Sort it Like You Mean It: Discovering Semantically Interesting Attribute Augmentations to Sort Tables

Akash Khatri
University of Utah
akash.m.khatri@utah.edu

Mir Mahathir Mohammad
University of Utah
mahathir.mohammad@utah.edu

El Kindi Rezig
University of Utah
elkindi.rezig@utah.edu

ABSTRACT

Sorting is a fundamental operation in table analysis. Data scientists frequently sort tables to uncover key insights—for example, identifying the top 10 products by sales. However, this process is largely manual. Data scientists must (1) understand the semantics behind the sorting they wish to apply, and (2) ensure the necessary attributes are present—often requiring manual augmentation of the table. But what if data scientists could receive suggestions for semantically meaningful ways to sort a table, powered by automatic augmentations from a data lake? In this demo, we present InsightSort, an end-to-end system that recommends attribute augmentations to enable richer, more insightful sorting for table exploration. InsightSort works by: (1) discovering potential augmentations by linking the input table with relevant data lake tables, and (2) leveraging a Large Language Model (LLM) to synthesize the top-k diverse sorting attributes based on their semantics. A companion video is available at [1].

1 INTRODUCTION

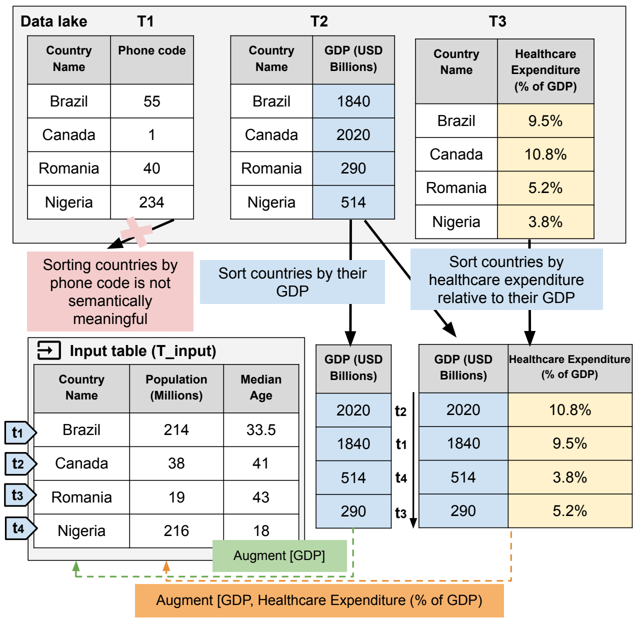
Sorting is a critical operation in both data management and exploratory analysis. For instance, when analyzing a table of employee records, sorting by attributes such as income or seniority can surface meaningful patterns or generate insightful reports. This might reveal, for example, employees who have stayed with the company for many years but have received relatively few raises—highlighting potential disparities or areas for further investigation.

However, this process assumes that users already know which attributes to sort by and that these attributes are readily available in the table. It also requires users to understand the semantics of the desired sorting logic—for example, recognizing that sorting employees by years of seniority may involve computing tenure from hire dates.

Data lakes—whether organizational or public [8]—are becoming increasingly ubiquitous, offering opportunities for data augmentation for several tasks [5]. We propose an approach to identify candidate sorting attribute sets within a data lake to augment a given input table. This enables the table to be sorted in semantically meaningful ways that go beyond its original schema.

Even when users have a general sense of the sorting semantics they want to apply, distinguishing between subtle variations in sorting strategies can be challenging. For example, if users wish to sort countries by "economic performance," potential sorting attributes might include GDP, GDP per capita, GDP growth rate, or inflation rate. Each of these captures a different facet of economic performance, and identifying the most relevant attributes often requires significant manual effort.

What if users could receive recommendations for interesting ways to sort a table—even when the required attributes are not initially present? These attributes could be automatically sourced from a data lake or even generated using a Large Language Model (LLM) [2].


Data lake
T1
T2
T3
Sorting countries by phone code is not semantically meaningful
Sort countries by their GDP
Sort countries by healthcare expenditure relative to their GDP
Input table (T_input)
Augment [GDP]
Augment [GDP, Healthcare Expenditure (% of GDP)]

Country Name	Phone code
Brazil	55
Canada	1
Romania	40
Nigeria	234

Country Name	GDP (USD Billions)
Brazil	1840
Canada	2020
Romania	290
Nigeria	514

Country Name	Healthcare Expenditure (% of GDP)
Brazil	9.5%
Canada	10.8%
Romania	5.2%
Nigeria	3.8%

	Country Name	Population (Millions)	Median Age
t1	Brazil	214	33.5
t2	Canada	38	41
t3	Romania	19	43
t4	Nigeria	216	18

GDP (USD Billions)	
2020	t2
1840	t1
514	t4
290	t3

GDP (USD Billions)	Healthcare Expenditure (% of GDP)
2020	10.8%
1840	9.5%
514	3.8%
290	5.2%

Figure 1: The input table can be augmented with several attributes from the data lake to enable semantically interesting sorting of countries. The tuple IDs are shown next to the suggested augmentation attributes to reflect a descending sort.

Example 1.1. A data scientist, Lou, has an input table of countries with the `Population` and `Median Age` attributes (Figure 1). Lou also has access to a data lake containing numerous tables, but it's unclear which sets of attributes can augment the current table to enable insightful sorting. The data lake tables suitable for augmentation must be joinable with the input table. For instance, joining T_1 with the input table using the columns `Country Name` introduces a new attribute `GDP`, enabling countries to be sorted by GDP, a crucial measure of economic performance.

Another semantically meaningful way to sort the input table is by joining it first with $T2$ and then $T3$, adding two attributes: `GDP` and `Healthcare Expenditure`. These attributes allow for sorting the countries first by GDP and then by healthcare expenditure, highlighting how much countries invest in healthcare relative to their GDP. However, not all sorting attributes are useful. For instance, Sorting countries using their phone code is not semantically meaningful.

We present a demo of InsightSort, an end-to-end system that (1) creates indexes that keep track of the sorting columns in data lake tables; (2) effiently determines, given an input table, what data lake columns are joinable with it; (3) selects sorting attributes that are semantically meaningful; and (4) diversifies the sorting strategies based on their semantic interestingness. Finally, evaluating LLMs for table sorting is an unexplored area. This demo offers VLDB participants a unique opportunity to assess the performance of several leading LLMs—such as GPT-4o [2], Perplexity AI [9], and Claude [3]—on this important table analysis task.

Related work. Several techniques for join discovery have been proposed [4, 11]. In this line of work, users specify a query (e.g., example table) and the goal is to find joinable tables from the data lake. In Metam [5], the goal is to perform attribute augmentation for a given downstream task by specifying its utility function. InsightSort differs from existing work because (1) its goal is orthogonal to the join discovery being used, one can choose any join discovery method to find joinable tables; and (2) it searches for semantically-meaningful sorting attributes to augment tables, which is a target task that has never been explored in prior work.

2 SYSTEM OVERVIEW

Figure 2 illustrates the high-level workflow of InsightSort. In a nutshell, InsightSort ingests a data lake of tables and builds necessary indexes to navigate it in the offline phase. In the online phase, InsightSort takes as input a table. InsightSort finds possible attribute augmentations for the input table from the data lake, scores them by their semantic interestingness, and finally diversifies the sorting results.

2.1 Offline processing of the data lake

In this phase, InsightSort ingests a data lake of tables and indexes it for joinability testing, i.e., given an input column, which data lake columns are joinable with it.

2.1.1 Building the joinability index. Since the data lake can contain a large number of tables, it is important to build indexes that enable efficient joinability testing. To this end, InsightSort proceeds as follows:

(1) We generate vector embeddings for each column in the data lake by following the approach proposed in [4], which leverages the pre-trained language model SBERT [10]. Briefly, this method constructs a textual summary for each column—such as a list of unique values—and then uses the language model to encode these summaries into vector embeddings.
(2) After the vector embeddings of all columns have been created, InsightSort index them using Hierarchical Navigable Small World (HNSW) [7]. This index enables fast Approximate Nearest Neighbor (ANN) queries to check if a query column embedding matches a data lake column embedding within a distance threshold.

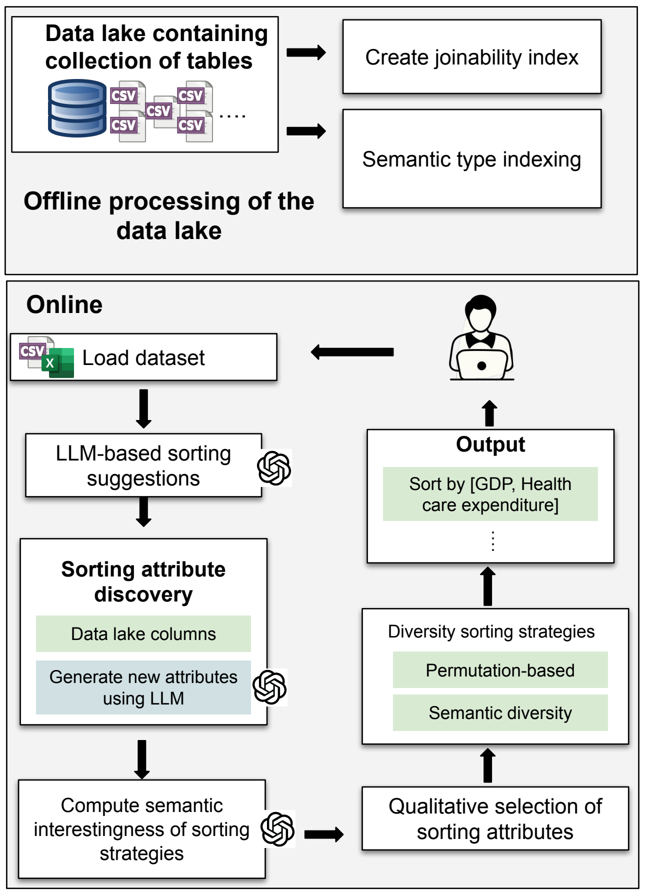

Data lake containing collection of tables
CSV
Create joinability index
Semantic type indexing
Offline processing of the data lake
Online
Load dataset
LLM-based sorting suggestions
Sorting attribute discovery
Data lake columns
Generate new attributes using LLM
Compute semantic interestingness of sorting strategies
Qualitative selection of sorting attributes
Diversity sorting strategies
Permutation-based
Semantic diversity
Output
Sort by [GDP, Health care expenditure]

Figure 2: High-level workflow of InsightSort. InsightSort ingests and indexes a data lake of tables (offline processing), which is then used in the online phase to discover sorting attributes.

2.1.2 Semantic types indexing. InsightSort creates another HNSW index that indexes the vector embeddings of textual description of columns. To this end, InsightSort performs the following:

(1) Because column names are most often not descriptive, for each table, InsightSort prompts the LLM (GPT-4o [2]) with the attribute names and sample values, and requests the semantic types of each column. The semantic type of a column is its textual description. For instance, a column's semantic type can be "US universities" if it contains names of US universities.
(2) InsightSort uses pre-trained language models (sentence transformers [10]) to generate vector embeddings of the semantic types.
(3) Finally, InsightSort indexes those vector embeddings using an HNSW index for fast ANN query answering.

2.2 Online phase

In the online phase, InsightSort leverages the previously created data lake indexes to find joinable columns to a given query table, and possible join columns for a given sorting description.

2.2.1 Uploading the input table. InsightSort allows users to upload their input table. InsightSort will then look for sorting columns by augmenting it with data lake columns or creating new columns using the LLM.

2.2.2 LLM-based sorting suggestions. InsightSort prompts the LLM with the input table schema, and sample rows, and asks for "interesting" sorting attributes that can be added to the input table. The LLM identifies a list of attributes that can be added to sort the rows. For instance, in the example table $T2$ in Figure 1, GPT-4o [2] suggested the sorting attributes in Table 1 (we are only including a sample of them).

2.2.3 Sorting attribute discovery. Given the suggested sorting attributes by the LLM, InsightSort attempts to locate them in the data lake. Of course, those attributes will not necessarily have the same name as suggested by the LLM, so InsightSort proceeds as follows:

(1) **Vector embedding generation:** InsightSort generates vector embeddings of the suggested sorting attribute textual description using a pre-trained language model. InsightSort uses Sentence Bert [10] to produce those embeddings.
(2) **Locating the query attributes:** InsightSort finds the data lake columns that match the query column by performing an ANN query on the semantic type HNSW index. After this, InsightSort has to determine if the tables of those matching columns are joinable with the query table. To this end, it uses the joinability index that was created offline.

Additionally, InsightSort can also ask the LLM to generate a sorting attribute directly. This is appropriate for cases where the sorting attribute is "general-knowledge" and not private (e.g., company-specific data).

Attribute	**Description / Use for Sorting**
Population (millions)	Sort by population size
GDP per Capita (USD)	Economic comparison across countries

Table 1: A sample of the suggested sorting attributes by GPT-4o for table $T2$ in Figure 1.

2.2.4 Computing semantic interestingness of sorting columns. We now define the semantic interestingness of a set of sorting attributes with respect to a query table:

Semantic interestingness of a sorting attribute. Given an input table T and a candidate set of columns $C = \{c_1, c_2, ..., c_m\}$, the semantic interestingness of a given sort strategy (attributes by which to sort) captures the following two components:

- **Semantic utility.** How semantically interesting the attributes in C are in generating an interesting sort order for T ($semantic\_utility(T, C)$). This is computed by prompting an LLM with the column names of T and C and asking it to derive a score for how interesting sorting T with C is.
- **Column uniqueness.** We also want to make sure the attributes in C do not overlap significantly with the attributes in T. That is, we don't want to augment the table with duplicate attributes. Column uniqueness is computed by averaging the cosine similarity between the columns in C and the attributes in T (we get the top-1 match from T for each column in C).

In summary, the semantic interestingness is defined as:

$$I(T, C) = \frac{semantic\_utility(T, C) + uniqueness(T, C)}{2} \quad (1)$$

Seamntic interestingness allows us to determine, for each joinable table in the data lake, whether adding its attributes to the input table would result in semantically-interesting sorting attributes.

2.2.5 Diversifying the results. InsightSort returns a set of *diverse* sorting columns. That is, we want to make sure we minimize duplicate sorting attributes. To this end, we adopt two strategies:

- **Permutation-based:** for two given sorting attributes A and B, applied on the input table T with rows identifiers $\{1, 2, ..., n\}$, result in row permutations S_A and S_B. That is, A and B result in different shuffling of the input table's rows.
  Because we want to produce diverse sorted tables, this means that two sorted tables should have significantly different permutations of rows. To this end, we use the hamming distance or levenshtein distance between the permutation arrays of two sorted tables to determine if they are diverse if the distance is greater than θ. Presumably, if two sorting attributes are correlated (e.g., GPD and HDI), they are likely to result in the same permutation of rows.
- **Semantic diversity:** In this strategy, we want to diversify sorting attributes if they are "semantically" diverse. To this end, InsightSort (1) generates the textual description of each sorting strategy using the LLM; (2) generates vector embeddings of those descriptions using a pre-trained language model (e.g., sentence transformers); and (3) uses an HNSW index to determine if two sorting strategies are semantically similar, i.e., if they are within a given distance threshold d.

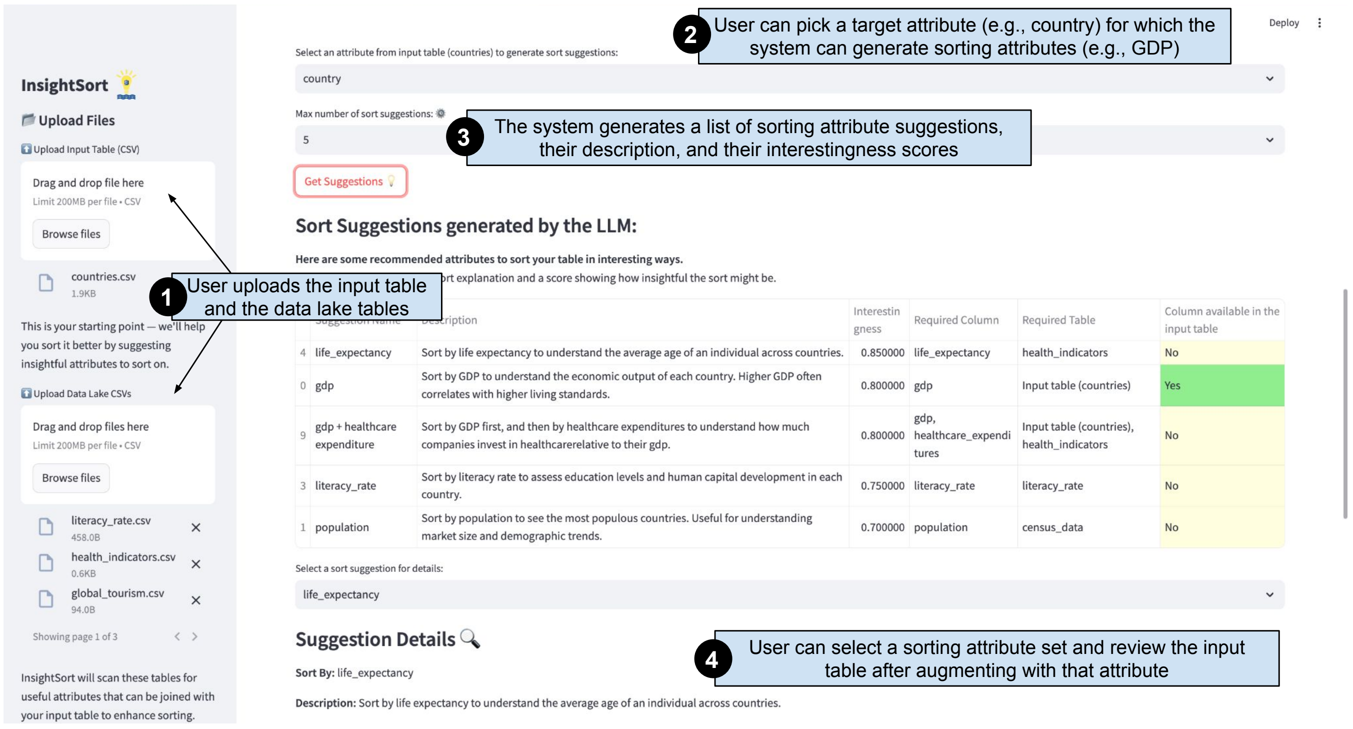
3 DEMONSTRATION PLAN

We demonstrate InsightSort on the following real-world datasets: (1) Stackoverflow survey dataset [1]; (2) Adult dataset [2]; and (3) Chicago crime dataset [3]. As for the data lakes, we will be using a repository of over 1000 CSV files that we crawled from data.gov.

Demonstration outline. In this demo, we aim to showcase (1) the interactive interface of InsightSort that allows users to explore various table augmentations for different sorting semantics; (2) the importance of interestingness scores in exposing various "interesting" sorting attribute sets; and (3) the system's diversity sorting which allows users to inspect sorting attributes that are semantically unique. We now present the demonstration scenario:

① **Uploading the input table and the data lake.** The participants start by uploading their input table as well as the table repository from which InsightSort will discover augmentations (Figure 3 ❶).
② **Customizing joinability thresholds.** Once the data lake has been uploaded, InsightSort finds joinable columns with the input

[1] https://survey.stackoverflow.co
[2] https://www.kaggle.com/datasets/wenruliu/adult-income-dataset
[3] https://www.chicago.gov/city/en/dataset/crime.html


InsightSort
Upload Files
Upload Input Table (CSV)
Drag and drop file here
Limit 200MB per file • CSV
Browse files
countries.csv
1.9KB
Upload Data Lake CSVs
Drag and drop files here
Limit 200MB per file • CSV
Browse files
literacy_rate.csv
458.0B
health_indicators.csv
0.6KB
global_tourism.csv
94.0B
Showing page 1 of 3
Deploy
Get Suggestions
1 User uploads the input table and the data lake tables
2 User can pick a target attribute (e.g., country) for which the system can generate sorting attributes (e.g., GDP)
3 The system generates a list of sorting attribute suggestions, their description, and their interestingness scores
4 User can select a sorting attribute set and review the input table after augmenting with that attribute

This is your starting point — we'll help you sort it better by suggesting insightful attributes to sort on.

InsightSort will scan these tables for useful attributes that can be joined with your input table to enhance sorting.

Select an attribute from input table (countries) to generate sort suggestions: country

Max number of sort suggestions: 5

Sort Suggestions generated by the LLM:

Here are some recommended attributes to sort your table in interesting ways.

ort explanation and a score showing how insightful the sort might be.

	Suggestion Name	Description	Interestingness	Required Column	Required Table	Column available in the input table
4	life_expectancy	Sort by life expectancy to understand the average age of an individual across countries.	0.850000	life_expectancy	health_indicators	No
0	gdp	Sort by GDP to understand the economic output of each country. Higher GDP often correlates with higher living standards.	0.800000	gdp	Input table (countries)	Yes
9	gdp + healthcare expenditure	Sort by GDP first, and then by healthcare expenditures to understand how much companies invest in healthcarerelative to their gdp.	0.800000	gdp, healthcare_expenditures	Input table (countries), health_indicators	No
3	literacy_rate	Sort by literacy rate to assess education levels and human capital development in each country.	0.750000	literacy_rate	literacy_rate	No
1	population	Sort by population to see the most populous countries. Useful for understanding market size and demographic trends.	0.700000	population	census_data	No

Select a sort suggestion for details: life_expectancy

Suggestion Details

Sort By: life_expectancy

Description: Sort by life expectancy to understand the average age of an individual across countries.

Figure 3: The user interface of InsightSort. Participants will have the opportunity to discover several sorting attribute sets with different semantics.

table. Participants can adjust the joinability thresholds that are internally used by InsightSort to determine if two columns are joinable. The goal here is to show that less joinable columns can create several spurious augmentations.

③ **Picking a target attribute.** InsightSort allows users to pick a particular target attribute (e.g., country) for which InsightSort will discover sorting attributes (e.g., GDP) as illustrated in Figure 3 ❷. Users can also elect not to choose a target attribute, in which case InsightSort will discover sorting attributes for all possible attributes in the input table.

④ **Sorting attribute suggestions.** InsightSort discovers a list of sorting attributes from the data lake, and displays their description, as well as their interestingness scores to the user (Figure 3 ❸).

⑤ **Input table augmentation.** Upon selecting a particular sorting attribute set suggestion, InsightSort proceeds to augment the input table with the selected attribute set (Figure 3 ❹).

⑥ **Comparing sorting attributes.** InsightSort allows users to compare various sorting sets based on how unique they are. This uniqueness is determined by how similar the sorting permutes the tuples of the input table, and how semantically unique it is. Users can adjust the thresholds of diversity to explore different sorting suggestions. The goal is to demonstrate that including no diversity filtering results in a large amount of similar sorting suggestions.

⑦ **Comparing different LLMs.** InsightSort will also feature connections to LLMs other than GPT-4o. Those include Gemini [6], Perplexity AI [9], and [3]. This will allow the participants to select sorting suggestions across different LLMs, which would provide a vast array of options.

Demonstration engagement. In addition to a guided demonstration of InsightSort, we will encourage participants to upload their own datasets to perform an interactive table exploration session on them. Participants will be able to interactively examine sorting attributes that are semantically relevant to their task and compare them across different LLMs, joinability thresholds, and interestingness scores.

REFERENCES

[1] [n.d.]. InsightSort demo video. https://drive.google.com/drive/folders/1Hke2MTKtGF0FRHmrbOwx-nNIFPUcojC9?usp=sharing
[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv:2303.08774* (2023).
[3] Anthropic. 2023. *Claude.* https://www.anthropic.com/index/introducing-claude Constitutional AI-based large language model.
[4] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. DeepJoin: Joinable Table Discovery with Pre-Trained Language Models. *Proc. VLDB Endow.* 16, 10 (June 2023), 2458–2470. https://doi.org/10.14778/3603581.3603587
[5] Sainyam Galhotra, Yue Gong, and Raul Castro Fernandez. 2023. METAM: Goal-Oriented Data Discovery. *arXiv preprint arXiv:2304.09068* unspecified, unspecified (2023), unspecified–unspecified.
[6] Google DeepMind. 2023. *Gemini.* https://deepmind.google/technologies/gemini/ Multimodal large language model.
[7] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (April 2020), 824–836.
[8] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data lake management: challenges and opportunities. *Proc. VLDB Endow.* 12, 12 (Aug. 2019), 1986–1989. https://doi.org/10.14778/3352063.3352116
[9] Perplexity AI. 2023. *Perplexity AI.* https://www.perplexity.ai
[10] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).
[11] El Kindi Rezig, Anshul Bhandari, Anna Fariha, Benjamin Price, Allan Vanterpool, Vijay Gadepally, and Michael Stonebraker. 2021. DICE: data discovery by example. (2021).