASSERT age.age\_group IS UNIQUE;

CREATE CONSTRAINT ON (b:BestSellers) ASSERT b.bestsellers\_rank IS UNIQUE;

CREATE CONSTRAINT ON (p:Publication) ASSERT p.publication\_date IS UNIQUE;

CREATE CONSTRAINT ON (i: Images) ASSERT i.image\_path IS UNIQUE;

//Create Book Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE(book:Book{book\_id:row.id})

ON CREATE SET

book.title=row.title, book.edition=row.edition, book.edition\_statement=row.edition\_statement, book.illustrations\_note =row. illustrations\_note, book.index\_date= row.index\_date, book.imprint= row.imprint, book.isbn10=row.isbn10, book.isbn13=row.isbn13, book.rating\_avg= row.rating\_avg, book.rating\_count= row.rating\_count, book.x\_dimension= row.dimension\_x, book.y\_dimension= row.dimension\_y, book.z\_dimension= row.dimension\_z, book.weight\_kg= row.weight, book.url= row.final\_url;

//Create Language Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE(l:Language{InLanguage:row.lang});

// Load Author

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE

(a:Author{author\_id: toInteger(row.author\_id)})

ON CREATE SET

a.author\_name=row.author\_name;

//Create Format Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE

(f:Format{format\_id: row.format\_id})

ON CREATE SET

f.format\_type=row.format\_name;

//Create Category Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE

(c:Category {category\_id: row.category\_id})

ON CREATE SET

c.category\_name=row.category\_name;

//Create BestSellers Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE(b:BestSellers { bestsellers\_rank :row. bestsellers\_rank });

//Create Images Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE

(i: Images { image\_path :row. image\_path })

ON CREATE SET

i.image\_path=row.image\_path, i.image\_url= row.image\_url;

//Create AgeGroup Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE(age: AgeGroup { age\_group :row.for\_ages });

//Create Publication Node

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MERGE(p:Publication { publication\_date :row.publication\_date  });

//Relationships

//Book-lang relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv'  AS row    MATCH (book:Book {book\_id:row.id})

MATCH (l:Language{InLanguage: row.lang})

MERGE (book)-[:available\_in]->(l);

//Book-author relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv'  AS row

MATCH (book:Book { book\_id: row.id})

MATCH (a: Author{author\_name: row.author\_name})

MERGE (book)-[:written\_by]->(a);

//Book-format relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MATCH (book:Book { book\_id: row.id})

MATCH (f: Format{format\_type: row.format\_name})

MERGE (book)-[:available\_in]->(f);

//Book-category relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MATCH (book:Book { book\_id: row.id})

MATCH (c:Category { category\_name : row.category\_name})

MERGE (book)-[:belongs\_to]->(c);

//Book-publication relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MATCH (book:Book { book\_id: row.id})

MATCH (p:Publication {publication\_date: row. publication\_date })

MERGE (book)-[:published\_on]->(p);

//Book-age\_group relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MATCH (book:Book { book\_id: row.id})

MATCH (age:AgeGroup {age\_group: row. for\_ages })

MERGE (book)-[:read\_by]->(age);

//Book-Images relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MATCH (book:Book { book\_id: row.id})

MATCH (i:Images {image\_path: row.image\_path })

MERGE (book)-[rel: has\_an]->(i);

//Book-BestSellers relationship

:auto USING PERIODIC COMMIT 500

LOAD CSV With HEADERS FROM 'file:///file\_100000.csv' AS row

MATCH (book:Book { book\_id: row.id})

MATCH (b:BestSellers { bestsellers\_rank: row.bestsellers\_rank  })

MERGE (book)-[rel: has\_a]->(b);

//Visualizing the graph model

call db.schema.visualization

## Data Mapping and Integration

We have 5 csv files

1 Main\_data file that has all the book depository information.

4 lookup files that need to be merged with our main\_data to get more information.

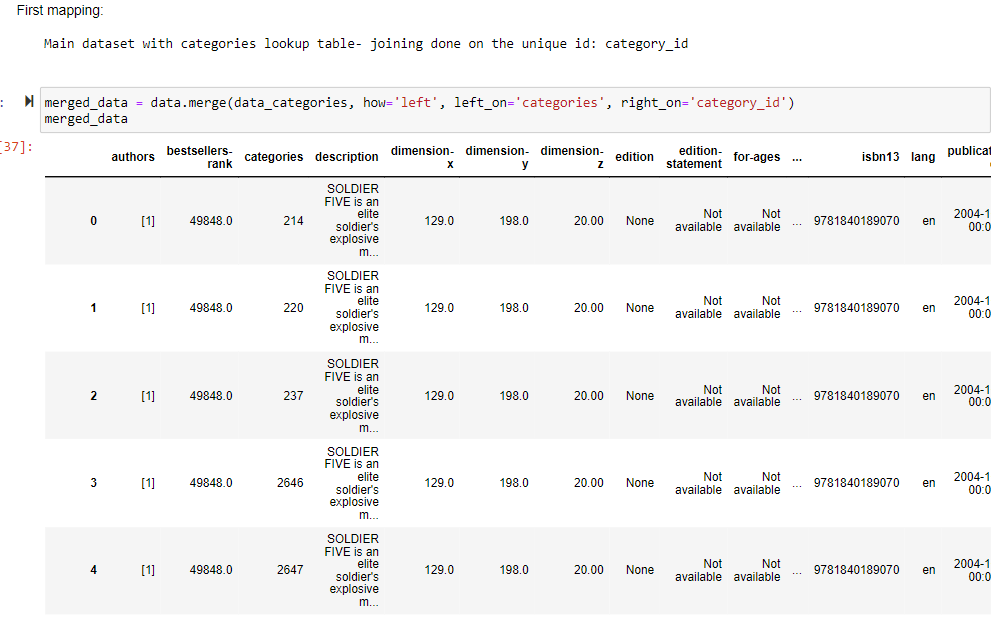
1 lookup file:places.csv was supposed to have information on all the places where the books are published. However, the file is completely empty and cannot be used for any analysis. Since there was absolutely no data, we couldn’t perform any kind of data augmentation to get a slightly better picture of the publication data.

The 3 remaining csv files are as follows:

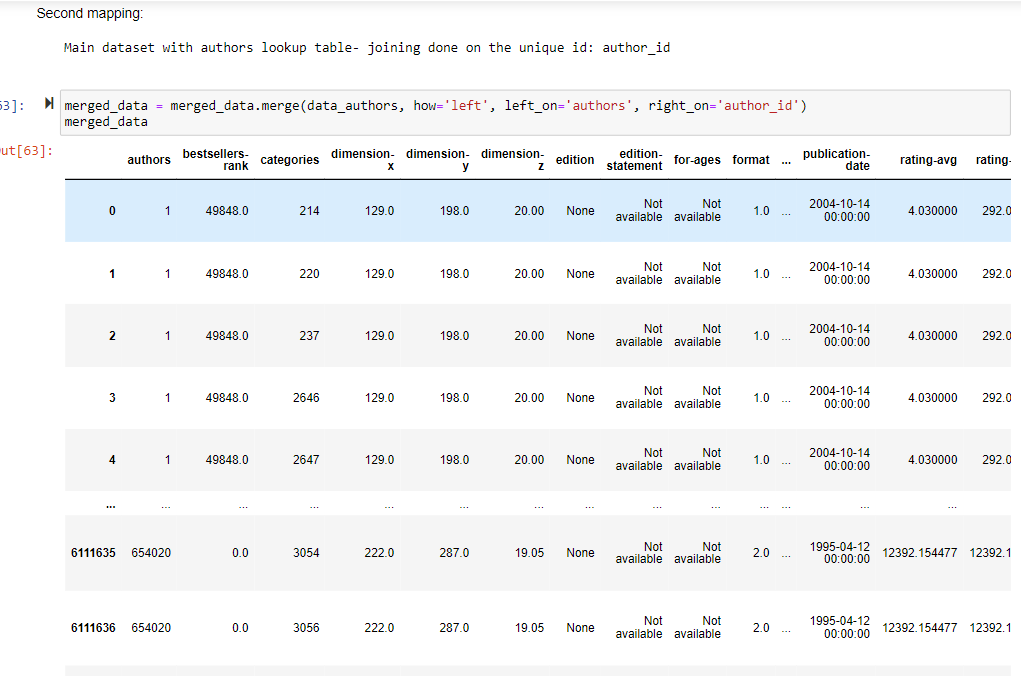
|  |  |  |
| --- | --- | --- |
| **File name** | **Primary key (main\_data)** | **Primary key(lookup\_file)** |
| Authors.csv | authors (unique id for every author) | author\_id |
| Categories.csv | Categories (unique id for every category) | category\_id |
| Formats.csv | Format (unique id for every format) | format\_id |

We performed mapping and integration of the above in Jupyter notebook

**Categories mapping and integration:**



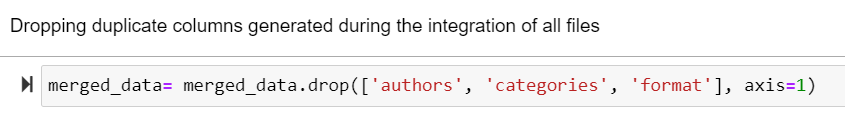
**Author mapping and integration**



**Format mapping and integration:**



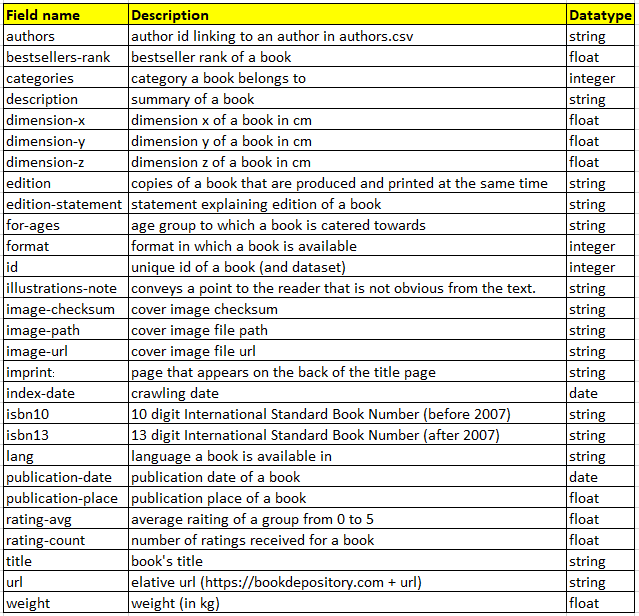
The merging of the lookup tables with the dataset resulted in duplicate columns of authors, categories, format and description which were then dropped.



The dataset after exploding the category and author fields, mapping and integration consists of 6111640 rows and 29 columns.

## Technical and Business Metadata

Business Metadata:

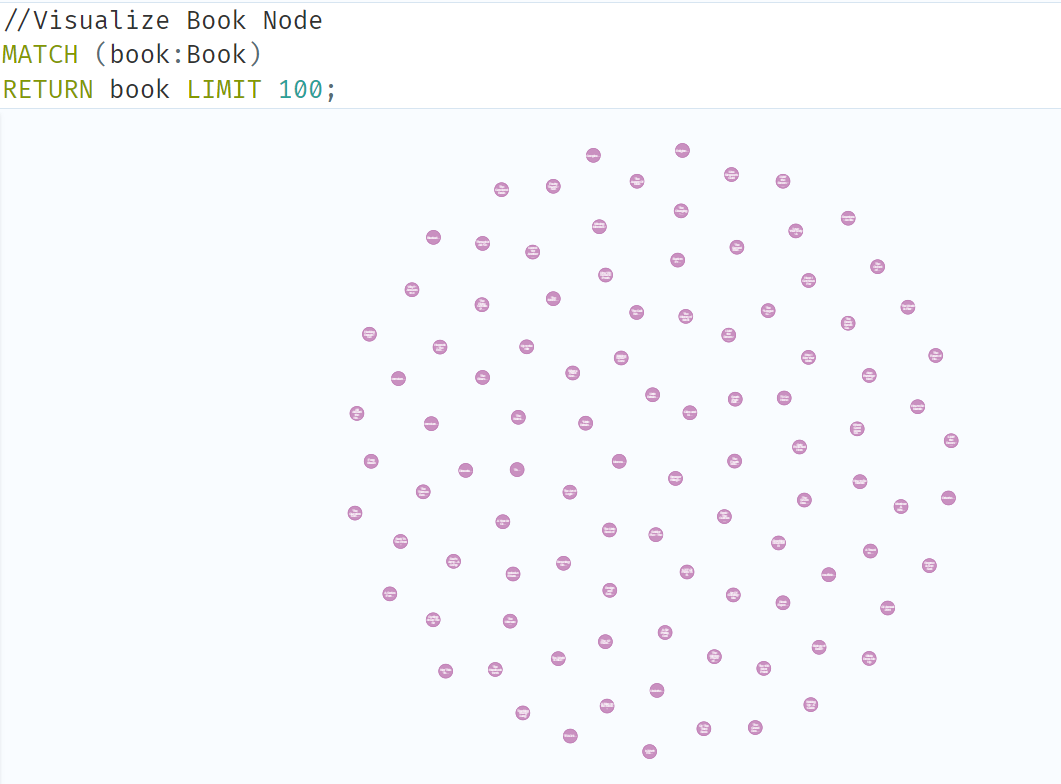
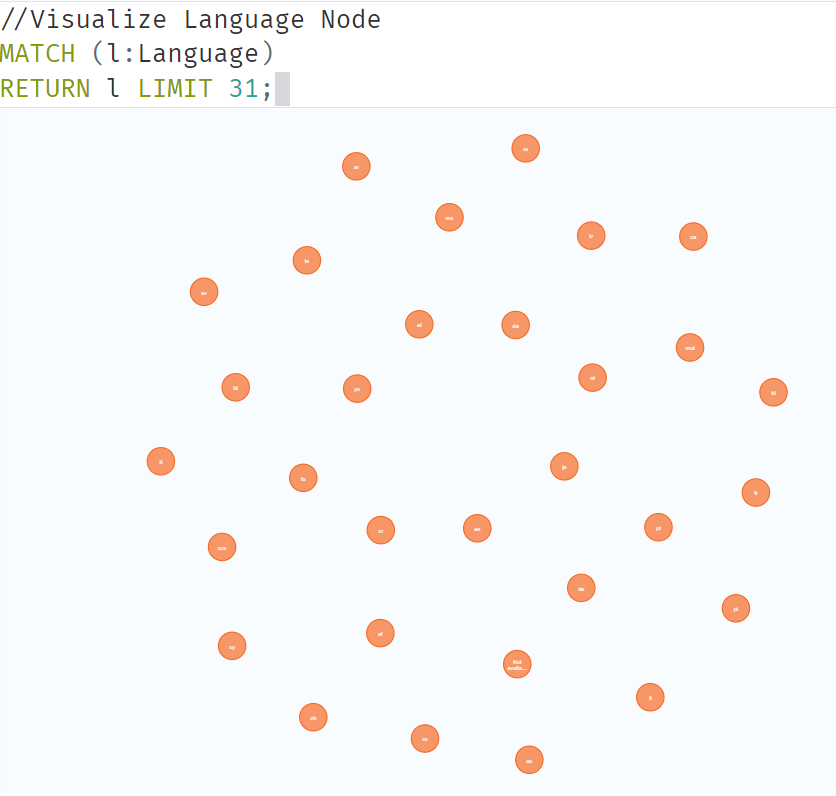


## Data Validation and Data Visualization

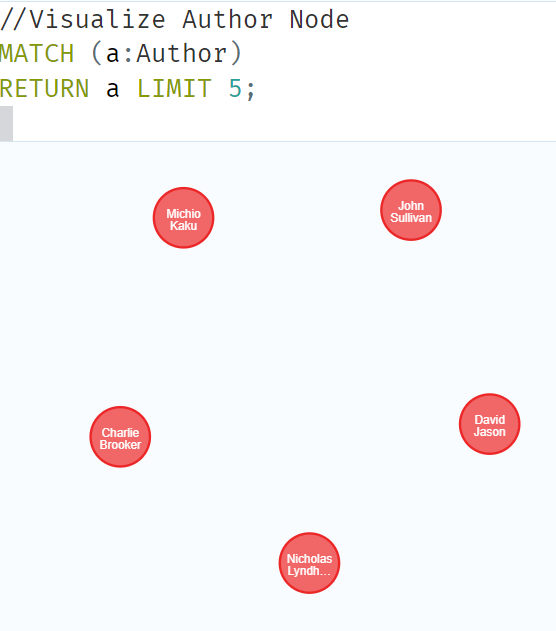
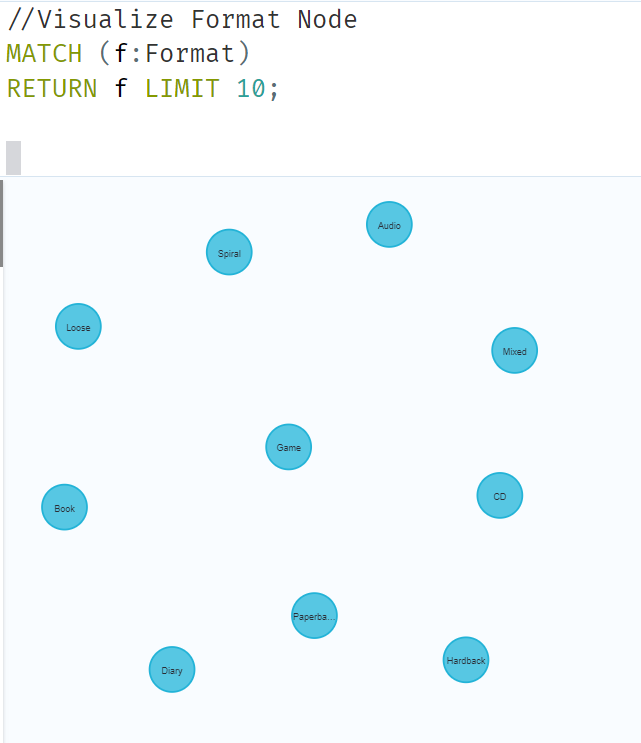
**Graph model visualization:**

**Nodes:**

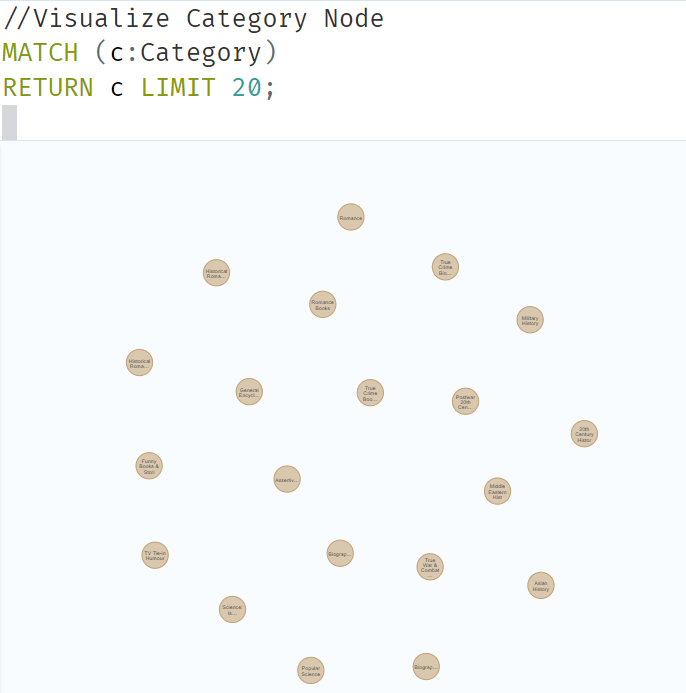
Book Language

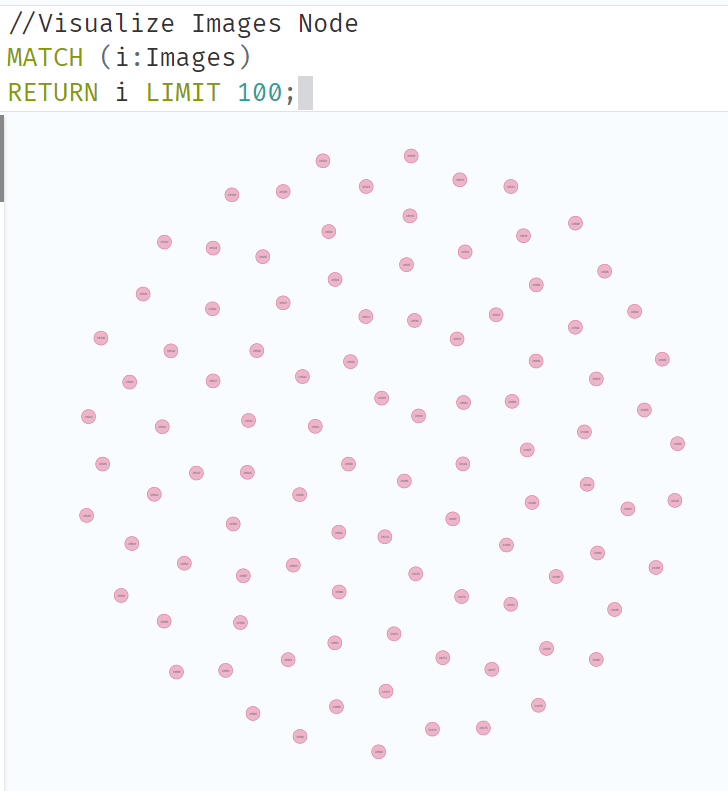
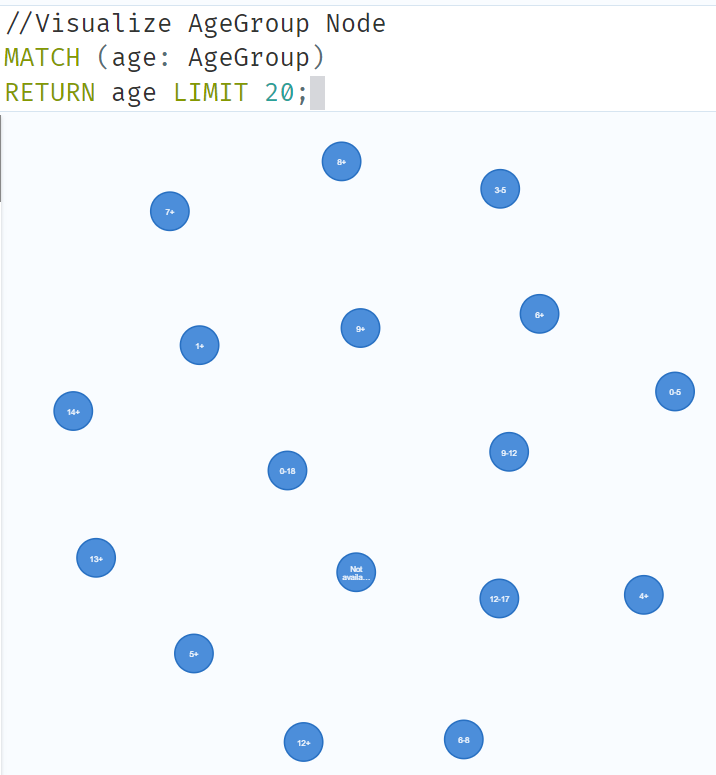
Author Format

Category BestSellers

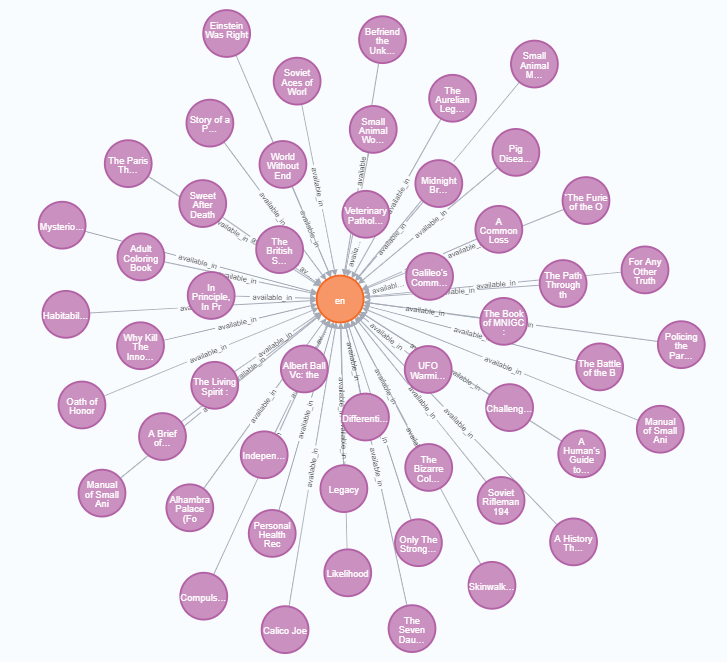
 

Images AgeGroup

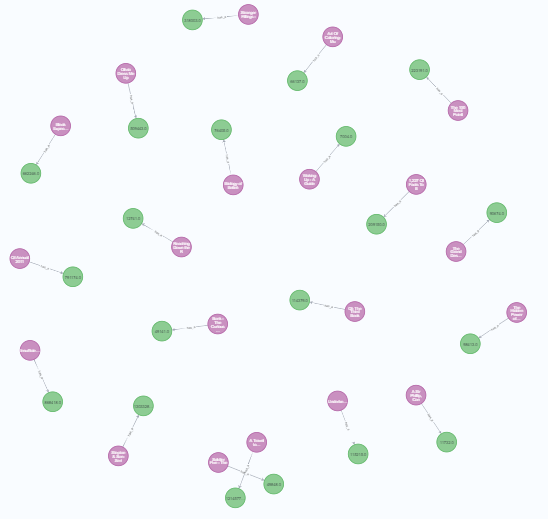
 

**Relationships:**

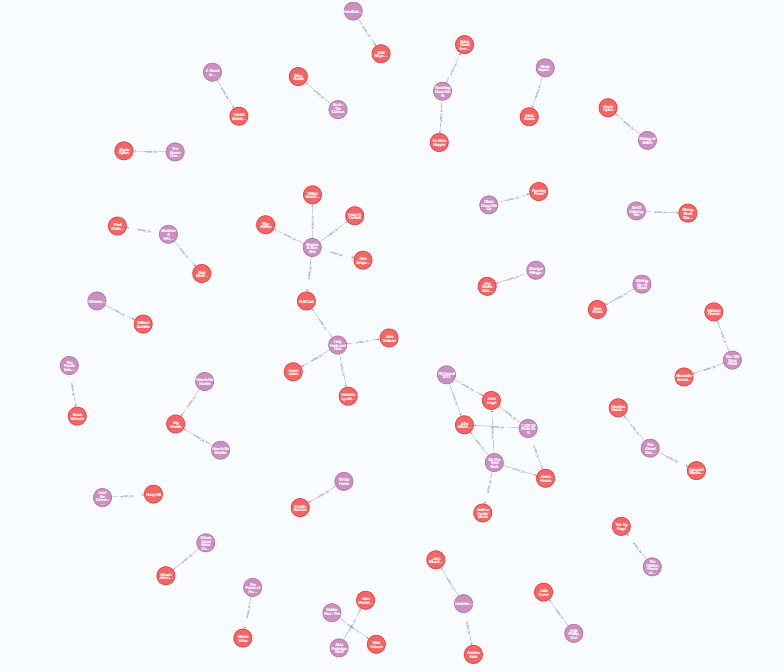
Book language relationship



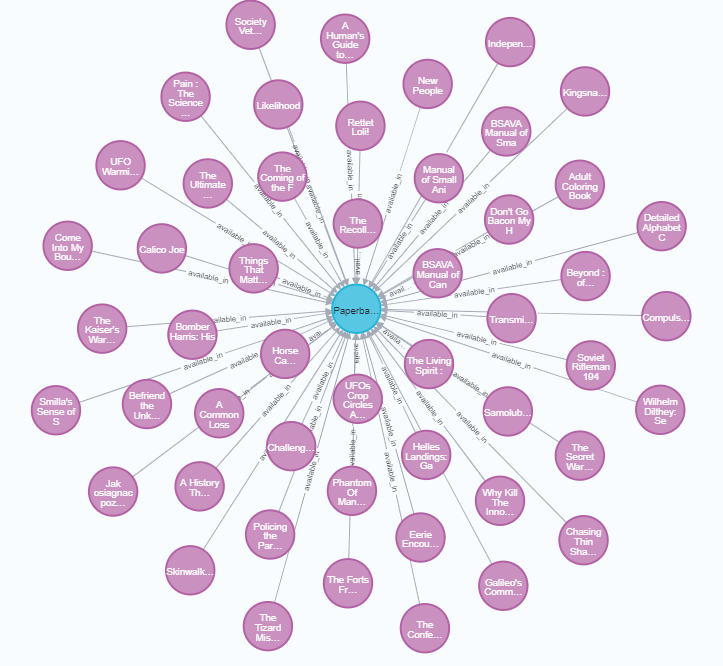
Book Best Seller Relationship



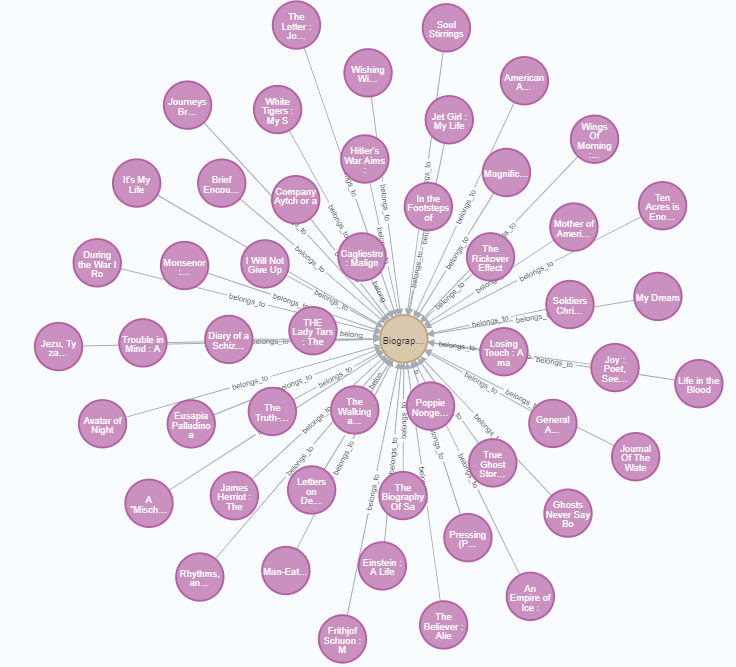
Book Author Relationship



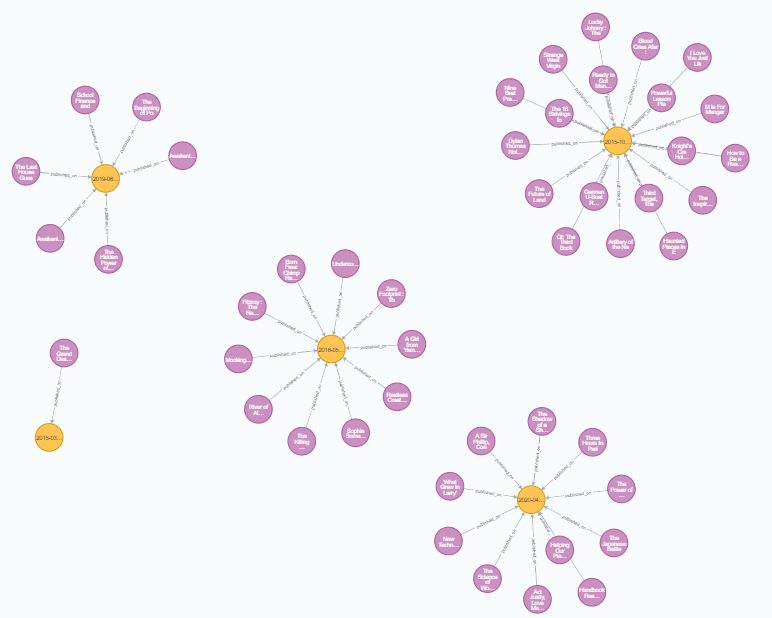
Book Format Relationship



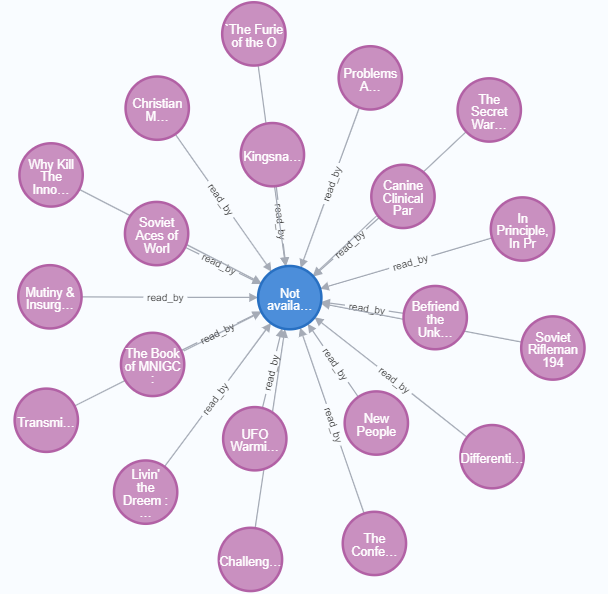
Book Category Relationship



Book Publication relationship



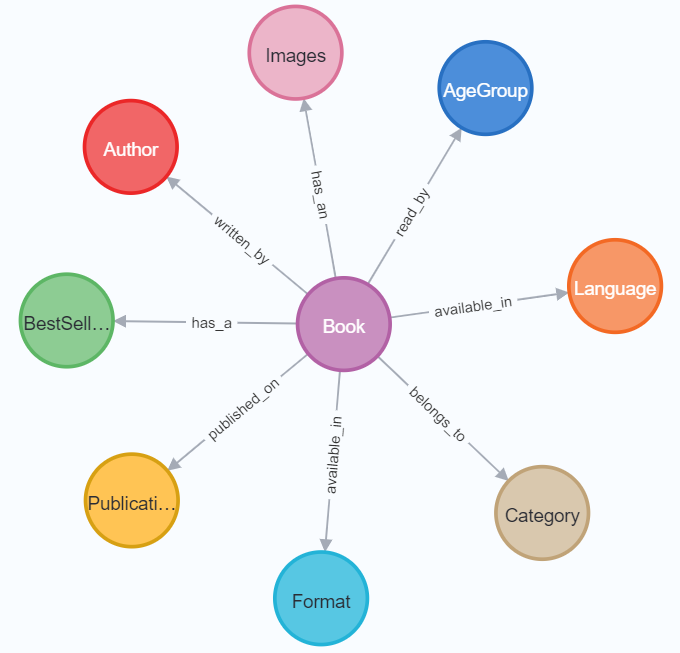
Book Age group relationship



Book Images relationship

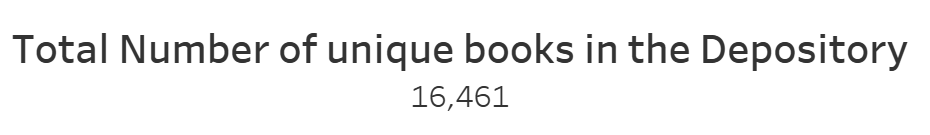


Final graph model:



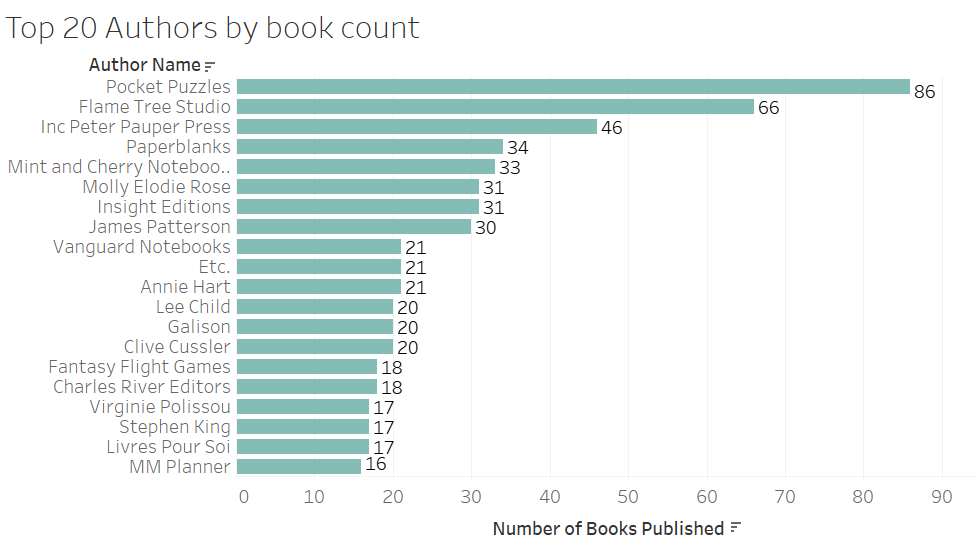
## Visualization Interpretations

Here are our Tableau visualizations and interpretations:



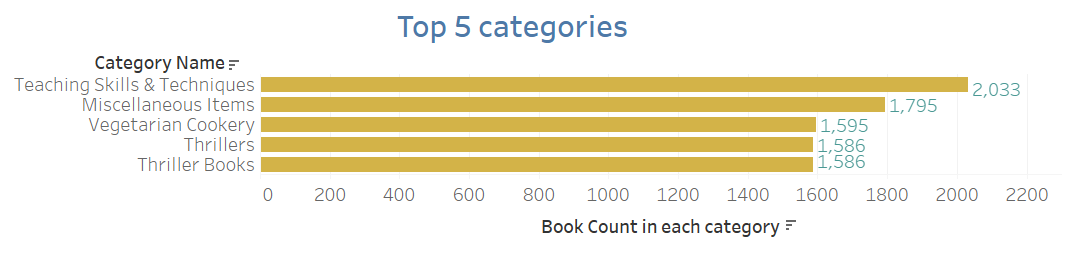
Interpretation:

This is the total number of unique books that we have in the depository.



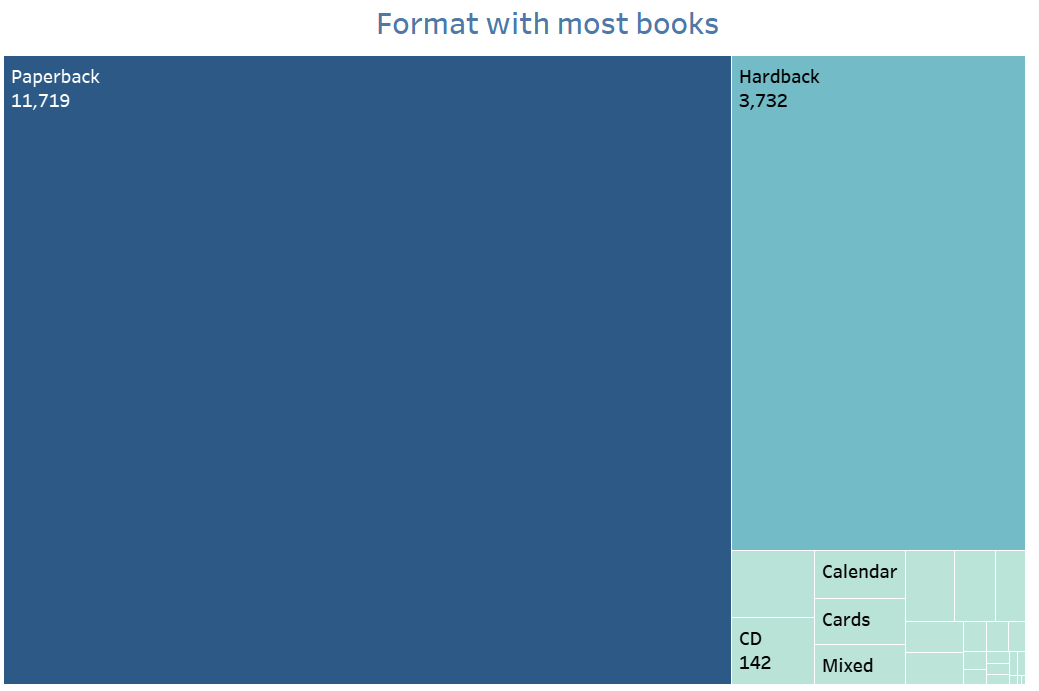
Interpretation:

The above visualization shows Top 20 Authors who have the most number of books published in the book depository. One observation we made is that most author names are essentially publication names.

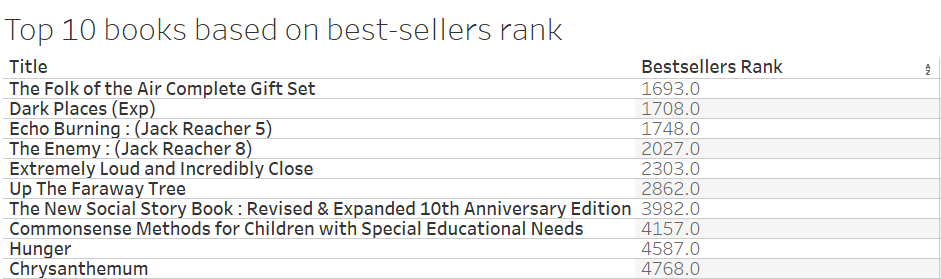


Interpretation:

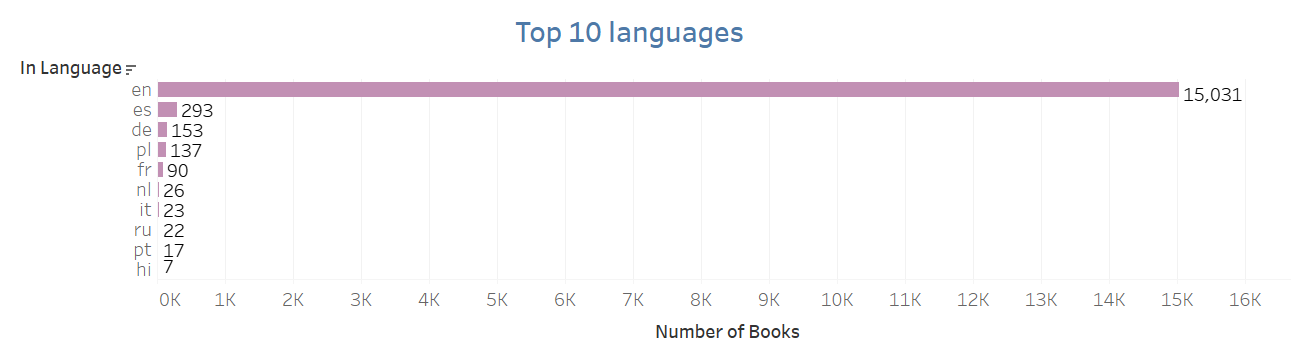
This visualization is about the Top 5 categories of books that were published in the depository. We observed that there is a bigger gap in number of books published between the Top 2 categories relative to any other categories.



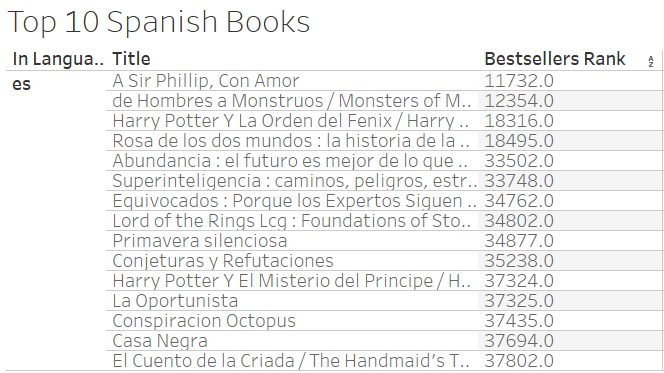
Interpretation: This treemap is about the most number of books based on formats. We clearly observe that paperback is the most preferred format in books followed by hardback.



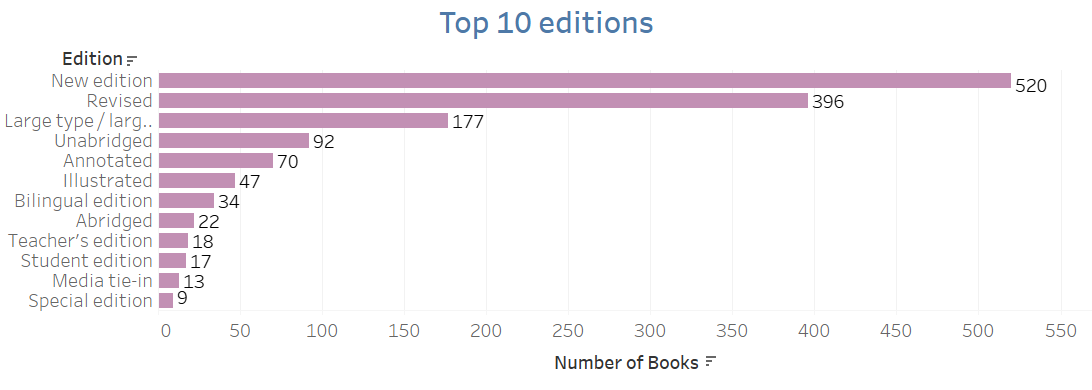
Interpretation: The above visualization is about the Top 10 books published based on Bestsellers Rank. We observed that there are no books with bestsellers ranked between 1-1000.



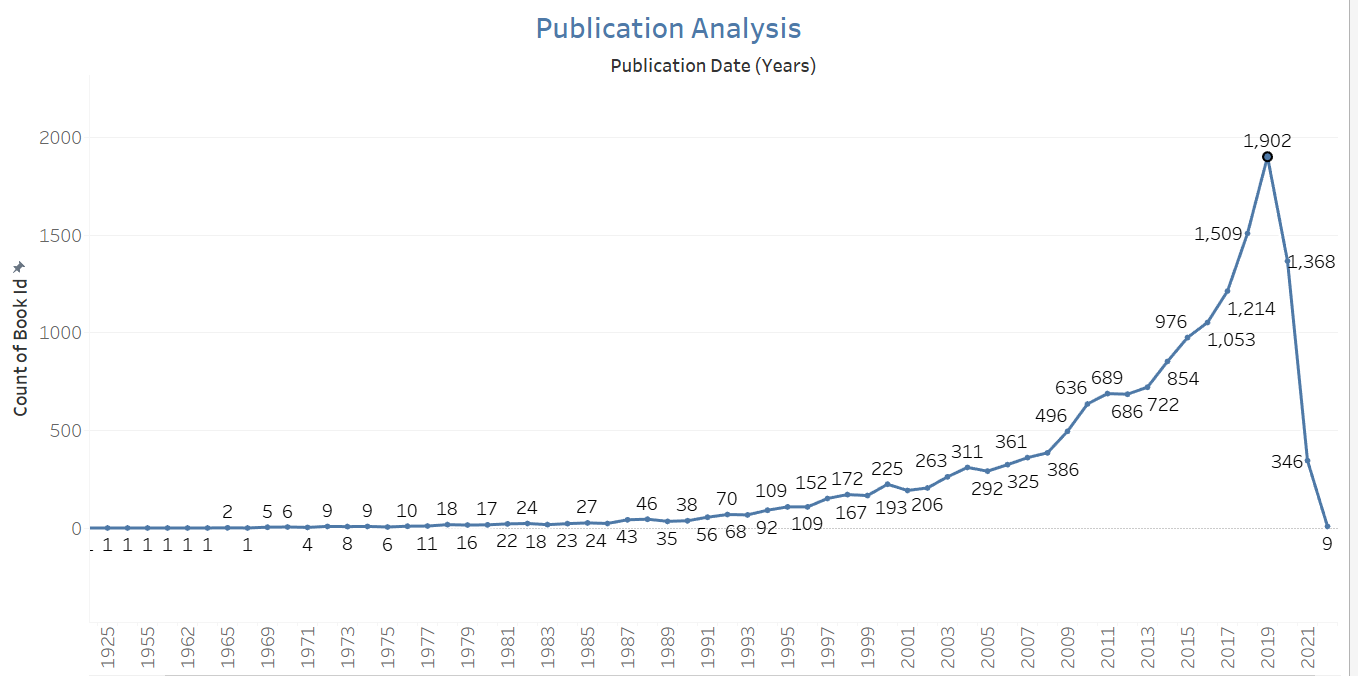
Interpretation: The above visualization is about Top 10 languages in which the books were published. We observed that English is clearly the most known language the books are published in.



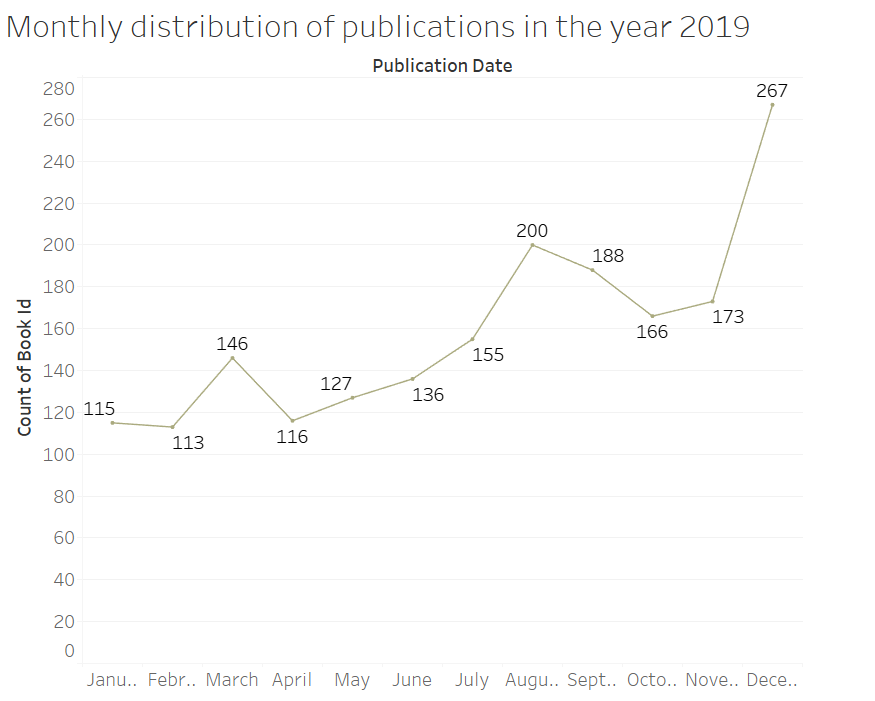
Interpretation: We wanted to observe books published in Spanish because of its reach.



Interpretation: The above visualization shows Top 10 editions of books published.



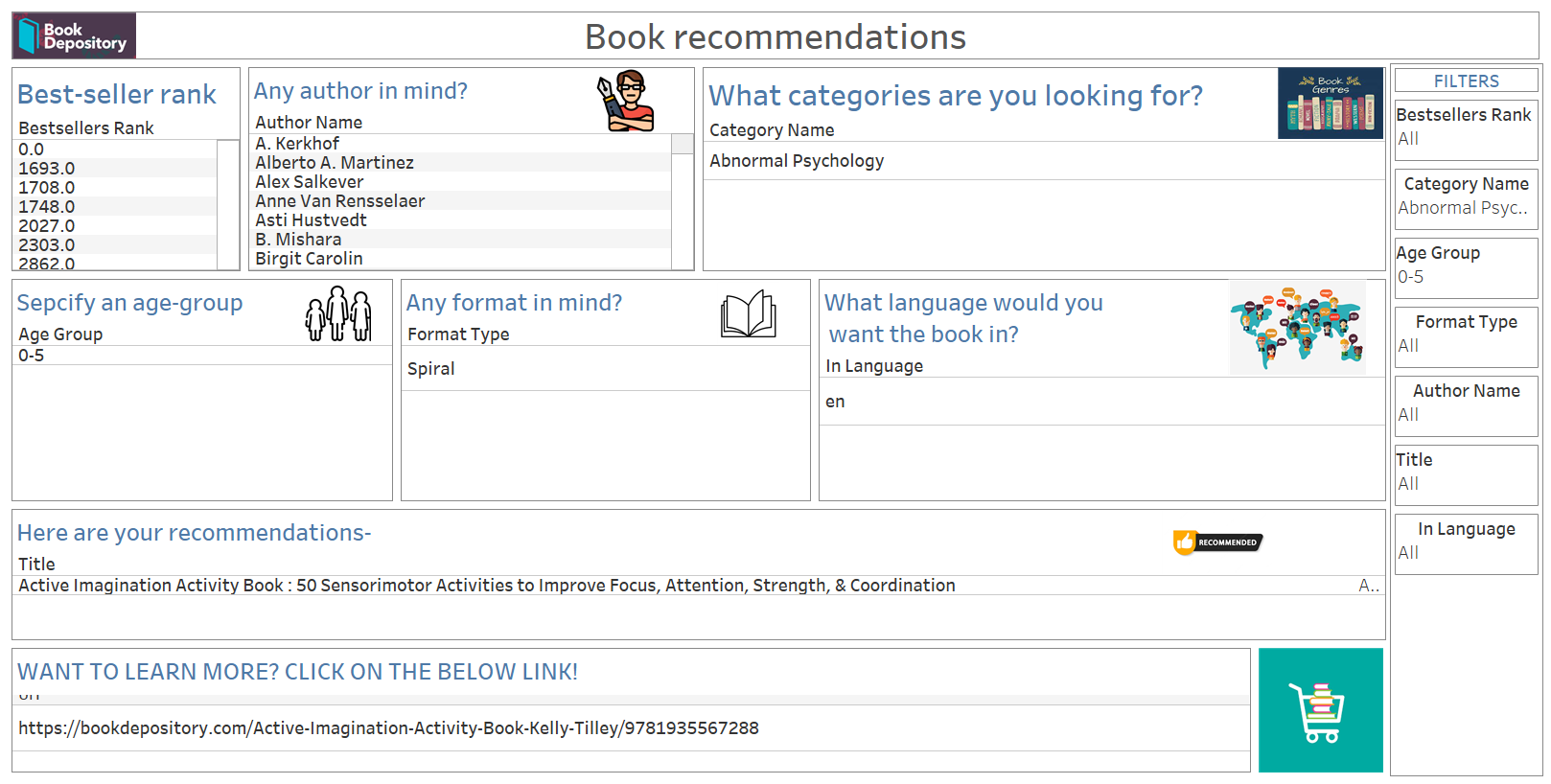
Interpretation: This visualization shows the books published over the years. We observe that 2019 had the highest number of publications i.e. 1902



Interpretation: From the previous visualization, we wanted to observe the number of books published monthly in the year 2019 as it was the year in which most books were published.

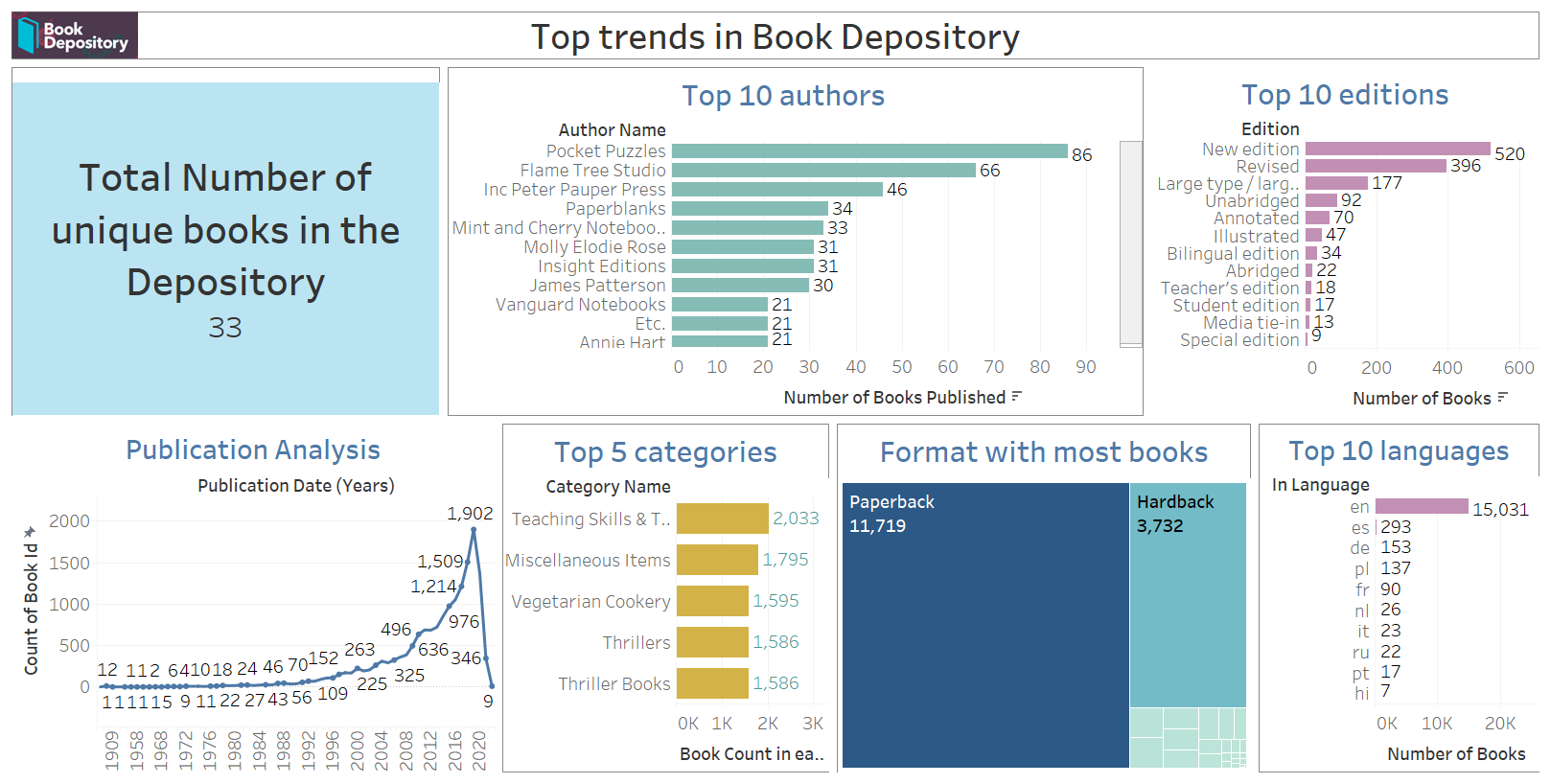
**DASHBOARD 1: Book Recommendations**

We have created a Book Recommendation Dashboard where the user can give their reading preferences and get book recommendations. It will then take you to the book’s online depository link from where you can learn more about the book, and even make a purchase.



**Dashboard 2: Top trends**

This dashboard is a visualization of the top trends in our Book Depository Dataset.



## System Integration and User Acceptance Testing

**Test 1: Attribute null value check**

Number of null values was accessed before and after data wrangling. Null values were not removed but were altered for use in data analysis.

2 attributes with all 1109383 records being NULL: Index-date, publication-place were dropped as they were of no use.

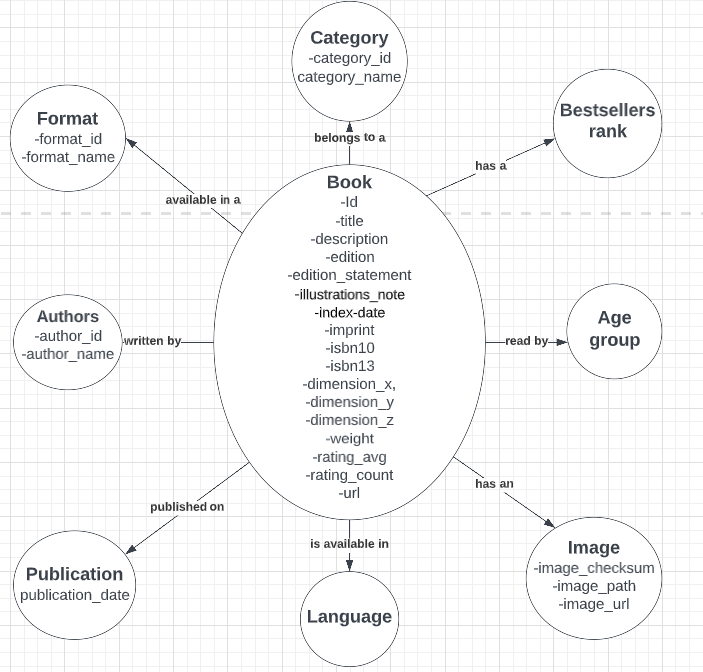
Please note: Right side analysis doesn’t have those 2 attributes

|  |  |
| --- | --- |
| BEFORE NULL HANDLING | AFTER NULL HANDLING |
|  |  |

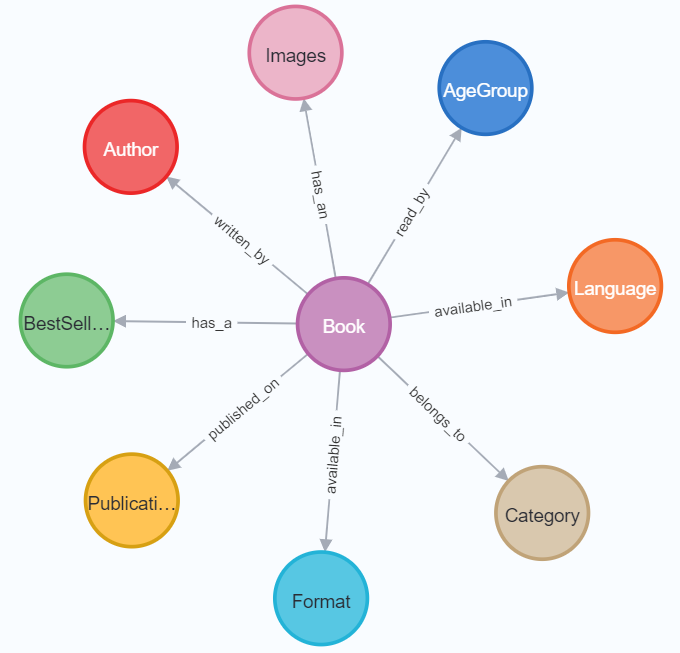
**Test 2: Model design check**

We started off with a model design before uploading our csv file to neo4j and creating the nodes and relationships. This test is to ensure that we implement a graph model based on the design we had in mind while cleaning and integrating our data. This will also test if the properties of the node match with the attributes in our initial model design

Design before creating a graph model:



Graph model



**Test 3: Record count match in source and target**

After data cleaning, mapping and integration on Jupyter notebook, we loaded the final data on a csv file. This was then loaded to a neo4j database.

This test is to ensure that our Jupyter notebook created csv is properly loaded to the neo4j databases

|  |  |
| --- | --- |
| SOURCE (dataset.csv) | TARGET (neo4j) |
|  |  |

**Test 4: Unique record count match in source and target**

Once we check the total record count, we test a unique count of each attribute with respect to the node created on neo4j.

This will test the mapping conducted on Jupyter notebook, and should ideally give an equal record count between source and target.

|  |  |
| --- | --- |
| SOURCE (dataset.csv) | TARGET (graph model on neo4j) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Risks and Issues

Following are the risks and issues faced during our project. Mitigation techniques for each have been specified as well.

|  |  |
| --- | --- |
| Issue/Risks | Mitigation Techniques |
| **Issue: Many Missing Values in dataset**  Fields needed to track publication details, editions, book ratings had a lot of missing data which might be an analytical risk leading to biased visualization | Missing attributes were replaced with mean for numerical attributes and a Customized 'missing information' message for categorical attributes  Publication date was used for a time series analysis of the books instead of focusing on the completely NULL publication place |
| **Issue: Neo4j processing time = very high for our dataset with 6 million records**  Neo4j was not able to process our very huge dataset. It would take close to 4 hours to process creation of a node or relationship, and the software would crash thereafter. | After several trials, we decided to process a smaller batch of our dataset to test the software capability. After trying 500k, 250k and 100k sample size, we concluded that neo4j was able to process 100k records smoothly and give proper node creation and relationships. |
| **Issue: Lack of Knowledge in terms of a book depository dataset.**  There were attributes such as index date, ISBN10, ISBN13 that we were not aware of, and didn’t understand its effect on analysis and unbiased visualization. No extra information found on Kaggle either | Research of literary terminology to understand the attributes of our dataset. Starting from basic difference between ISN10 and ISN13 to learning what an imprint of a book means, this has led us to understanding the data better and given better visualization ideas |
| **Risk: Data loss**  Data loss can be the result of accidental deletion, power outages and spikes, and it can also be due to a deterioration of the media.  Interrupting the Jupyter Kernel or not scheduling runs on time could lead to variable value losses resulting to data loss | Scheduled runs to ensure that Jupyter notebook does not lose any data  Use virus-detection protection programs and keep the system updated. |
| **Risk: License Agreements**  Publishers often require book depositories to sign agreements that enforce use restrictions, such as on copying and downloading, which are more stringent than those that copyright law would impose.  Each country has a different policy for acquiring / accessing the different types of content and agreements would be a risk. | Detailed discussion with government/ publishers/ vendors for collection management covering all aspects of the future needs/ demands/ changes. |
| **Risk: Use of Technology**  For accessing the online book depository, the proper use of the website, by having sufficient connectivity and systems is one of the risks. People may have different types of systems, and if there is a change in software and formats for access, downloading from the website to that system is a risk. | Ensuring that the book depository website is user friendly and provides customer assistance if required.  Having robust antivirus software on your computer is a must |
| **Issue: Staff Training and Recruitment**  The staff who are already working are to be retrained to update their knowledge and may have to recruit new entrants with better / higher qualifications. The risk involved here is the willingness to attend the training and risk of taking new staff, by thinking that they will be better than the existing ones. | Hiring staff with competitive skills and also develop skills for the existing staff, by providing in-house training |

## Challenges Encountered

* We were provided with 5 Datasets (authors.csv,categories.csv,dataset.csv,formats.csv,places.csv). Out of which, places.csv was completely empty
* During data profiling, we found out that there were several columns in our main dataset(i.e.dataset.csv) which were entirely null making it redundant to data analysis
* The language column had 163 distinct fields consisting of different language codes. We figured around 30 languages based on our needs according to the ISO 639 nomenclature.( ISO 639 is a standardized nomenclature used to classify languages)
* Our analysis on publications was narrowed down due to the absence of data in the lookup file- places.csv
* Categories and authors were given in the form of a list for each book record. Transformation had to be performed on this to make it usable for deeper analysis.
* After successfully cleaning and integrating our data, we had 6111640 records and 29 columns. Loading this on Neo4j was a tedious task. neo4j was not able to process this size. Our laptop RAM was not good enough to process this as well. The query would run for 3 hours ,create close to 100k relationships (out of 5 million) and then crash. We tried loading in different batch sizes of 1000000 rows ,followed by 200,000 rows and then 100000 rows which led to successful execution
* Scripting was a minor problem as we learned Cypher querying while loading data into Neo4j
* Double quotes were used in our ‘description’ column attribute in a particular format which was throwing errors when loading to neo4j
* Most of our attribute names had a ‘hyphen’ in them. Neo4j failed to recognize and read these names as is- .In order to solve this, we cleaned our attribute names and renamed them with an underscore in place of hyphen.
* After successfully loading data on the Neo4j database, there were issues in downloading precise connectors for connecting neo4j to Tableau.

## Velero Portal

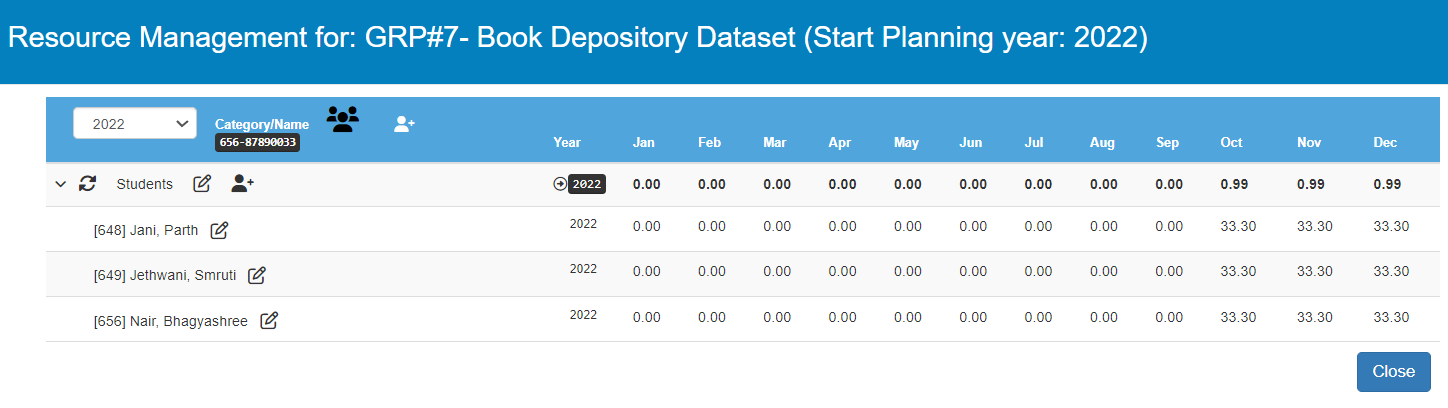
* Project



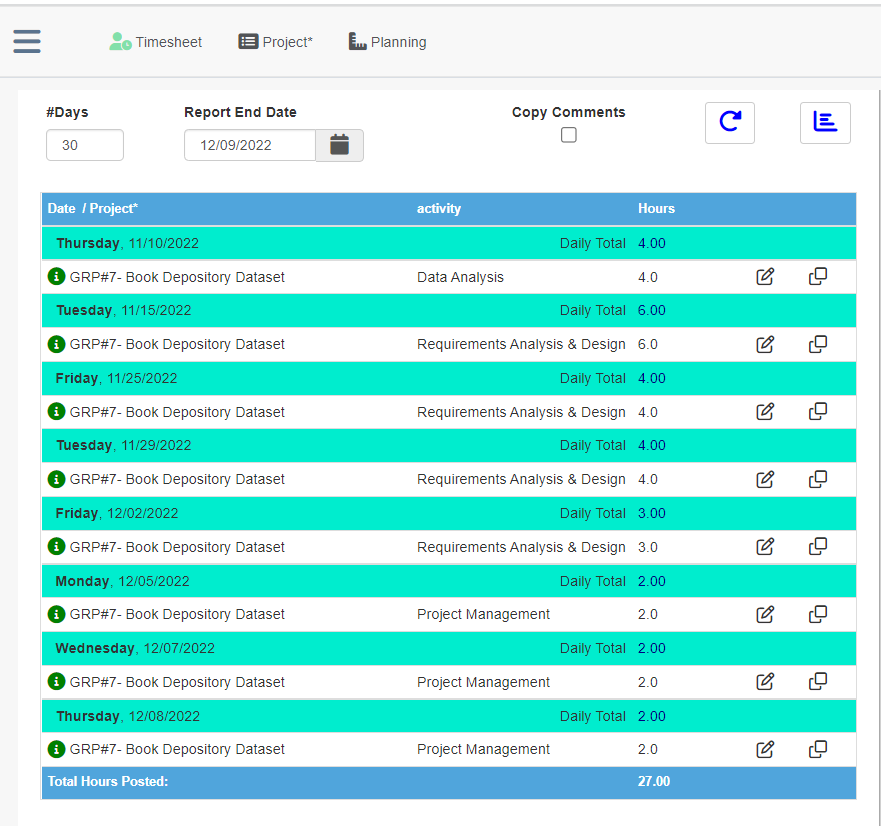
* Project Plan



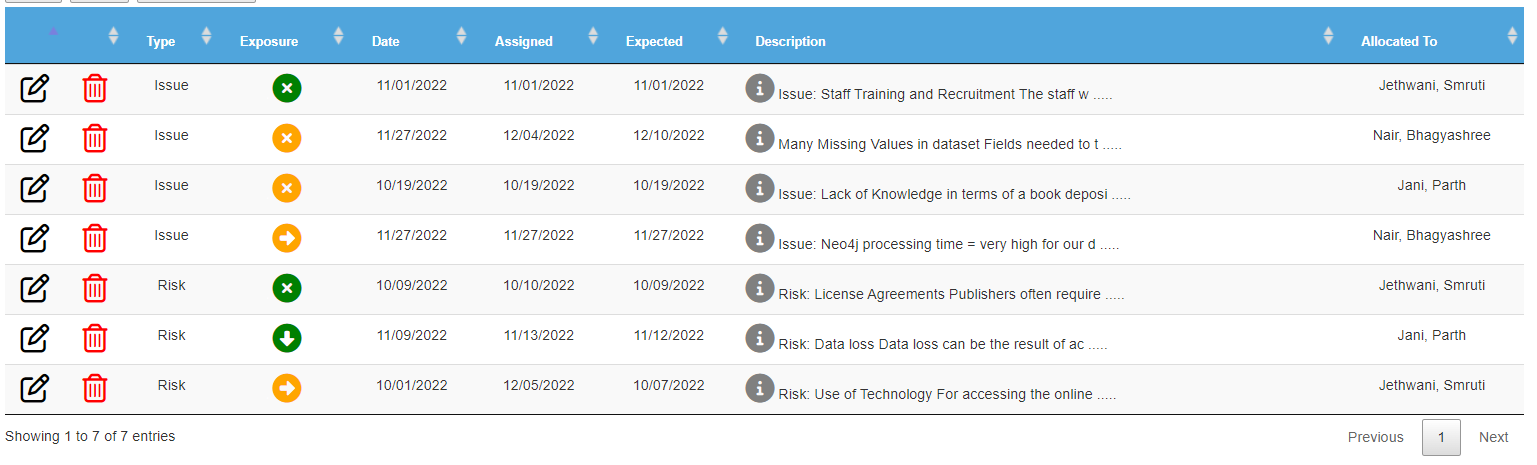
* Resource Allocation



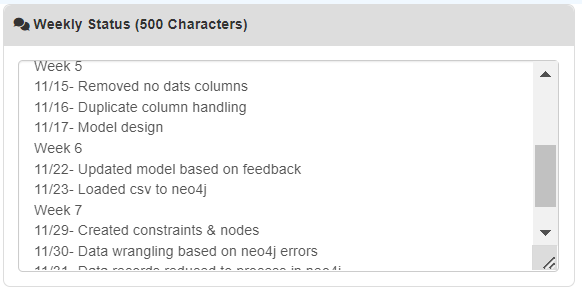
* Timesheet



* Issues & Risks



* Weekly Progress:



## End User Instructions

**End User Instruction:**

This instruction guide will provide you with the steps that you need to follow to set up the environment before starting the project.

**STEPS** **for Installation of softwares:**

1. **Installing Jupyter Notebook interface and downloading important python packages**

* This section includes instructions on how to get started with Jupyter Notebook and installing important python libraries.
* Prerequisite: Python
* While Jupyter runs code in many programming languages, Python is a requirement for installing the Jupyter Notebook. The Python version required differs between Jupyter Notebook releases (e.g. Python 3.6+ for Notebook v6.3, and Python 3.7+ for Notebook v7) .
* We recommend using the [Anaconda](https://www.anaconda.com/download) distribution to install Python and Jupyter.

**Installing Jupyter using Anaconda**

* For new users, we highly recommend [installing Anaconda](https://www.anaconda.com/products/distribution). Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

* Download [Anaconda](https://www.anaconda.com/download). We recommend downloading Anaconda’s latest Python 3 version (currently Python 3.9).
* Install the version of Anaconda which you wish to download.
* Install jupyter notebook by running command pip install notebook
* Then, install dash framework by running pip command pip install jupyter-dash
* After installing dash framework, install other required packages:

· Pandas: pip install pandas

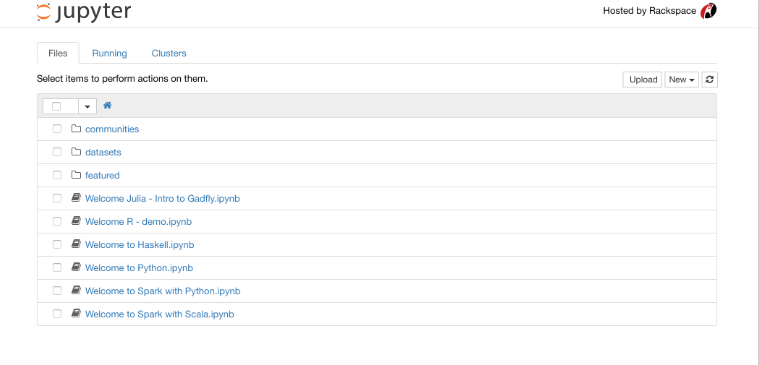
· Matplotlib: pip install matplotlib

· Plotly: pip install plotly=5.7.0

· NumPy: pip install numpy

· Py2Neo: pip install py2neo

* Congratulations, you have installed the Jupyter Notebook.
* You can now run the notebook server. You can start the notebook server from the [command line](https://docs.jupyter.org/en/latest/glossary.html#term-command-line) (using [Terminal](https://docs.jupyter.org/en/latest/glossary.html#term-terminal) on Mac/Linux, [Command Prompt](https://docs.jupyter.org/en/latest/glossary.html#term-Command-Prompt) on Windows) by running: jupyter notebook

When the notebook opens in your browser, you will see the [Notebook Dashboard](https://docs.jupyter.org/en/latest/glossary.html#term-Notebook-Dashboard), which will show a list of the notebooks, files, and subdirectories in the directory where the notebook server was started.

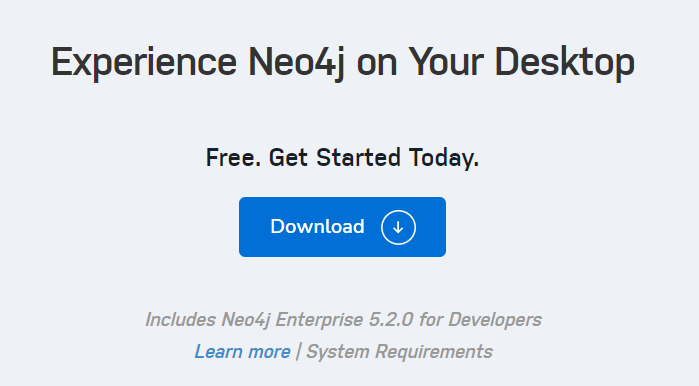
* Open app.ipynb in Jupyter Notebook
* Run app.ipynb script

1. **Installation of Neo4j and creation of database**

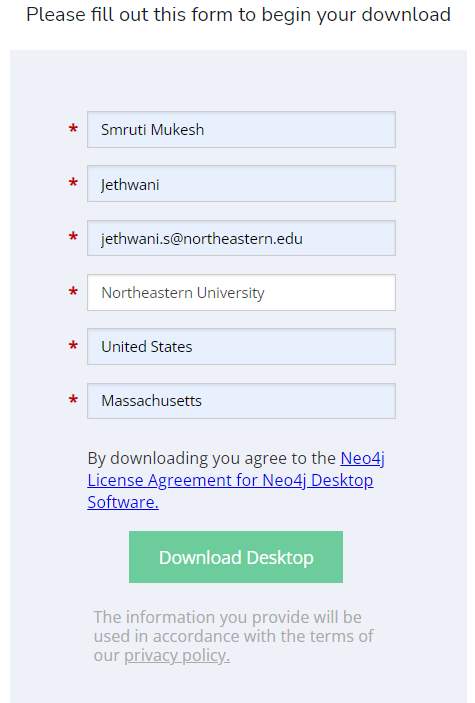
Neo4j is a *native graph database*, which means that it implements a true graph model all the way down to the storage level. Beyond the core graph, Neo4j provides what you’d expect out of a database; ACID transactions, cluster support, and runtime failover.

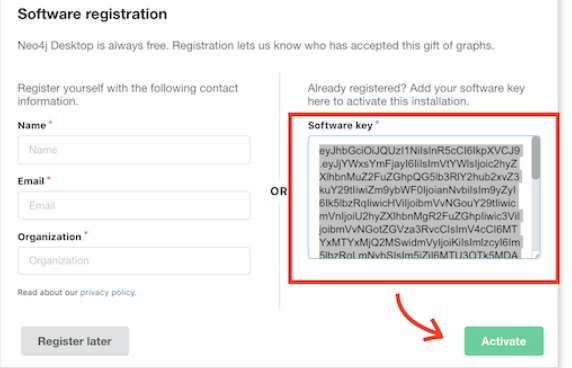
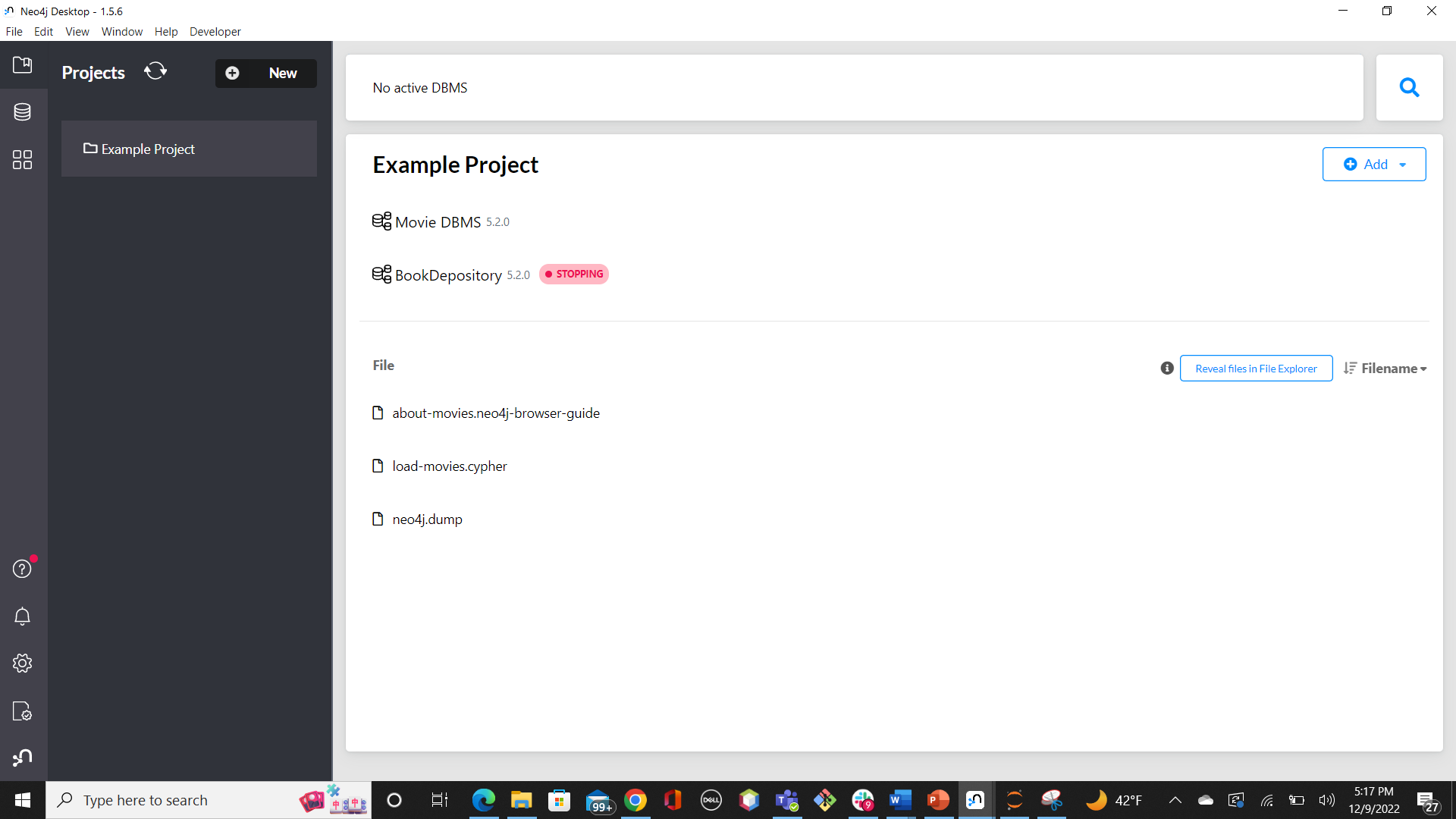
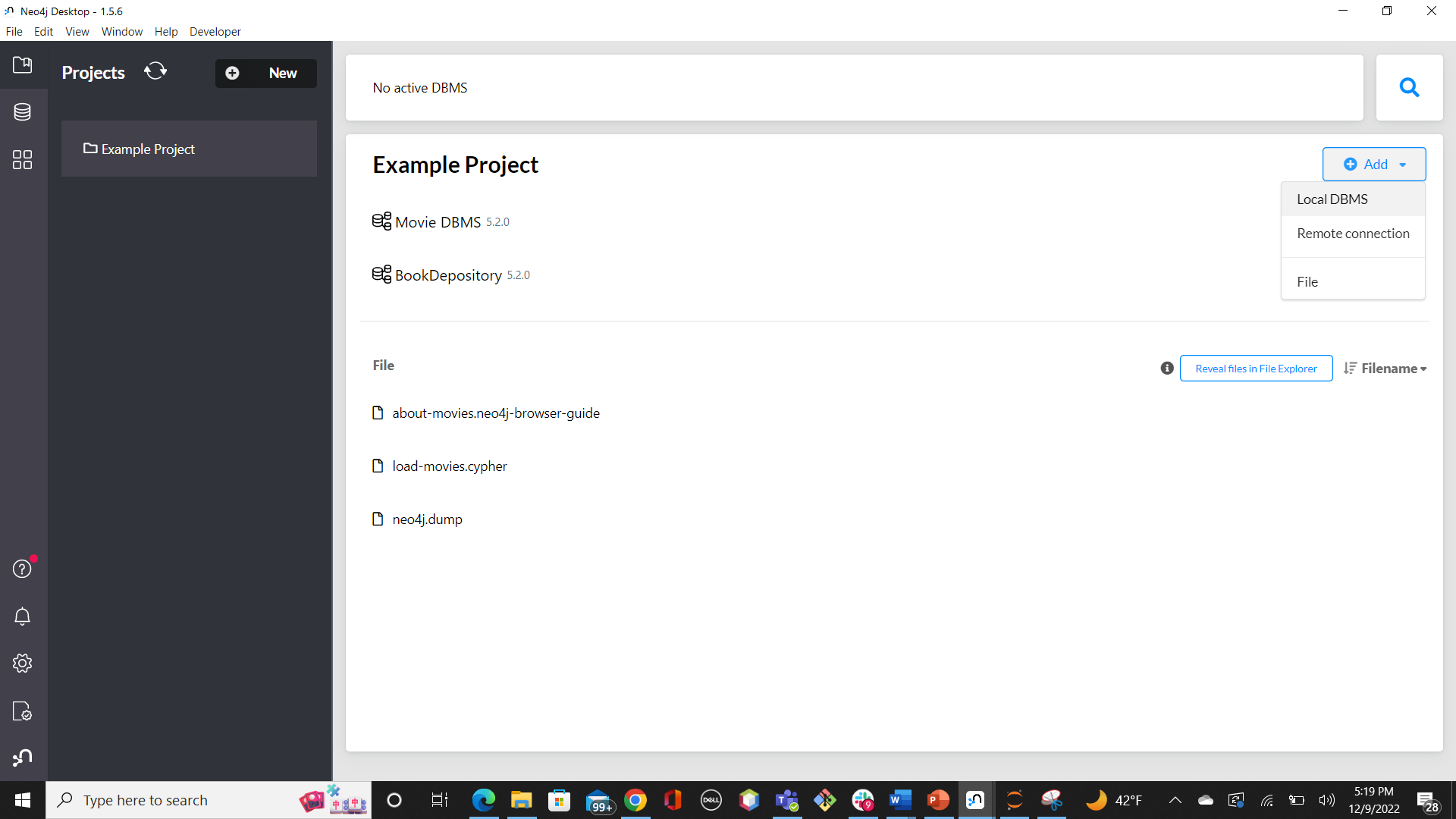
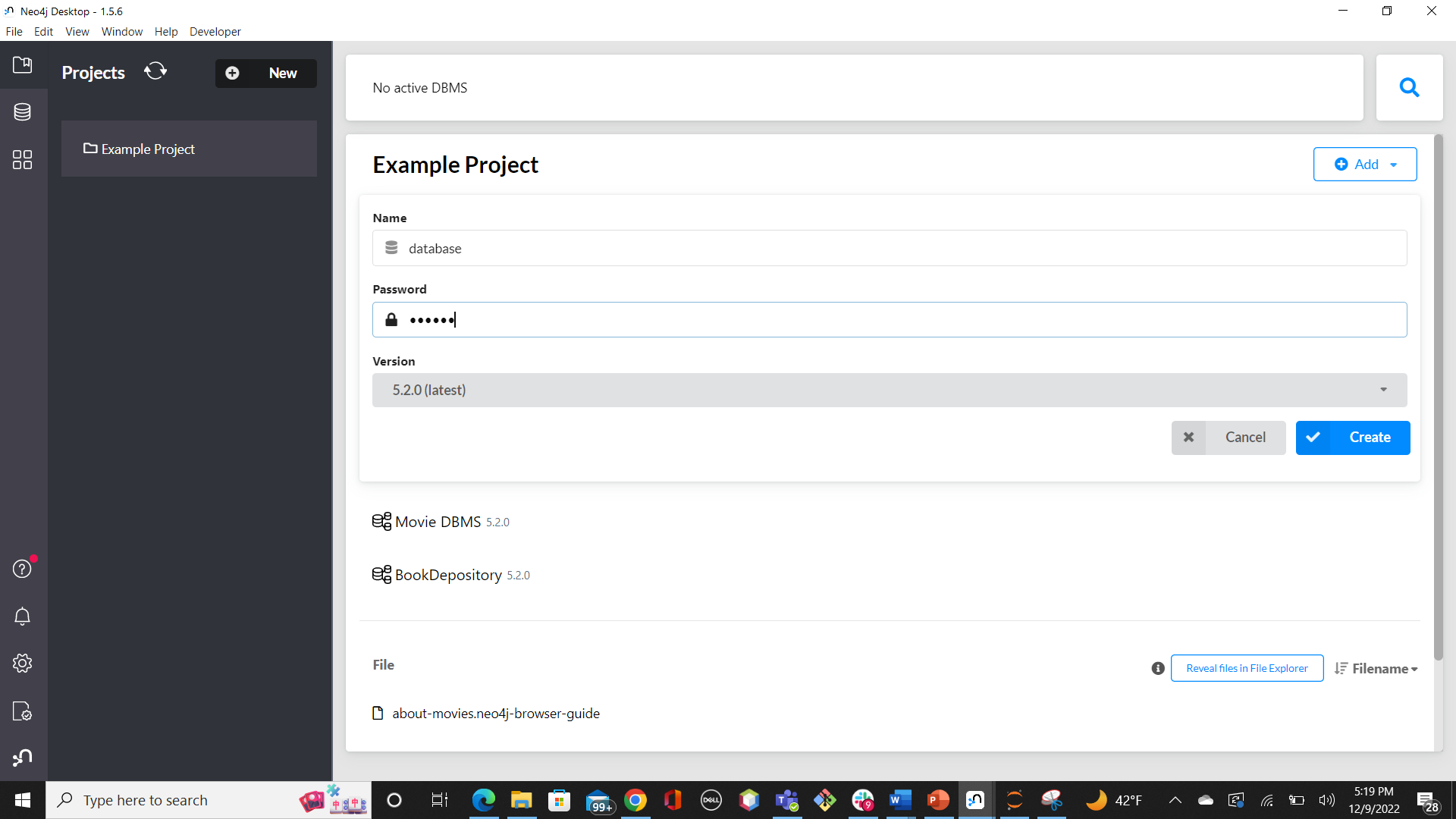
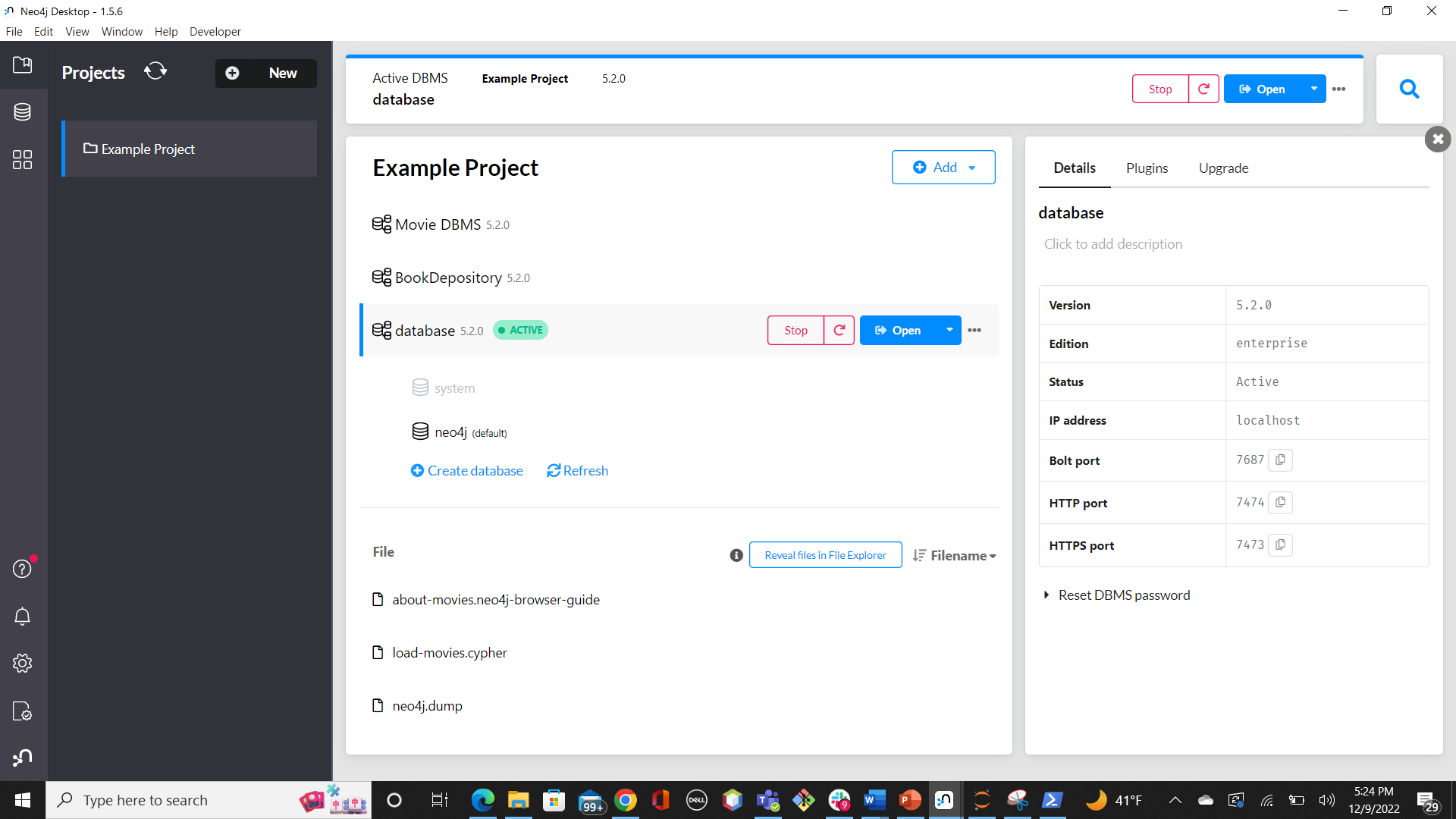
**Steps for Installation**

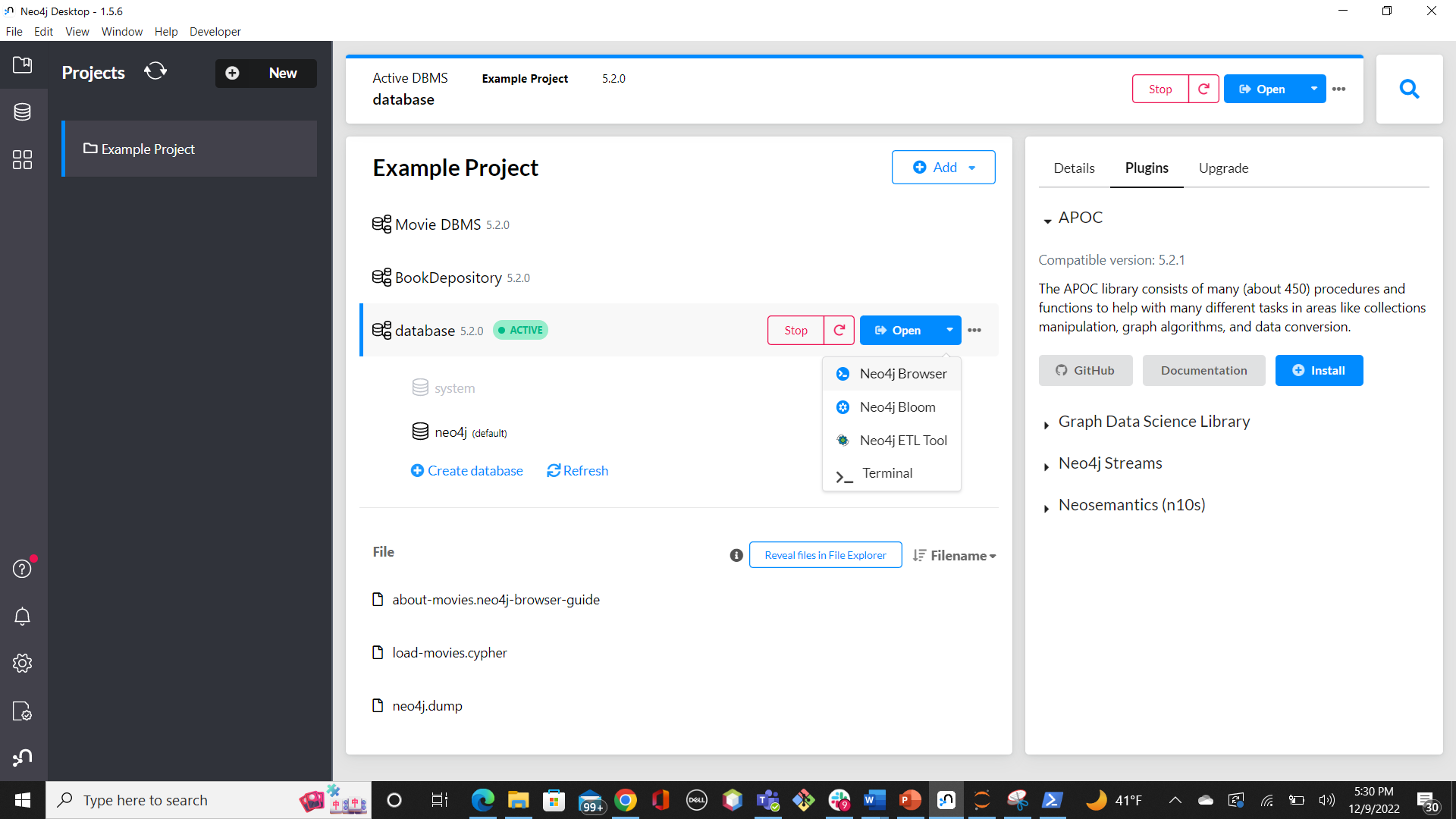
* You can download the Neo4j desktop from the official website. <https://neo4j.com/download/>

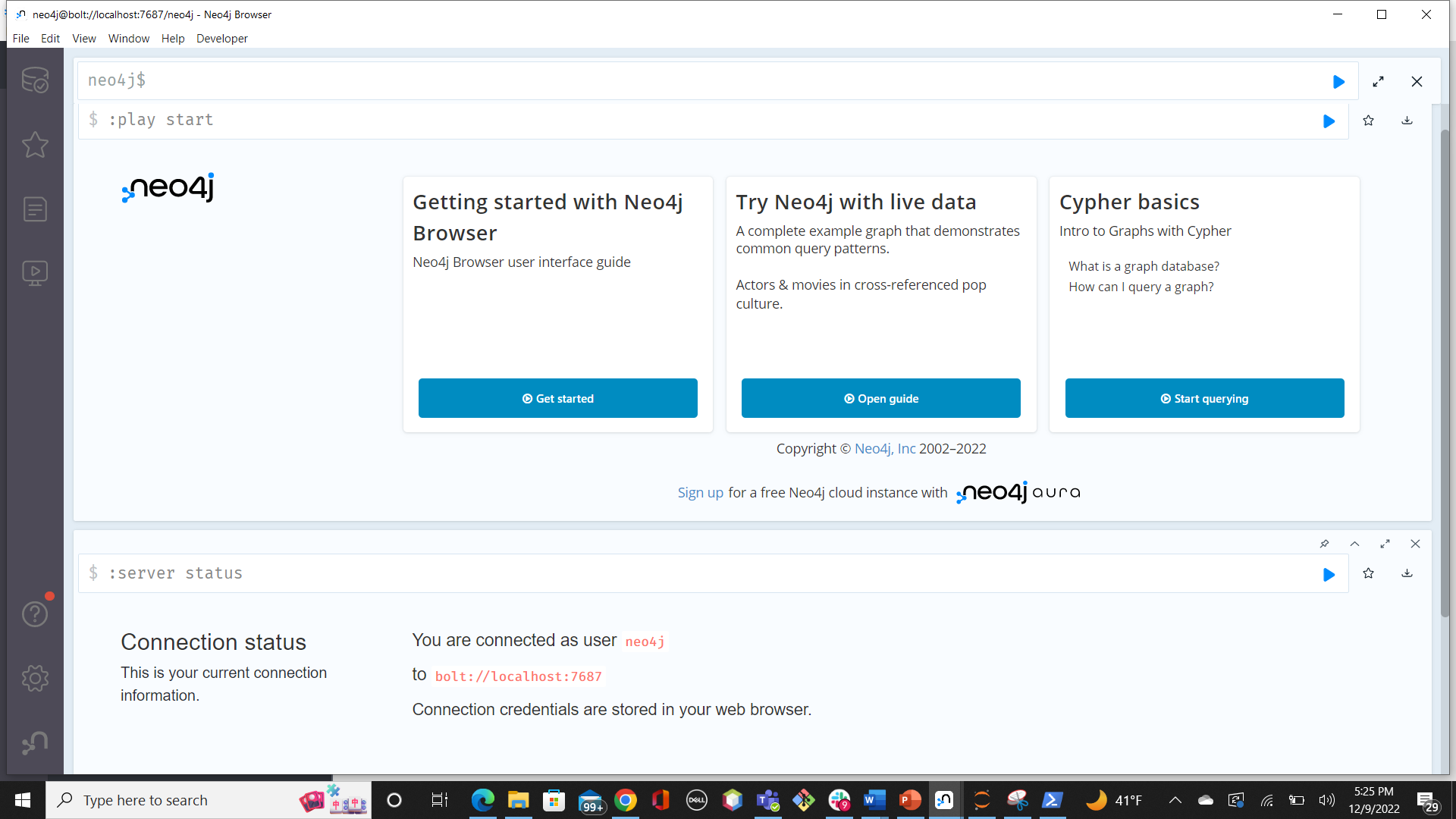


* You will be required to fill a form before downloading, as shown below



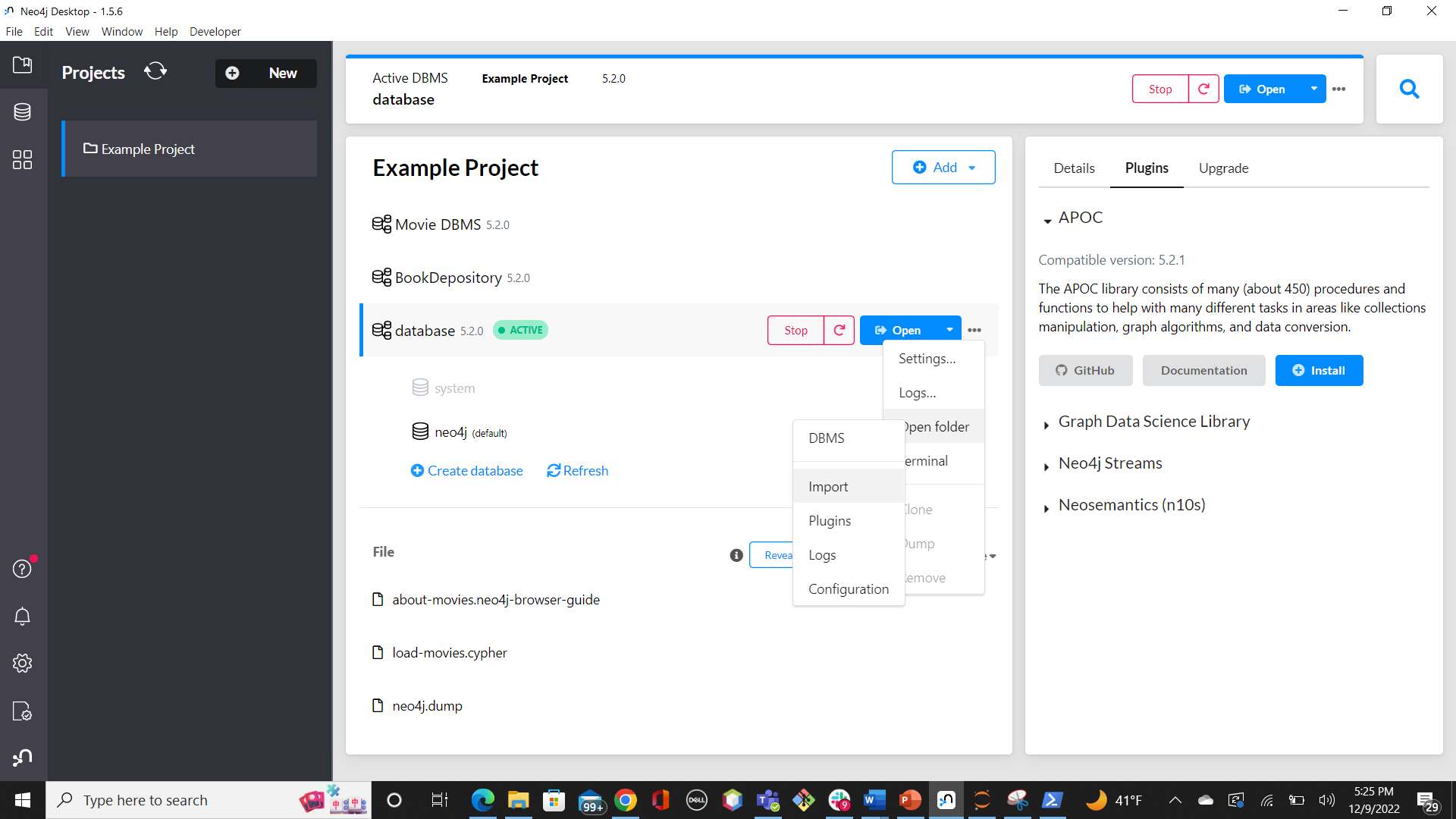
* Inorder to launch the app, Copy and Paste the activation at the top of this page in the "Activation Key" box in the Neo4j Desktop app. Alternatively, you can also generate a key from within the app by filling out the form on the right side of the app screen.
* This is what the desktop version will look like 
* To create a database, click on Add button to the top right and then click on localDBMS
* You may now enter the name of the database and password and click on Create button 
* Once the database is created, click on the Start button to activate it.
* To start querying, you are supposed to open Neo4j Browser. Click on Open—> Neo4j Browser. This will open another window of the browser. You can now start querying.





**Neo4j Browser**

**Importing files in Neo4j:**

To load CSV files, click on the ellipsis icon to the right of the Open button, then click on Open Folder—> Import. This allows you to import any type of files. 

**Loading CSV files in Neo4j:**

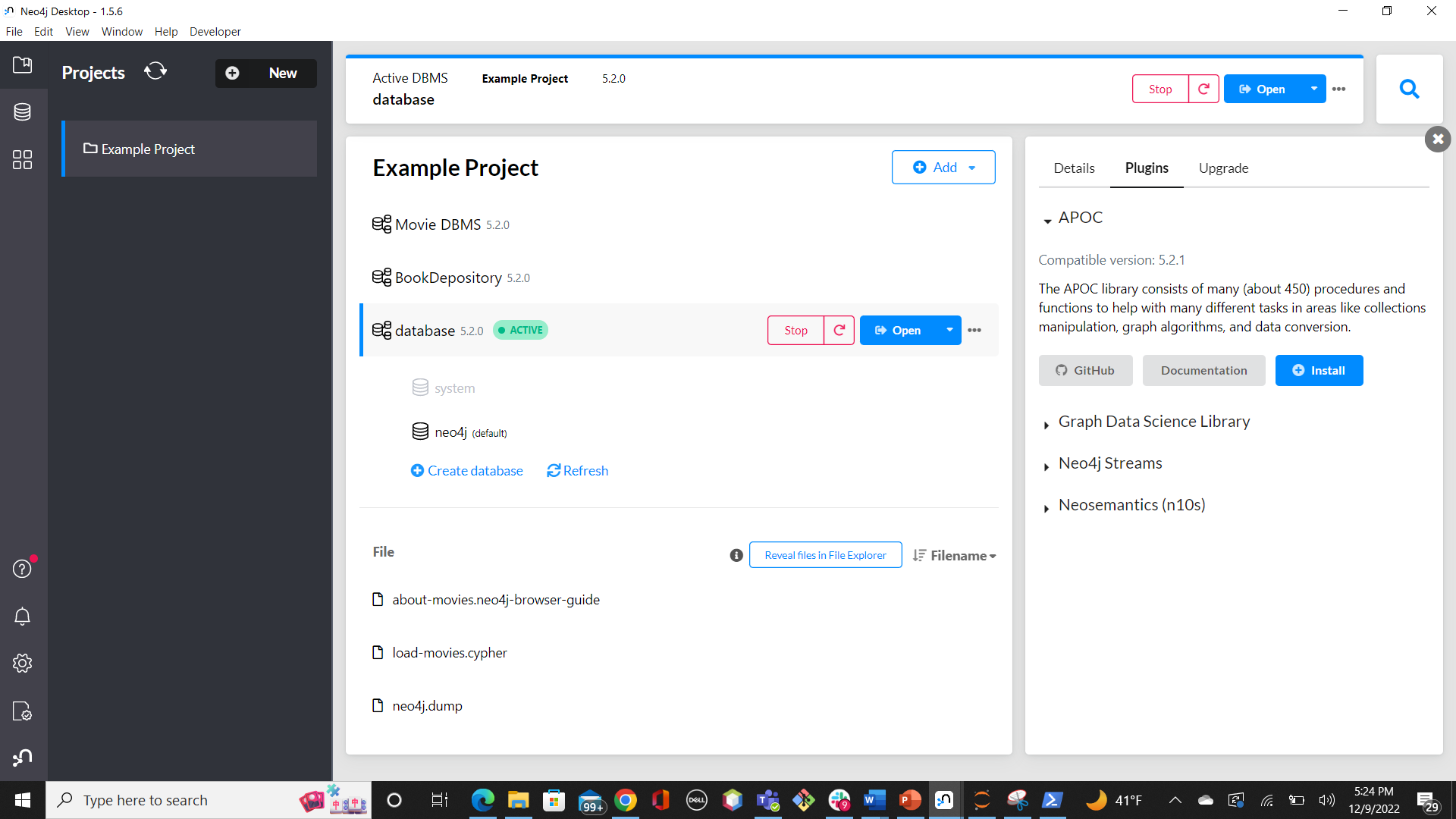
To load the files, the syntax below can be written in the Neo4j Browser



1. **Data Visualization Tool: Tableau**

* Tableau Installation can be done through [Tableau\_Desktop](https://www.tableau.com/support/releases/desktop/2022.3.1#esdalt) as per your operating system.

1. **Connection between Tableau and Neo4j**

* For connection between tableau and neo4j, you require a JDBC connector. You can download it from this link [Neo4j-BI\_JDBC\_connector](https://northeastern.instructure.com/courses/123601/files/18394780/download?download_frd=1).
* You are required to install the APOC (Awesome Procedure on Cypher) library. APOC is an add-on library for Neo4j that provides hundreds of procedures and functions adding a lot of useful functionality and helps in connecting Neo4j to Tableau with ease. You can find this under the Plugins Tab. 

For connection, once you open Tableau, click on Other Databases. Now enter the url which can be obtained from Neo4j browser with the name and password of the database you created and click on Sign in. You have now established a connection between Tableau and Neo4j.

