



# TABLECOPILOT: A Table Assistant Empowered by Natural Language Conditional Table Discovery

Lingxi Cui

The State Key Laboratory of  
Blockchain and Data Security,  
Zhejiang University  
Hangzhou High-Tech Zone (Binjiang)  
Institute of Blockchain and Data  
Security  
Hangzhou, China  
cuilingxi.cs@zju.edu.cn

Guanyu Jiang

The State Key Laboratory of  
Blockchain and Data Security,  
Zhejiang University  
Hangzhou, China  
djangoo40805@zju.edu.cn

Huan Li\*

The State Key Laboratory of  
Blockchain and Data Security,  
Zhejiang University  
Hangzhou High-Tech Zone (Binjiang)  
Institute of Blockchain and Data  
Security  
Hangzhou, China  
lihuan.cs@zju.edu.cn

Ke Chen

The State Key Laboratory of  
Blockchain and Data Security,  
Zhejiang University  
Hangzhou High-Tech Zone (Binjiang)  
Institute of Blockchain and Data  
Security  
Hangzhou, China  
chenk@zju.edu.cn

Lidan Shou\*

The State Key Laboratory of  
Blockchain and Data Security,  
Zhejiang University  
Hangzhou High-Tech Zone (Binjiang)  
Institute of Blockchain and Data  
Security  
Hangzhou, China  
should@zju.edu.cn

Gang Chen

The State Key Laboratory of  
Blockchain and Data Security,  
Zhejiang University  
Hangzhou, China  
cg@zju.edu.cn

## ABSTRACT

The rise of LLM has enabled natural language-based table assistants, but existing systems assume users already have a well-formed table, neglecting the challenge of table discovery in large-scale table pools. To address this, we introduce TABLECOPILOT, an LLM-powered assistant for interactive, precise, and personalized table discovery and analysis. We define a novel scenario, NLCDD, where users provide both a natural language condition and a query table, enabling intuitive and flexible table discovery for users of all expertise levels. To handle this, we propose CROFUMA, a cross-fusion-based approach that learns and aggregates single-modal and cross-modal matching scores. Experimental results show CROFUMA outperforms SOTA single-input methods by at least 12% on NDCG@5. We also release an instructional video, codebase, datasets, and other resources on GitHub to encourage community contributions. TABLECOPILOT sets a new standard for interactive table assistants, making advanced table discovery accessible and integrated.

## PVLDB Reference Format:

Lingxi Cui, Guanyu Jiang, Huan Li, Ke Chen, Lidan Shou and Gang Chen. TABLECOPILOT: A Table Assistant Empowered by Natural Language Conditional Table Discovery. PVLDB, 18(12): 5399 - 5402, 2025. doi:10.14778/3750601.3750681

\*Huan Li and Lidan Shou are the corresponding authors.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097. doi:10.14778/3750601.3750681

## PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at <https://sudis-zju.github.io/table-copilot/>.

## 1 INTRODUCTION

The rise of large language models (LLMs) has extended their capabilities to tabular data, enabling the development of table assistants for tasks such as TableQA, tabular data manipulation and analytics [12]. These approaches have achieved notable success in downstream tasks, often assuming that users already have a table with sufficient data. However, in practice, users may start without a table or need to enhance an existing one before the analysis begins [5]. This introduces the need for an additional, often overlooked step: **table discovery** over large-scale table pools, which has received limited attention in the context of table assistants.

To empower table assistants with table discovery capabilities, we face two primary challenges. (1) **LLM Incompatibility with Table Discovery**: The foundation of table assistants, LLMs, are not inherently optimized for table discovery tasks. Importing a large-scale pool with millions of tables into an LLM is impractical due to token limits and privacy concerns. Moreover, LLMs trained primarily on sequential textual data struggle to interpret structured, tabular data, which are of two-dimensional nature. This necessitates the use of specialized table discovery methods. (2) **Gaps in Existing Table Discovery Methods**: Current methods typically rely on either keyword(s) or a table as a query to retrieve relevant tables. However, keywords and single-table queries often fail to fully capture user intent, particularly in interactive table assistants where users may specify detailed natural language requirements (see Example 1). Traditional methods focusing solely on keywords or query tables are therefore insufficient for such use cases.

**Example 1.** Suppose a recruiter wants to populate a shortlist table with applicants from an application database for subsequent competence evaluation. This shortlist table includes fields such as name, graduation year, CS grade. The recruiter is looking for records that can be directly filled into this table, with a specific request: applicants must graduate in 2023 or 2024, and have a CS grade above 90. In this context, keyword-based table discovery methods cannot handle such complex and personalized NL requirements, and the retrieved tables cannot be directly unioned with the shortlist. Moreover, query-table-based discovery methods are even less applicable, as the retrieved tables cannot meet the NL requirements. Therefore, a table assistant with discovery capabilities is the ideal solution. It is designed for users unfamiliar with database structures, meets complex and personalized needs, and facilitates further analysis and processing of data.

To empower LLM-based table assistants with table discovery capabilities, we make the following efforts:

- (1) **New User Scenario.** We introduce a new user scenario: *Natural Language Conditional Table Discovery* (NLCTD), allowing users to provide both natural language (NL) requests and a query table. By incorporating NL conditions into NLCTD, we bridge the gap between LLM-based table assistants and table discovery. This integration enables interactive, precise, and personalized searches, enhancing user experience. Moreover, detailed NL conditions facilitate highly efficient searches across vast table repositories, reducing LLM calls and improving response speed, scalability, and cost-effectiveness.
- (2) **Cross-Fusion-Based Solution.** To close the gap, we propose CROFUMA, a novel *cross-fusion-based matching* method for table discovery. CROFUMA calculates single-modal table matching scores and cross-modal NL condition matching scores separately, then aggregates them during online query processing (Section 2.3). CROFUMA can handle situations with only NL or a single table as input, making it compatible with traditional table discovery scenarios. In the offline phase, we employ pretrained language models (PLMs) as encoders for both NL and tables. To better capture table structure and semantics, we enhance PLMs through contrastive learning during pretraining. Table contents and metadata are encoded separately and indexed using HNSW [6, 7], achieving an average query time of under 500 ms and enabling real-time interactions with LLMs (Section 2.2). Experimental results show that our CROFUMA outperforms SOTA single-input discovery methods by at least 12% on NDCG@5 in the NLCTD scenario (Section 2.4).
- (3) **End-to-end Prototype.** We present TABLECOPILLOT, a fully functional prototype that seamlessly integrates CROFUMA into an LLM-powered table assistant. TABLECOPILLOT supports the entire workflow from NL-conditional table discovery to downstream applications like TableQA. To ensure robustness and facilitate evaluation, TABLECOPILLOT also incorporates existing table discovery methods for direct comparison (search panel in Figure 2(a)). This design demonstrates the practicality and superiority of equipping LLM-based assistants with native, efficient table discovery capabilities.

## 2 TABLECOPILLOT AND KEY TECHNIQUES

### 2.1 System Overview

At a high level, TABLECOPILLOT functions as an LLM-based table assistant capable of invoking table discovery methods, as shown in Figure 1. Users submit a natural language request or perform

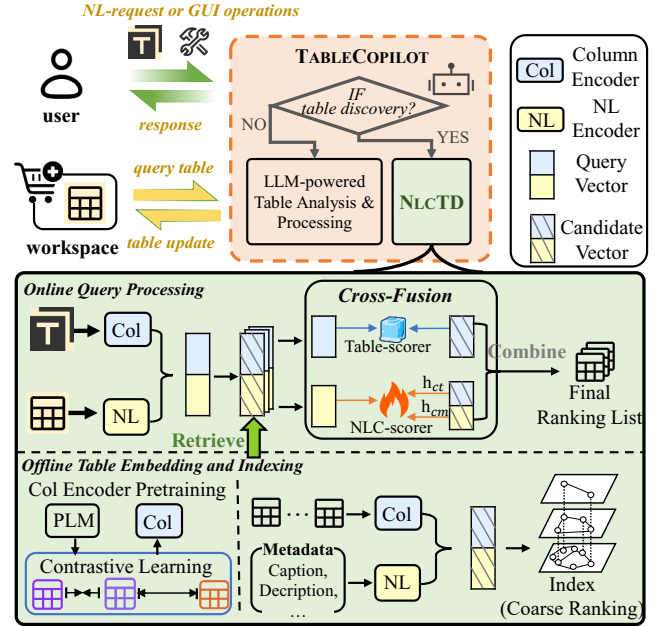


Figure 1: Overall Architecture of TABLECOPILLOT.

GUI actions, invoking the table assistant to respond and potentially update the workspace based on the user’s request and the current workspace status (e.g., an empty or incomplete table for search, and a complete table for analysis and manipulation).

**Table Assistant Pipeline.** The assistant performs two key tasks: (1) identifying whether the human instruction involves a table discovery request and (2) handling table analysis and processing tasks beyond discovery. For a table discovery request, the assistant redirects the NL condition and a potential query table at the workplace to the online processing module of CROFUMA. We employ an LLM, such as GPT-4o, to power the pipeline due to its ability to understand and process tables while following human instructions effectively. With carefully designed prompts, the LLM can accurately distinguish table discovery requests for appropriate redirection. Additionally, users can opt for alternative LLMs, including those fine-tuned specifically for table-related tasks [9].

**Processing NLCTD.** Given a table repository  $\mathcal{T}$ , and a user query  $q \in Q$  consisting of a query table  $T^q$  and an NL condition  $C$ , the NLCTD task aims to retrieve from  $\mathcal{T}$  a top- $k$  ranked list of tables  $\mathcal{T}' = \{T_i\}$  that are *semantically* relevant to both  $T^q$  and  $C$ , as determined by a relevance scoring function,  $\rho(T^q, C, T_i)$ . Our CROFUMA follows a two-stage approach: (1) **the offline phase** (Section 2.2) encodes and indexes the entire table repository  $\mathcal{T}$ , and (2) **the online phase** (Section 2.3) encodes the query and computes the similarity between  $q$  and candidate tables  $T_i$  for top- $k$  search.

### 2.2 Offline Data Preparation

**Column Encoder Pretraining.** We adopt a column-based representation, which applies to both union (aggregating column similarity scores) and join (targeting join key column) searches within NLCTD. We adopt contrastive learning, as used in Starmie [7], to train a PLM-based column encoder. This allows the model to better

capture contextual information, improving performance in cases where semantics are similar, but values differ. Notably, our system is compatible with other column representation methods [6], and users can integrate alternative column encoders.

**Table Repository Embedding.** To facilitate NLCTD processing, CROFUMA encodes both table content and metadata for all tables in the repository during the offline phase. For *table embedding*, we use the pretrained column encoder to derive the column embeddings and then concatenate them. For *metadata embedding*, we employ PLMs such as BERT or RoBERTa to encode textual information, including table captions and table descriptions. These embeddings are then used to match natural language conditions during the online phase. Finally, we concatenate the table embedding and the metadata embedding for subsequent indexing.

**Index Construction.** Constructing an index for coarse ranking is crucial when searching large table repositories, especially in our prototype, which contains both column and metadata embeddings. We choose to implement the Hierarchical Navigable Small World (HNSW) in CROFUMA due to its proven efficiency and lower loss rates [6, 7]. This index achieves an average query time of less than 500 ms for each in a repository containing 7,500+ tables.

### 2.3 Online Query Processing

We propose a cross-fusion-based matching approach, CROFUMA, with two key modules: a table scorer (for query-table matching) and an NLC scorer (for NL-condition matching). These modules compute scores that are aggregated to produce the final result. This design seamlessly supports scenarios where only one input (query table or NL condition) is available, covering all three types of NLCTD cases (Section 1). Before matching, the query table  $T^q$  and NL condition  $C$  are embedded using a method similar to offline embedding (Section 2.2), with column embeddings  $\mathbf{a}^i$  for  $T^q$  and the NL condition embedding  $\mathbf{c}$ . To enhance efficiency, a limited set of candidate tables is first retrieved from the offline-constructed HNSW index, followed by cross-fusion matching on these candidates.

**Table Scorer** matches the query table with candidate tables using a learning-free method that directly computes similarity. This approach is chosen as it operates within the same modality, leverages the pretrained table encoder, and minimizes online query time. CROFUMA supports both union and join scenarios, accounting for the diversity of NL conditions.

**NL-conditional Table Join Search:** For join scenarios, the focus is on matching values in the key column. We use the designated key column embedding  $\mathbf{a}^k$  as the table embedding  $\mathbf{t}^q$  and derive the table joinability score  $\rho^t$  using cosine similarity.

**NL-conditional Table Union Search:** For union scenarios requiring multiple overlapping columns, column similarity scores are aggregated. Inspired by previous works [3, 7], we construct a bipartite graph  $G = (V^q, V^T, E)$ , where nodes  $V^q$  and  $V^T$  represent the column sets of the query and candidate tables, and the edges  $E$  represent column similarity scores. The table unionability score  $\rho^t$  is computed by finding the maximum bipartite matching in  $G$ .

**NLC Scorer** matches the NL condition with candidate tables using a learning-based method for improved cross-modal matching. The NLC score is derived from two modules: condition-table matching and condition-metadata matching.

**Condition-Table Matching:** This module models the interaction between the NL condition and a candidate table. We concatenate the condition vector  $\mathbf{c}$  and candidate table vector  $\mathbf{t}_i$  along with their element-wise subtraction and Hadamard product<sup>1</sup>:

$$\hat{\mathbf{h}}_i = \text{concat}(\mathbf{t}_i, \mathbf{c}, (\mathbf{t}_i - \mathbf{c}), (\mathbf{t}_i \circ \mathbf{c})). \quad (1)$$

A nonlinear transformation is applied to produce hidden vectors:

$$\mathbf{h}_i = \text{Tanh}(\mathbf{W}_1 \hat{\mathbf{h}}_i + \mathbf{b}_1). \quad (2)$$

Finally, we compute the hidden vectors for all  $|\mathcal{T}|$  candidate tables using a max-pooling operation, which extracts the most relevant content for the NL condition [11]:

$$\mathbf{h}^{ct} = \text{MaxPooling}(\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_{|\mathcal{T}|}). \quad (3)$$

**Condition-Metadata Matching:** This module matches the NL condition with the table metadata (e.g., table captions), which often reflects the table’s content [3]. Treating this as a text matching task, we compute  $\mathbf{h}^{cm}$  using RoBERTa as the backbone, though models like T5 can also be used.

**Optimization of CROFUMA.** The NLC score  $\rho^c$  is obtained by concatenating  $\mathbf{h}_c = [\mathbf{h}^{ct}, \mathbf{h}^{cm}]$  and passing it through a multi-layer perceptron (MLP). The final score  $\rho_k$  for each table  $T_k$  is obtained by combining the table score  $\rho^t$  and condition score  $\rho^c$  with a weighting factor  $\lambda$ . This approach remains robust even if one of the scores is unavailable, ensuring CROFUMA’s compatibility with single-input table discovery. To compare  $\rho_k$  with the gold label  $y_k$ , we optimize the MLP using mean square error (MSE) loss, following previous table discovery studies [11]:

$$\arg \min_{\theta} \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|\mathcal{T}|} \sum_{k=1}^{|\mathcal{T}|} ((\text{MLP}(\mathbf{h}_c, \theta) + \lambda \cdot \rho^t) - y_k)^2. \quad (4)$$

### 2.4 Effectiveness of CROFUMA

We evaluate our CROFUMA on the NLCTABLES dataset [1]. We include six representative approaches published at top-tier venues, two each for keyword-based search (GTR [11] and StruBert [10]), table union search (Santos [8] and Starmie [7]), and table join search (Josie [13] and Deepjoin [6]). As shown in Table 1, our CROFUMA outperforms other single-input methods, demonstrating its superior ability to capture both query table and NL condition information, making it better suited for the new NLCTD scenario.

**Table 1: Comparisons of CROFUMA with SOTA methods.**

	nlc-Union			nlc-Join	
	NDCG@5	NDCG@5		NDCG@5	NDCG@5
Santos	0.2076	0.2912	Josie	0.4691	0.5170
Starmie	0.3066	0.4086	Deepjoin	0.2985	0.3959
GTR	0.3211	0.4459	GTR	0.5071	0.5571
StruBert	0.3675	0.3992	StruBert	0.5270	0.5291
<b>CROFUMA</b>	<b>0.4124</b>	<b>0.4825</b>	<b>CROFUMA</b>	<b>0.6674</b>	<b>0.6976</b>

## 3 DEMONSTRATION

**Artifacts and Demonstration Materials.** TABLECOPLOT is built on the Flask framework and comprises over **10,000 lines of backend and frontend code**. To encourage participation from developers and researchers, we provide an instructional video, the codebase,

<sup>1</sup>This concatenation method is proven effective to model embedding interactions [11].

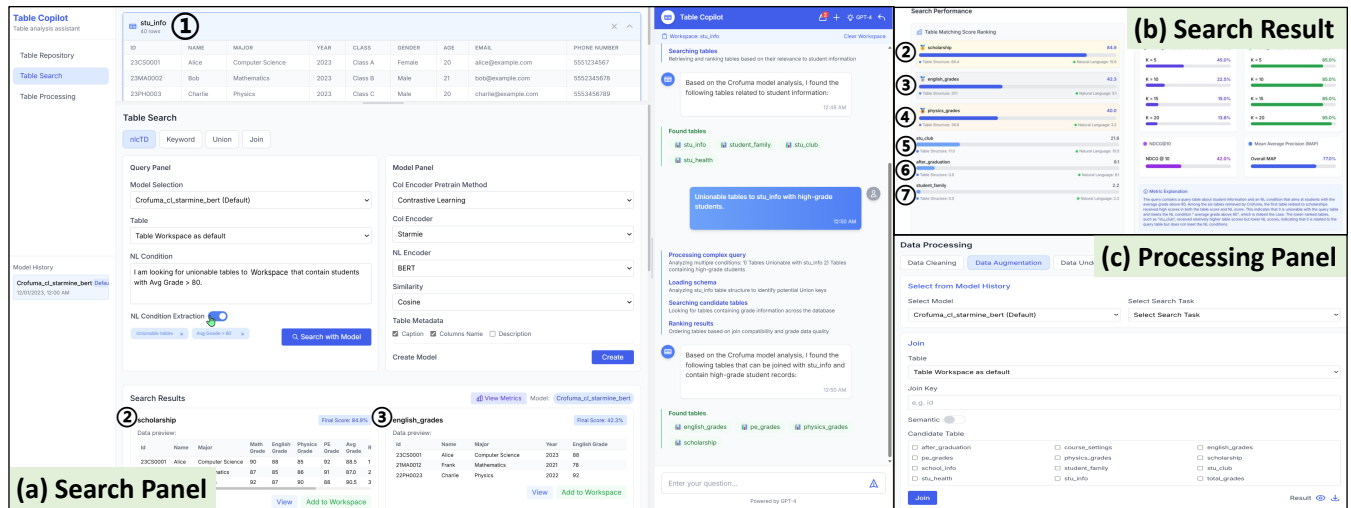


Figure 2: The Graphical User Interface of TABLECOPLOT. For a live demonstration, visit our project website on GitHub [2].

and additional resources on GitHub [2] to foster exploration of advanced table assistants capable of table discovery. For more details, please refer to our extended report [4].

**Runtime Case Study.** Figure 2 demonstrates a real application of CROFUMA to the NLCTD scenario, showcasing its ability to handle complex NL conditions. The query involves a query table about student information (Table ① in Figure 2(a)) and an NL condition that specifies: “Find unionable tables containing students with an average grade above 80.” **Search Panel** (Figure 2(a)): Users input the query table and define the NL condition via an intuitive interface. The search panel allows selection of algorithms, such as CROFUMA, with the model panel inside providing flexibility in configuring search parameters. **Search Results** (Figure 2(b)): Among the six retrieved tables (Table ② - ⑦ in Figure 2(b)), the top-1 table “scholarship” (Table ②) received high scores for both table score and NL score, confirming it is unionable with the query table and satisfies the NL condition. Lower-ranked tables, such as “english\_grades” (Table ③), scored high in table scores but failed to meet the NL condition, as reflected by their low NL scores. **Processing Panel** (Figure 2(c)): After identifying relevant tables, users can proceed with further data processing tasks like cleaning and augmentation. This panel enables seamless integration of discovery and processing workflows, ensuring a streamlined user experience.

**Key Benefits for the Audience.** TABLECOPLOT provides a distinctive platform to experience LLM-powered data management via an intuitive and engaging interface:

1. **Accessible and Flexible NLCTD Scenario.** Explore a highly user-friendly interface (Figure 2(a)) that supports flexible NL condition edits on themes, categories, and distributions. The LLM enables natural language interactions, making advanced table discovery accessible to users of all expertise levels.
2. **Deep Insights into CROFUMA.** Interact directly with CROFUMA through a dynamic model panel, comparing its performance against leading algorithms like Starmie. Visualizations of intermediate results (Figure 2(b)) demonstrate how CROFUMA

effectively handles complex queries involving both union and join tasks, outperforming traditional approaches.

3. **Integrated Table Discovery and Analysis.** Learn about seamless integration of LLMs in table discovery workflows. TABLECOPLOT automatically detects discovery needs, invokes optimal algorithms, and facilitates advanced table manipulation (Figure 2(c)), connecting AI advances with database research.

## ACKNOWLEDGMENTS

This work was supported by the Major Research Program of the Zhejiang Provincial Natural Science Foundation (Grant No. LD24F020015), the Pioneer R&D Program of Zhejiang (Grant No. 2024C01021), NSFC Grant No. U24A201401, and Zhejiang Province “Leading Talent of Technological Innovation Program” (No. 2023R5214).

## REFERENCES

- [1] 2025. NLCTABLES. <https://github.com/SuDis-ZJU/nlctables>
- [2] 2025. TABLECOPLOT Project. <https://sudis-zju.github.io/table-copilot/>
- [3] C Chai, Y Deng, Y Zhan, Z Cao, Y Zhang, L Cao, Z Wang, Yand Zhang, Y Yuan, G Wang, and N Tang. 2024. LakeCompass: An End-to-End System for Data Maintenance, Search and Analysis in Data Lakes. *PVLDB*, 4381–4384.
- [4] L Cui, G Jiang, H Li, K Chen, L Shou, and G Chen. 2025. TableCopilot: A Table Assistant Empowered by Natural Language Conditional Table Discovery. *ArXiv:2507.08283*.
- [5] L Cui, H Li, K Chen, L Shou, and G Chen. 2024. Tabular Data Augmentation for Machine Learning: Progress and Prospects of Embracing Generative AI. *ArXiv:2407.21523*.
- [6] Y Dong, C Xiao, T Nozawa, M Enomoto, and M Oyamada. 2023. DeepJoin:Joinable Table Discovery with Pre-Trained Language Models. *PVLDB*, 2458–2470.
- [7] G Fan, J Wang, Y Li, D Zhang, and R J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *PVLDB*, 1726–1739.
- [8] A Khatiwada, G Fan, R Shraga, Z Chen, W Gatterbauer, R. Miller, and M Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. *Proc. ACM Manag. Data*, 1–25.
- [9] P Li, Y He, D Yashar, W Cui, S Ge, H Zhang, D R Fainman, D Zhang, and S Chaudhuri. 2024. Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. *Proc. ACM Manag. Data*, 176.
- [10] M Trabelsi, Z Chen, S Zhang, B D. Davison, and J Heflin. 2022. StruBERT: Structure-aware BERT for Table Search and Matching. In *WWW*. 442–451.
- [11] F Wang, K Sun, M Chen, J Pujara, and P Szekely. 2021. Retrieving Complex Tables with Multi-Granular Graph Representation Learning. *SIGIR*, 1472–1482.
- [12] X Zhang, S Luo, B Zhang, Z Ma, J Zhang, G Li Y Li, Z Yao, K Xu, J Zhou, D Zhang-Li, J Yu, S Zhao, J Li, and J Tang. 2024. TABLELLM: Enabling Tabular Data Manipulation by LLMs in Real Office Usage Scenarios. *ArXiv:2403.19318*.
- [13] E Zhu, D Deng, F Nargesian, and R J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *Proc. ACM SIGMOD*. 847–864.