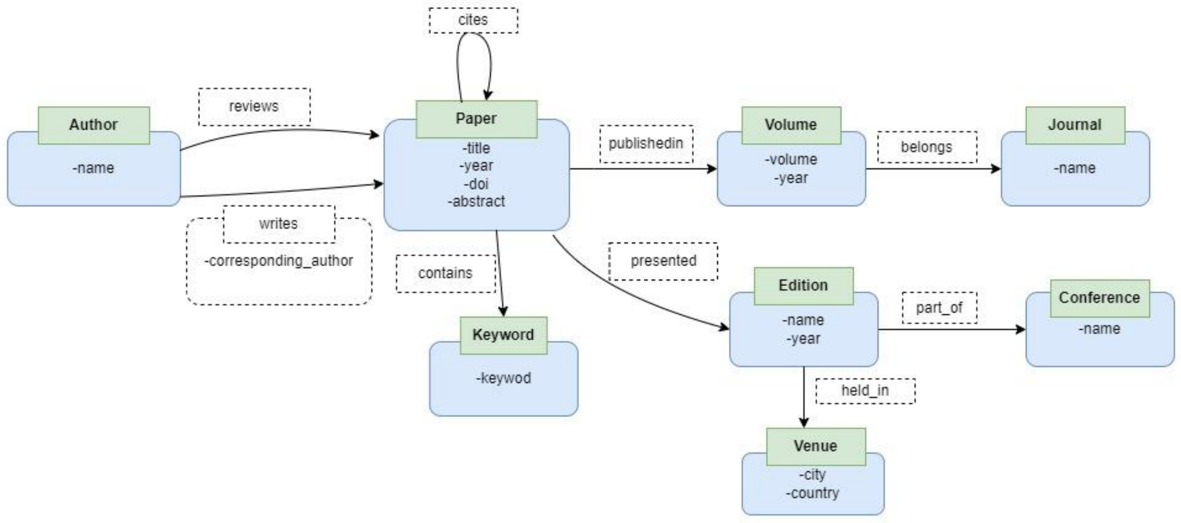
SEMANTIC DATA MANAGEMENT- LAB1 (Property Graphs)

This document includes the solutions of the Property Graphs exercises.

A Modeling, Loading, Evolving

A.1 Modeling

The visual representation of the graph that models research publications can be found below.



After reading the statements for the modeling and taking into account the queries in *Part B*, we eventually designed a graph model consisting of 8 vertices and 9 edges.

Firstly, we began to create the concept `Paper` where each of its instances can either belong to a `conference/workshop` or a `journal`, in `Paper` we add properties of paper's general information that all together can identify uniquely each paper, such as `doi`, `title`, `year`, `etc`. We also add the abstract of the paper as a property, the reason is because we think that it does not play any importance in the paper searching process, as the keywords are more suitable and oftenly used to carry on this task.

Secondly, we created the concept `Author` to allow easy queries access to this information, as a person could be at the same author and reviewer of a paper, it is better to have a concept that can represent both roles than put it as a property of `Paper`. Through the edges `reviews` and `writes` we can easily retrieve author or reviewer's data. As a paper can be written by many authors, we add the property `corresponding_author` to identify who is the main author.

Thirdly, as for papers published in a conference/workshop, we thought that it's better to store `Edition` information in a separate concept rather than in the `Conference` concept . As we could have many editions per year for one conference, when adding a new paper to the DB, we will need to create 2 nodes and 2 relations. In the opposite case, if we store `edition` and `conference` information in one concept, only 1 node and 1 relation are needed to be created. Although in the last case, we reduce the node creation cost in 2 times, generally the task of optimizing the searching cost of nodes is a higher priority than the node creation cost one. Especially in this context of research paper publication, where there is no high volume of node creation involved. Also taking into account the first two queries in *Part B*, it's better to have a `Conference` concept separately because in this way, we can achieve an efficient searching and retrieval of nodes. The same analogy applies to the `Journal` concept.

Finally, we decided to have a concept to store only paper keywords because by doing so, we could avoid keyword duplication. For example, it's quite probable that papers in the same community have the same list of keywords.

A.2 Instantiating/Loading

In order to obtain data to load into the graph we used different sources. We found a Kaggle dataset called BYU Engineering Publications in Scopus 2017-2021 that already has real data about authors, keywords, papers, affiliations, etc. Since the dataset has only data from 2017 to 2021 and from the latter there were few entries, we generated more data for 2021 and also for 2022.
As we needed more information to meet all the requirements, we also used data from DBLP. For this, we transformed the .xml data to .csv using the tool https://github.com/ThomHurks/dblp-to-csv. This generated many .csv files, but we only employed two of them: `output_inproceedings.csv and output_article.csv.`
Finally, in order to get venues for the conferences we used data from this kaggle repo https://www.kaggle.com/datasets/juanmah/world-cities.

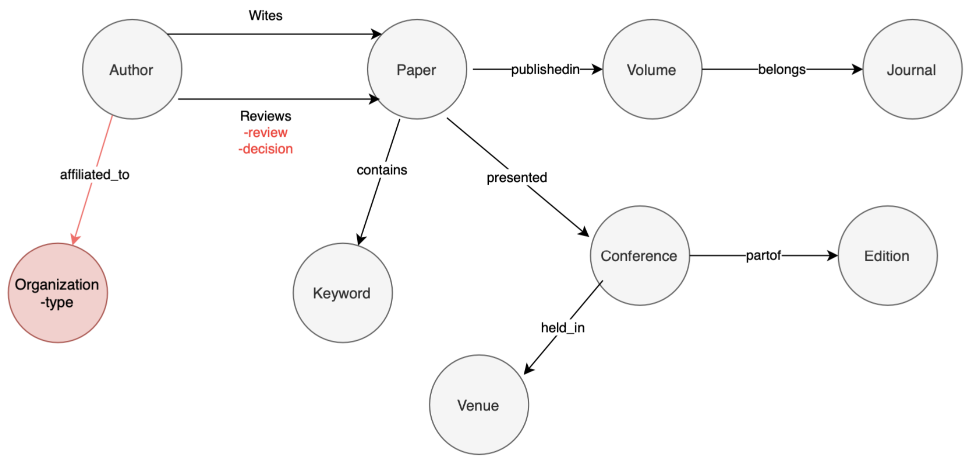
In order to simplify the loading of the data to the graph database we decided to generate one .csv file for each node and one for each relationship in our schema. So, once we gathered all the data we needed from the different repositories , we preprocessed all the files with a script called `'data_generator.ipynb'.` In this script we carried out tasks to clean the data, like removing duplicates, removing rows with no values, etc. and we created all the csv files.

To insert the data into Neo4j we first added all the generated files in the preprocessing step to the import directory in our graph database. Then, we created a script called `PartA.2_XuMoure.py` where we make use of the `LOAD CSV` command. The first step we do is creating the nodes, then we create indexes on the nodes ids, which help to accelerate the creation of the relationships, and finally we create the relationships (the edges) between the nodes.

To be able to generate the data using the script, you must change the database credentials in it to your credentials.
The scripts generated used to carry out the lab are inside the folder called `output`.

A.3 Evolving the graph

In the following image the implemented modifications are highlighted in red.



First, to store the reviews made by a reviewer we decided to add two properties to the edge Reviews. One is the review itself (textual description) and the other the decision.

Second, to store the affiliation of an author we added a node called Organization and we created the edge `affiliated_to` between Author and Organization. To distinguish between the type of organization we added a property called `type` to the organization node.

Since we knew that we were going to have to add affiliations and properties to the review edge, during preprocessing we already created the csv files for this. This was done just to have all the preprocessing done at the beginning of the project, but it could have been done during any stage of it. So, for loading the data to the graph we just created a script called `PartA.3_XuMoure` which is quite similar to the file `PartA.2_XuMoure.py` in the sense that it makes use of the `LOAD CSV` command.

B. Querying

All the queries in this section are available in the file `PartB_XuMoure.` Evidence of the results of the query execution can be found in the Appendix.

1. **Find the top 3 most cited papers of each conference**

```
MATCH
(p:paper)-[c:cites]->(citedPaper:paper)-[pres:presented]->(e:edition)-[part:p
artof]->(conf:conference)
WITH conf, citedPaper, COUNT(c) AS citations
ORDER BY conf.name, citations DESC
WITH collect(citedPaper.title)[..3] as top_papers, conf
RETURN conf.name, top_papers
```

2. **For each conference find its community: i.e., those authors that have published papers on that conference in, at least, 4 different editions.**

```
MATCH
(a:author)-[w:writes]->(p:paper)-[:presented]->(e:edition)-[part:partof]->(co
nf:conference)
```

```
WITH conf, a, collect(DISTINCT e) AS editions
WHERE size(editions) >= 4
RETURN conf.name AS conference, collect(a.name)AS authors
```

---

3. Find the impact factors of the journals in your graph

First we get the number of citations for a journal and year and then we get the number of publications of the two previous years.

```
MATCH
(j:journal)<-[:belongs]-(v:volume)<-[:publishedin]-(p:paper)<-[c:cites]-
(:paper)
WITH j, v.year AS year, toFloat(count(c)) AS citations
CALL {
      WITH year,j
      MATCH (j)<-[:belongs]-(vol:volume)<-[pu:publishedin]-(p:paper)
      WHERE toInteger(vol.year) = toInteger(year)-1 OR toInteger(vol.year) =
toInteger(year)-2
      WITH count(pu) AS publications
      RETURN publications
}
RETURN j.name AS journal, year, CASE publications WHEN 0 THEN 0 ELSE
toFloat(citations) / publications END AS impactFactor
ORDER BY impactFactor DESC
```

---

4. Find the h-indexes of the authors in your graph

The below code checks firstly the percentage of acceptance of the papers (by iterating through the reviewers and getting their final decision on paper publication), if this value is greater than ⅔, then it proceeds to count the number of citations per paper per author. As result we get a list of #citations, by iterating through this list, it calculates h-index using the following rule:

The h-index is equal to i if the value in ith position of the citations list is greater or equal than i.

```
MATCH (p:paper)<-[r:reviews]-(reviewer:author)

WITH p,collect(case when r.decision=true then 1 else 0 end) as votes

WHERE (reduce(s = 0, x IN votes | s + x)/size(votes) ) >= (2/3)

CALL{

MATCH (paper:paper)-[c:cites]->(p)<-[w:writes]-(author:author)

WITH author.name as author,p,count(*) AS citations

ORDER BY citations DESC
with author,collect(citations) as citations

RETURN author,citations

}

UNWIND range(size(citations),1,-1) as h

CALL{

WITH citations,h

WITH citations,h,[i IN range(size(citations) - 1,0,-1 ) | i] AS indices

WITH citations,h,indices

WITH [i IN indices WHERE citations[i] >= h | i] AS filteredIndices

RETURN HEAD(filteredIndices)+1 as hIndex

}

WITH author,citations,h,hIndex

WHERE h=hIndex

RETURN author,citations,h AS hIndex LIMIT 100
```

C. Recommender

The code for this task can be found in the file called PartC_XuMoure.
Part 1

Firstly, we create a new concept `Community` with the name “database”.

```
CREATE(:community {name: 'database'})
```

Secondly, taking into account that what defines a community is a set of keywords, we create an edge called `definedby` between the database community and the keywords provided in the laboratory statement.

```
MATCH (k: keyword)
WHERE toLower(k.keyword) IN ['data management', 'indexing', 'data modeling',
' big data', 'data processing', 'data storage', 'data querying']
MATCH(c:community {name:'database'})
CREATE (c)-[:definedby]->(k)
```

Part 2

For the second stage we used two different queries: one to get the conferences, and another one to get the journals that are related to the database community and that are above the 90% threshold. In each query we create the edge between the conferences and the community and between the journals and the community. To do this, we create an edge called `relates`.

```
MATCH
(p:paper)-[pres:presented]->(e:edition)-[part:partof]->(conf:conference)
WITH conf, toFloat(COUNT(DISTINCT p)) as total_number_papers
CALL {
        WITH conf
        MATCH
        (conf)<-[:partof]-(e:edition)<-[:presented]-(pap:paper)-[:contains]->(
        k:keyword)<-[:definedby]-(c:community {name: 'database'})
        RETURN toFloat(COUNT(DISTINCT pap)) as community_papers, c
}
WITH community_papers / total_number_papers AS proportion_community, conf, c
WHERE proportion_community >= 0.9
CREATE (conf)-[:relates]->(c)
```

```
MATCH (p:paper)-[pub:publishedin]->(v:volume)-[b:belongs]->(jour:journal)
WITH jour, toFloat(COUNT(DISTINCT p)) as total_number_papers
CALL {
        WITH jour
        MATCH
        (jour)<-[:belongs]-(v:volume)<-[:publishedin]-(pap:paper)-[:contains]-
        >(k:keyword)<-[:definedby]-(c:community {name: 'database'})
        RETURN toFloat(COUNT(DISTINCT pap)) as community_papers, c
}
WITH community_papers / total_number_papers AS proportion_community, jour, c
WHERE proportion_community >= 0.9
CREATE (jour)-[:relates]->(c)
```

Part 3

In order to find the top-100 papers we used the Page Rank algorithm provided by Neo4j Graph Data Science library.
We started by creating a projection of the graph (a subgraph) called `page_rank_database_papers`.

```
CALL gds.graph.project.cypher(
'page_rank_database_papers',
'MATCH (n:paper)-[r]->()-[]->()-[:relates]->(c:community { name: "database"
}) WHERE type(r) IN ["publishedin", "presented"] WITH DISTINCT n RETURN id(n)
AS id',
'MATCH (n)-[c:cites]->(p) RETURN id(n) AS source, id(p) AS target',
{validateRelationships: false})
YIELD graphName AS graph, nodeQuery, nodeCount AS nodes, relationshipQuery,
relationshipCount AS rels
```

Then, we calculate the pagerank for within citations and we add a label called `top_100_db_community` to the `paper` node in order to distinguish if they are one of the top 100 papers. We order the results by score and we use the clause `LIMIT` in order to just add the label to the top 100.

```
CALL gds.pageRank.stream('page_rank_database_papers')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS paper
ORDER BY score DESC
```

```
LIMIT 100
SET paper :top_100_db_community
RETURN paper
```

Part 4

Lastly, to get the potential reviewers we just need to query authors that have written papers with the `top_100_db_community` label.

```
MATCH (a:author)-[:writes] -> (p:top_100_db_community)
RETURN DISTINCT a
```

And to get gurus we used the following query.

```
MATCH (a:author)-[:writes]->(p:top_100_db_community)
WITH a, COUNT(p) AS papers
WHERE papers >= 2
RETURN DISTINCT a
```

D. Graph algorithms

In this part we chose two Graph algorithms and applied them to our graph.
All the queries in this section are available in the file `PartD_XuMoure`.

1. Community Detection Algorithms(Louvain)

This algorithm was chosen to find the existent research communities in the DB, get insights of the different research areas and with this information, we can get results of the following commonly queried problems in research article domain:

- What are the latest research findings on “specific community/specific research topic”? → By filtering the keywords of papers in a community per recent date we can retrieve this information.
- Which are the articles published by the most impactful journals during the past two years on “specific community” ? → Through the impact factor we can retrieve this information, by filtering per community.
- What are the most highly cited research articles in “specific community” in recent years?
  We implemented the solution for this query, which is shown below. For that we chose the `Paper` concept and the `cites` relation to do the community detection. The idea is that usually if one paper cites another it’s because they work on similar research topics. Nodes with high degree of connectivity and centrality are good candidates for community detection, in this case these would be the most cited papers in the graph.

  CODE
  ```
  CALL gds.graph.project.cypher(
  'graphPaperc',
  ```

```
 'MATCH (n:paper)-[r:cites]->(p:paper) WITH DISTINCT n RETURN id(n) AS
id',
'MATCH (n)-[c:cites]->(p) RETURN id(n) AS source, id(p) AS target',
{validateRelationships: false})
YIELD graphName AS graph, nodeQuery, nodeCount AS nodes,
relationshipQuery, relationshipCount AS rels
```

```
CALL gds.louvain.stream('graphPaper')
  YIELD nodeId, communityId, intermediateCommunityIds
  WITH gds.util.asNode(nodeId) AS p, communityId
  MATCH (pa:paper)-[r:cites]->(p)
  WITH p,count(*) as citations,communityId
  RETURN p.title as title,communityId,p.year as year,citations
  ORDER BY p.title,communityId,p.year,citations
```

2. Similarity Algorithms(Node similarity)

We chose this algorithm to compute the similarity degree between two papers. The idea was that the more keywords that coincide in two papers, the higher is the probability that they are talking about the same topic. By knowing this information, we can proceed to do the following queries:

- Which research papers are similar to the “provided paper”?
- Which authors have published papers on similar topics to the one I published? Do we have the same research interests?
- Can we identify the actual trending research topics based on the high number of similar papers on one topic?
- Can we detect possible plagiarism actions based on the high similarity degree in content between two papers if they don't share the same list of citations?

The below query retrieves the degree of similarity between two papers in the graph.

CODE

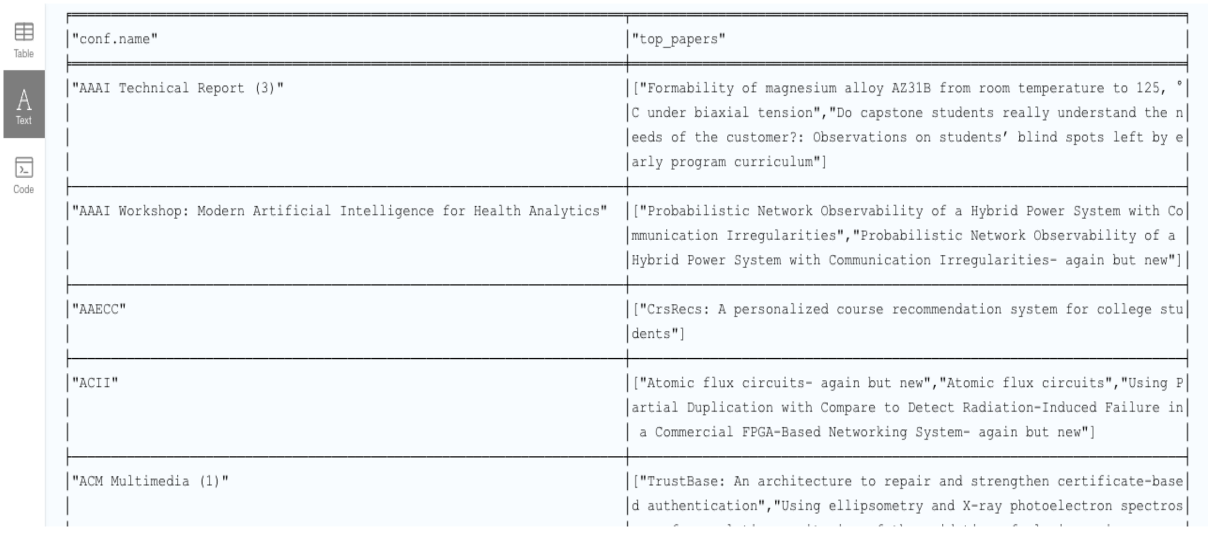
```
CALL gds.graph.project(
        'graphKeywordsp',
        ['paper', 'keyword'],
        {contains: {
                properties: {}
         }})
```

```
 CALL gds.nodeSimilarity.stream('graphKeywordsp')
      YIELD node1, node2, similarity
      WITH similarity,gds.util.asNode(node1) AS
      paper1,gds.util.asNode(node2) AS paper2
      RETURN similarity,paper1.title as title1,paper2.title as title2
      ORDER BY similarity DESCENDING, paper1, paper2
```

Appendix

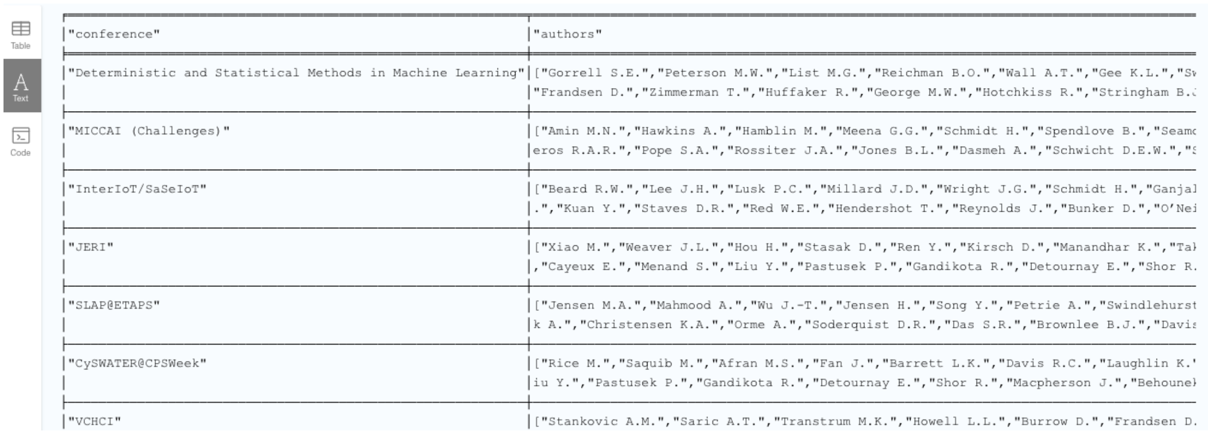
As evidence of the queries in part B, we show a print with just a few results as they were quite big.

Query 1



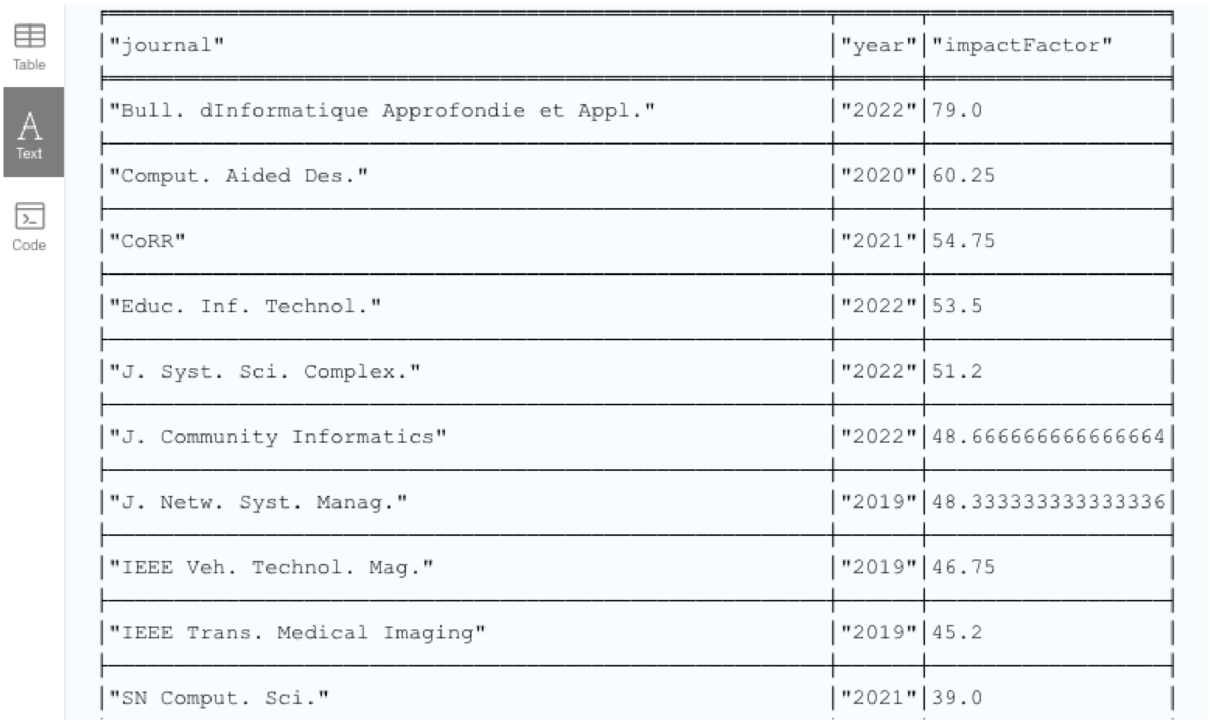
```
|"conf.name"                                                       |"top_papers"                                                        |
|"AAAI Technical Report (3)"                                       |["Formability of magnesium alloy AZ31B from room temperature to 125, °|
|                                                                  |C under biaxial tension","Do capstone students really understand the n|
|                                                                  |eeds of the customer?: Observations on students’ blind spots left by e|
|                                                                  |arly program curriculum"]                                             |
|"AAAI Workshop: Modern Artificial Intelligence for Health Analytics" |["Probabilistic Network Observability of a Hybrid Power System with Co|
|                                                                  |mmunication Irregularities","Probabilistic Network Observability of a |
|                                                                  |Hybrid Power System with Communication Irregularities- again but new"]|
|"AAECC"                                                           |["CrsRecs: A personalized course recommendation system for college stu|
|                                                                  |dents"]                                                               |
|"ACII"                                                            |["Atomic flux circuits- again but new","Atomic flux circuits","Using P|
|                                                                  |artial Duplication with Compare to Detect Radiation-Induced Failure in|
|                                                                  | a Commercial FPGA-Based Networking System- again but new"]           |
|"ACM Multimedia (1)"                                              |["TrustBase: An architecture to repair and strengthen certificate-base|
|                                                                  |d authentication","Using ellipsometry and X-ray photoelectron spectros|
```

Query 2



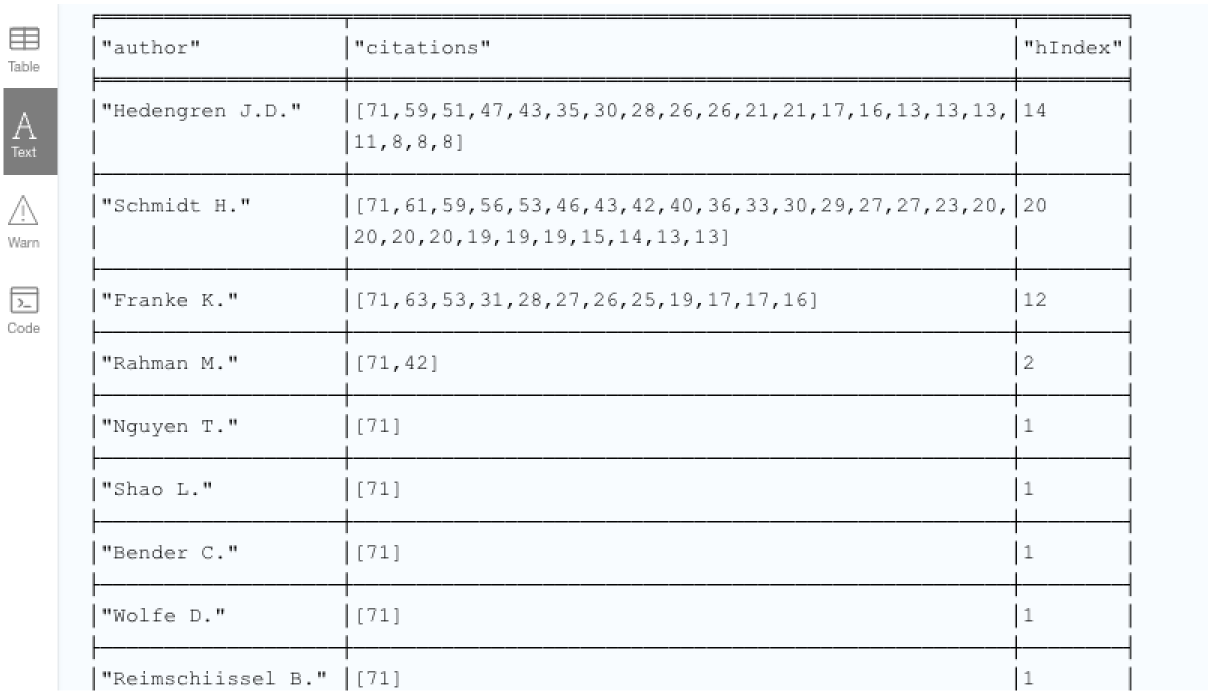
```
|"conference"                                                   |"authors"
|"Deterministic and Statistical Methods in Machine Learning"|["Gorrell S.E.","Peterson M.W.","List M.G.","Reichman B.O.","Wall A.T.","Gee K.L.","S
|                                                               |"Frandsen D.","Zimmerman T.","Huffaker R.","George M.W.","Hotchkiss R.","Stringham B.
|"MICCAI (Challenges)"                                          |["Amin M.N.","Hawkins A.","Hamblin M.","Meena G.G.","Schmidt H.","Spendlove B.","Seam
|                                                               |eros R.A.R.","Pope S.A.","Rossiter J.A.","Jones B.L.","Dasmeh A.","Schwicht D.E.W.","
|"InterIoT/SaSeIoT"                                             |["Beard R.W.","Lee J.H.","Lusk P.C.","Millard J.D.","Wright J.G.","Schmidt H.","Ganja
|                                                               |.","Kuan Y.","Staves D.R.","Red W.E.","Hendershot T.","Reynolds J.","Bunker D.","O’Ne
|"JERI"                                                         |["Xiao M.","Weaver J.L.","Hou H.","Stasak D.","Ren Y.","Kirsch D.","Manandhar K.","Ta
|                                                               |,"Cayeux E.","Menand S.","Liu Y.","Pastusek P.","Gandikota R.","Detournay E.","Shor R.
|"SLAP@ETAPS"                                                   |["Jensen M.A.","Mahmood A.","Wu J.-T.","Jensen H.","Song Y.","Petrie A.","Swindlehurs
|                                                               |k A.","Christensen K.A.","Orme A.","Soderquist D.R.","Das S.R.","Brownlee B.J.","Davi
|"CySWATER@CPSWeek"                                             |["Rice M.","Saquib M.","Afran M.S.","Fan J.","Barrett L.K.","Davis R.C.","Laughlin K.
|                                                               |iu Y.","Pastusek P.","Gandikota R.","Detournay E.","Shor R.","Macpherson J.","Behoune
|"VCHCI"                                                        |["Stankovic A.M.","Saric A.T.","Transtrum M.K.","Howell L.L.","Burrow D.","Frandsen D.
```

Query 3



```
╒══════════════════════════════════════════════════════════╤══════╤═════════════════╕
│"journal"                                                 │"year"│"impactFactor"   │
╞══════════════════════════════════════════════════════════╪══════╪═════════════════╡
│"Bull. dInformatique Approfondie et Appl."                │"2022"│79.0             │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"Comput. Aided Des."                                      │"2020"│60.25            │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"CoRR"                                                    │"2021"│54.75            │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"Educ. Inf. Technol."                                     │"2022"│53.5             │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"J. Syst. Sci. Complex."                                  │"2022"│51.2             │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"J. Community Informatics"                                │"2022"│48.666666666666664│
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"J. Netw. Syst. Manag."                                   │"2019"│48.333333333333336│
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"IEEE Veh. Technol. Mag."                                 │"2019"│46.75            │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"IEEE Trans. Medical Imaging"                             │"2019"│45.2             │
├──────────────────────────────────────────────────────────┼──────┼─────────────────┤
│"SN Comput. Sci."                                         │"2021"│39.0             │
```

Query 4



```
╒════════════════════╤═════════════════════════════════════════════════════╤════════╕
│"author"            │"citations"                                          │"hIndex"│
╞════════════════════╪═════════════════════════════════════════════════════╪════════╡
│"Hedengren J.D."    │[71,59,51,47,43,35,30,28,26,26,21,21,17,16,13,13,13,│14      │
│                    │11,8,8,8]                                            │        │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Schmidt H."        │[71,61,59,56,53,46,43,42,40,36,33,30,29,27,27,23,20,│20      │
│                    │20,20,20,19,19,19,15,14,13,13]                       │        │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Franke K."         │[71,63,53,31,28,27,26,25,19,17,17,16]                │12      │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Rahman M."         │[71,42]                                              │2       │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Nguyen T."         │[71]                                                 │1       │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Shao L."           │[71]                                                 │1       │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Bender C."         │[71]                                                 │1       │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Wolfe D."          │[71]                                                 │1       │
├────────────────────┼─────────────────────────────────────────────────────┼────────┤
│"Reimschiissel B."  │[71]                                                 │1       │
```