Demonstration of PI2: Interactive Visualization Interface Generation for SQL Analysis in Notebook

Jeffrey Tao
jeffrey.tao@columbia.edu
Columbia University
New York, NY, USA

Yiru Chen
yiru.chen@columbia.edu
Columbia University
New York, NY, USA

Eugene Wu
ewu@cs.columbia.edu
Columbia University
New York, NY, USA

ABSTRACT

We demonstrate PI2, the first notebook extension that can automatically generate interactive visualization interfaces during SQL-based analyses.

CCS CONCEPTS

• **Information systems** → **Structured Query Language**; • **Human-centered computing** → **Visualization systems and tools**; **User interface design**.

KEYWORDS

database usability, interface generation, data visualization, notebook analysis

ACM Reference Format:
Jeffrey Tao, Yiru Chen, and Eugene Wu. 2022. Demonstration of PI2: Interactive Visualization Interface Generation for SQL Analysis in Notebook. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3514221.3520153

1 INTRODUCTION

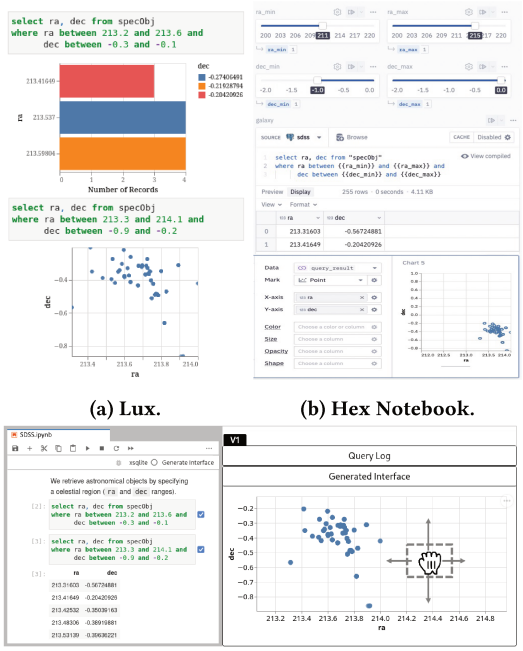
SQL is the dominant language for accessing and analyzing data. Along with the recent rise of notebooks, many popular computational notebooks for SQL have emerged such as xeus-sqlite [3] in Jupyter, SQL notebook [4], Hex [2], Count [1], etc. Data scientists have gradually ditched their traditional SQL IDEs, where they can only see one output at a time, in favor of computational notebooks so that they can enjoy narrative programming benefits [9].

Traditional SQL notebooks [3, 4] merely render query results as tables. These are of limited usefulness, as data scientists rely on interactive visualization interfaces to rapidly perform iterative analyses and to better present analyses in a compelling narrative [12].

In contrast to static tables, interactive visualization interfaces (or *interfaces*) consist of three main components: result visualizations (e.g., bar and line charts), query configuration widgets (e.g. dropdown, slider), and interactions within a visualization (e.g. brushing to select points, panning, clicking). Numerous recent notebooks and extensions, such as Lux [10], Count Notebook [1], and Hex

	Lux	*Count*	*Hex*	**PI2**
Visualizations	✓	✓	✓	✓
Widgets	×	*Parameter*	*Parameter*	**Arbitrary**
Vis. Interactions	×	×	×	✓
Zero Effort	✓	×	×	✓

Table 1: Comparison among different tools.



(a) Lux. **(b) Hex Notebook.**

(c) This paper: PI2.

Figure 1: Different interfaces for analysis of the SDSS dataset: (a) static visualization recommendation with Lux, (b) parameterized query with widgets and visualization with Hex, (c) automatically generated interactive interface with PI2.

Notebook [2] are designed to help users visualize data and create simple interactive visualizations during their analysis. Unfortunately, the type and complexity of interfaces that they can express are limited (Table 1). For instance, Lux [10] automatically recommends a static visualization for a dataframe, but does not support interactive analysis. Count [1] and Hex [2] let users visualize tables and add custom widgets that manipulate simple query parameters, but this requires explicit user effort. In short, existing notebooks have limited support for interactivity, do not support generating interactive visualizations, and require manual effort to create and lay out visualizations and widgets.

This paper demonstrates PI2, the first notebook extension which can automatically generate interactive visualization interfaces during SQL-based analyses. Users can select relevant queries during

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3520153

their analysis and invoke PI2 to synthesize a fully interactive interface without having to perform any manual specification. PI2 automatically chooses the appropriate visualizations, widgets, and visualization interactions to fully express the analysis represented by the user's selected queries.

Example 1. *Figure 1 illustrates the types of interfaces that Lux, Hex, and PI2 generate using queries from the Sloan Digital Sky Survey (SDSS) [11] query log. The example queries retrieve astronomical objects by specifying a celestial region (*`ra` *and* `dec` *ranges). Lux recommends visualizations per-query, so it generates different visualizations for each query, despite their similarities (Figure 1a). Users will need to repeatedly tweak and re-execute the queries to continue their analysis.*

Hex lets the user parameterize the `ra` *and* `dec` *values in the query, create custom sliders to control each parameter, and visualize the query results (Figure 1b). This enables more interactivity than Lux, since the user can drag the sliders instead of editing SQL strings. However, the user still needs to tell Hex to parameterize parts of the query on the slider values and choose an appropriate visualization. Furthermore, using the interface is cumbersome, as the user needs to manipulate four separate sliders to pan and zoom.*

In contrast, PI2 uses the same two queries to generate the interface in Figure 1c. The interface supports panning and zooming interactions, so the user can simply drag and scroll on the visualization to manipulate the `ra` *and* `dec` *ranges and receive immediate visual feedback. The collapsed Query Log tab archives the input queries.*

To enable PI2 to automatically generate suitable interfaces, we need to encode the systematic variation between queries in an analysis. To do so, we introduce a novel data structure called Difftrees that generalizes the abstract syntax trees (ASTs) of multiple queries and encodes structural differences between them at "choice nodes" in the tree. PI2 uses the Difftree schema and choice nodes to make appropriate choices for the three components of interactive visualization interfaces (visualizations, query configuration widgets, and visualization interactions). The key intuition behind Difftrees is that by fixing values at each choice node, a single query is produced, so by binding widgets and interactions to choice nodes, the user can transform the query expressed by the interface. Furthermore, this interface mapping process takes into account the available screen size—on a large screen, the interface may show multiple visualizations side by side, whereas a small screen may show a single visualization that can be changed via interactions. These techniques enable PI2 to go beyond existing notebooks by handling complex queries, expressing arbitrary query changes while ensuring that good interfaces are produced.

We further design the notebook extension to aid iterative analyses. Since notebook users often edit and re-run cells during analysis, we preserve a snapshot of the queries used to generate a new interface. In addition, we track interface versions, and let users see or revert back to previous interfaces. To avoid interruption of the normal notebook workflow, we choose to lay the *Generated Interface* panel side-by-side with the notebook cells.

The next section presents a brief overview of how PI2 generates interfaces from queries, and Section 3 illustrates these features in a case study. Please refer to our technical report[7] for more details.

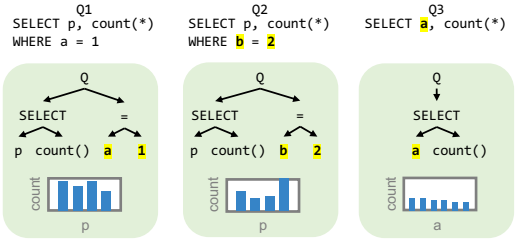


Figure 2: Example of three queries and their simplified ASTs. A static interface would render one chart for each query.

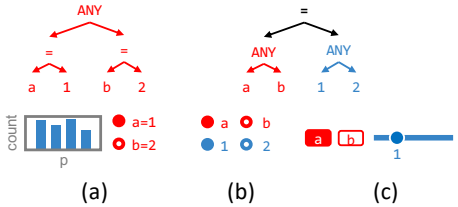


Figure 3: Example Difftrees and interfaces for `Q1`, `Q2`, focusing on the subtree for the predicate. The `ANY` choice node can choose one of its children. (a) the `ANY` node maps to a radio button that chooses between the two predicates, (b) the radio lists separately specify the left and right operands, (c) the choice nodes can instead be mapped to a button group and slider, and organized vertically.

2 INTERFACE GENERATION OVERVIEW

PI2 transforms an input sequence of queries into an interactive interface in four steps: it parses queries into a generalization of abstract syntax trees (ASTs) that we call Difftrees; maps the Difftrees to a candidate interface; estimates the interface's cost; and repeatedly transforms the Difftrees to generate new candidate interfaces, optimizing according to cost. This section walks through these steps and introduces key concepts via simple examples. For details on the interface generation process, please see the technical report [7].

Static Interfaces: Figure 2 lists three queries[1] and their ASTs, where attributes `p`, `a`, `b` are integers. `Q1` and `Q2` change the predicates, and `Q3` selects `a` instead of `p`. Since each AST is a Difftree, a valid interface simply maps each AST's results to a static chart.

Interactive Interfaces: Let us focus on the differing predicate in `Q1` and `Q2` to show how different Difftree structures can result in different interface designs. For instance, Figure 3(a) is rooted at an `ANY` node whose children are the two predicates. `ANY` is a *Choice Node* that can choose one of its child subtrees. In general, choice nodes encode subtree variations[2] that the user can control through the interface. In the example, the `ANY` node is mapped to two radio buttons (other widgets such as a dropdown are valid as well), where clicking on the first button would bind the `ANY` to its first child `a=1`. The Difftree output is visualized as a bar chart.

Tree Transformations: Both of `ANY`'s children are rooted at `=`, so the `=` can be refactored above the `ANY` node. This is an example of a *Tree Transformation Rule*. The resulting Difftree in Figure 3(b) shows two `ANY` nodes that can independently choose the left and right operands. This leads to an interface with two interactions—two

[1] For brevity, we omit the `FROM` and `GROUPBY` clauses and show simplified ASTs.
[2] Choice nodes generalize SQL parameterized literals to syntactic structures.

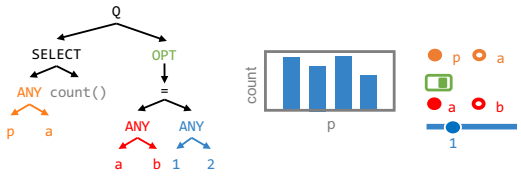


Figure 4: A DIFFTREE for Q1–3 and a candidate interface

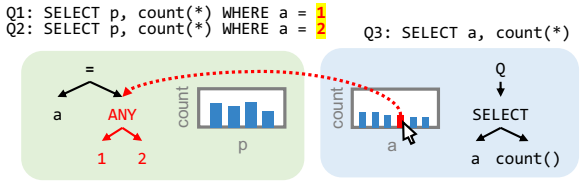


Figure 5: Multi-view interface where clicking on the right-side chart updates the left chart.

sets of radio buttons—and also generalizes the interface beyond the input queries. For instance, the query can now express `SELECT p, count(*) WHERE b=1`.

A DIFFTREE can map to many interface designs, each with different visualizations, interactions (including widgets and visualization interactions), and layouts. For instance, Figure 3(b) and (c) both express Q1 and Q2, however Figure 3(c) uses horizontal layout and the slider can choose a continuous range of numbers that generalize beyond the radio buttons in Figure 3(b).

All Three Queries: Now, let us add Q3. A simple approach would be to partition the queries into two clusters, where Q3 is rendered as a static chart, and Q1 and Q2 are mapped to one of the interfaces discussed so far. We can then choose to lay these two visualizations out horizontally or vertically. Another possibility is to merge all three queries into a single DIFFTREE, which would map to an interface with a single visualization. Figure 4 illustrates one such DIFFTREE structure, where an ANY node in the SELECT clause chooses whether to project p or a. This maps to an interface similar to Figure 3(c), but with a radio button to choose the attribute to project and a toggle for the optional WHERE clause. Naturally, which of these possible interface designs (or others not discussed here) that should be returned to the user depends on many factors, such as usability, layout, accessibility, and other factors that are difficult to quantify. Quantitative interface evaluation is an active area of research, and PI2 borrows current best practices to develop its cost function.

Multi-view Interfaces PI2 can also generate multi-view interfaces. Figure 5 illustrates a slightly different set of queries, where Q1 and Q2 only differ in the literal, and Q3 remains the same. Since the literal is compared to attribute a, an alternative to mapping the ANY node to a slider is to map it to a *visualization interaction* in Q3's bar chart. Specifically, each bar is derived from `(a, count(*))` in Q3's result. Thus, clicking on a bar can derive a valid value in attribute a's domain that can bind to the ANY node which will produce a new query for the left visualization. Then, the left visualization will update to render the new query's results.

Summary and Generation Pipeline PI2 transforms an input query log into an interactive interface in four steps (Figure 6). ① It first parses the input query sequence Q into DIFFTREES. ② PI2 maps the DIFFTREES to an interface. An interface mapping $\mathbb{I} = (\mathbb{V}, \mathbb{M}, \mathbb{L})$ is defined by a *Visualization Mapping* $\mathbb{V}$ from DIFFTREES results

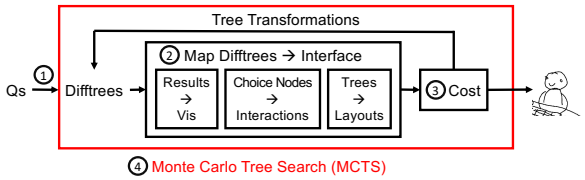


Figure 6: PI2 interface generation pipeline.

to visualizations, a *Interaction Mapping* $\mathbb{M}$ from choice nodes to interactions (including widgets and visualization interactions), and a *Layout Mapping* $\mathbb{L}$ from DIFFTREES structures to layouts. We formulate the interface mapping problem as a schema matching problem by defining schema for both DIFFTREES and interfaces. ③ A cost model $C(\mathbb{I}, Q)$ evaluates the interface and PI2 either returns the interface or chooses a valid transformation to apply to the DIFFTREES. Informally, our problem is to return the lowest cost interface $\mathbb{I}$ that can express all queries in Q. ④ The space of possible interfaces is enormous, so we solve this problem using Monte Carlo Tree Search [5, 6] (MCTS), which balances exploitation of good explored states (DIFFTREES) with exploration of new states.

3 DEMONSTRATION

3.1 Interface Design

We integrate PI2 as a Jupyter Lab [8] extension. We design the interface as shown in Figure 7. While authoring SQL queries in the Jupyter notebook, a user can check the checkbox next to each cell to include it as part of the query log for interface generation. Clicking the `Generate Interface` button invokes PI2 to generate a new interface, shown in the *Generated Interfaces* panel on the right.

The atomic unit of execution in notebooks allows users to easily refer back to previous cells to edit and potentially re-execute them. To adapt to edits and ensure reproducibility, our integration tracks interface versions in the version tabs at the top of the *Generated Interfaces* panel and archives the input query logs in the *Query Log* collapsible section for each version. PI2 lays out the interfaces and notebook cells side-by-side, so that the normal notebook analysis workflow will not be interrupted.

3.2 Use Case Walkthrough

We demonstrate how PI2 aids the data analysis process via a real-world scenario (shown in Figure 7): in late December 2021, an analyst named Jane at a news organization is analyzing a COVID-19 dataset of daily case counts per-state with the intent to give travel warning advice for the winter holiday season. For the sake of comparison, we show a static visualization recommendation below each cell, which is given by an existing system Lux [10]. More details on the full capabilities of PI2 are in the technical report [7].

Step 1: Overview and detailed look of the dataset. Seeking to get an overall view of the data, Jane writes Q1 and gets a line chart recommended by Lux showing total case count over time. Looking for a more detailed view, Jane restricts the date range in Q2 to look back over two preceding half-month periods to see more recent trends. Moreover, she would like to do this over different date ranges, which will result in many similar static visualizations and a lengthy notebook. With PI2, she can select these three queries via their corresponding checkboxes and automatically generate an interface. PI2 produces a unified interactive interface V1 consisting

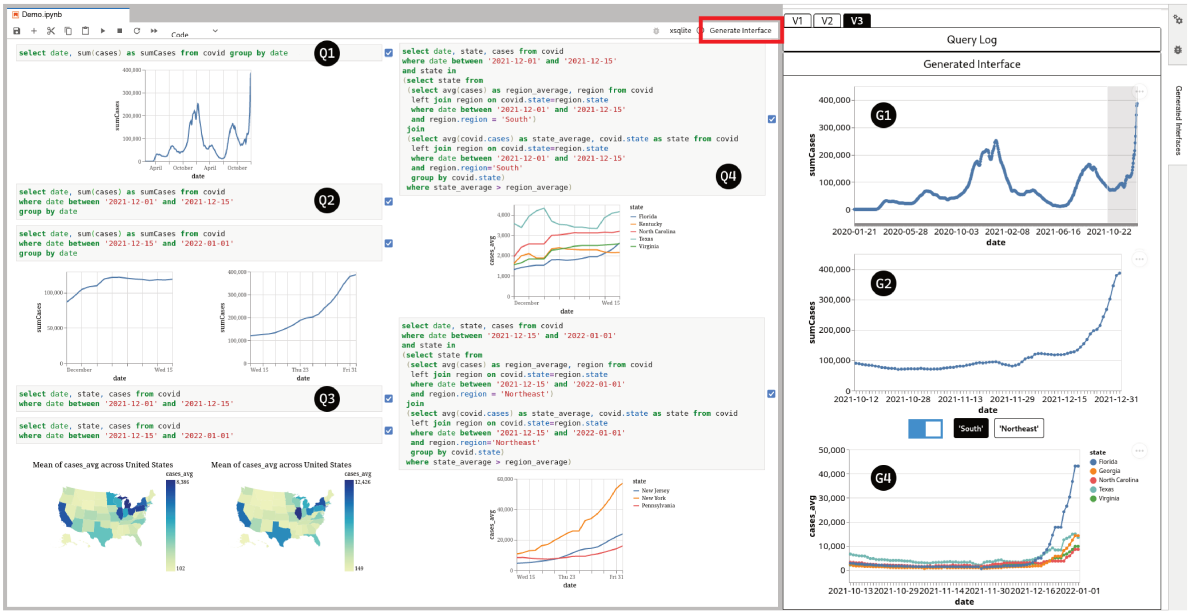


Figure 7: PI2 Jupyter extension interface. Left: SQL-based analyses. Right: An interface automatically generated by PI2.

of two plots: one showing the overall timeline (G1) and the other showing just the selected date range (G2). The two plots are linked by a brushing interaction so that brushing over G1 dynamically configures the date range of G2. In comparison, no existing notebook tools can create such visualization interactions. In this way, Jane can use this interface to further investigate trends in different date ranges interactively and without writing more SQL queries.

Step 2: Drill down into state Level. Jane's task is to give travel warning advice. She writes Q3 to study each state's trends over time. Lux generates choropleth maps visualizing the case count for each state averaged over the date window, which do not allow her to see trends over time. Alternatively, she can use PI2 to generate a new interface V2 with three plots: two are the sames as interface V1 and the third chart (G3) is a bar chart (x → date, y → cases, color → state). We omit G3 in Figure 7 due to space constraints. In this interface V2, the linked brushing interaction will configure the queries underlying both G2 and G3 such that Jane can brush over G1 to see the detailed trend and per-state breakdown trend within the selected date range at the same time.

Step 3: Focused region investigation. However, the state breakdown view is visually noisy due to the large number of states. Jane decides to group states into regions and only show those states whose average case counts over a period of time exceed the region's average, expressed as the complicated query Q4 consisting of joins, and correlative subqueries. Further, Jane would like to study the South and Northeast regions within different date ranges. She selects all the queries and invokes PI2. The interface V3 that PI2 generates in Figure 7 has three plots, allowing Jane to view the overall timeline (G1), detail view of the selected date range (G2), and state breakdown filtered for above average states (G4). This new interface maintains the date brushing functionality of previous versions and introduces query configuration widgets: a toggle which allows her to toggle between G3 and G4, and a pair of buttons that switch between the South and Northeast regions. Structurally, the toggle corresponds to an `OPT` choice node that distinguishes the existence of a complicated subquery—`{and state in...}` in Q4 not present in Q3. Through this interface, Jane is able to fluidly reconfigure the date range by brushing on G1, toggle off to see the overall state breakdown of cases, and toggle on to choose to observe trends in the Northeast or South. Noticing very high rates of growth in case count, Jane makes a recommendation that travelers avoid Florida in the South and New York in the Northeast.

Through the above scenario, we show PI2's ability to consume arbitrarily complex SQL queries and automatically generate complete interfaces, including visualization interactions and widgets expressing arbitrary query differences that no other tools can.

Demonstration engagement. Participants will be able to write their own analysis and generate interactive visual interfaces using the PI2 Jupyter extension. We will prepare three datasets – `COVID-19`, `SDSS`, and `S&P500` for users to explore.

Acknowledgements This work was supported in part by NSF 1845638, 2008295, 2106197, 2103794, Amazon, and Google.

REFERENCES

[1] 2021. Count – The SQL notebook. https://count.co/.
[2] 2021. Hex Technologies. https://hex.tech/.
[3] 2021. A Jupyter kernel for SQLite. https://github.com/jupyter-xeus/xeus-sql.
[4] 2021. SQL Notebook. https://sqlnotebook.com/.
[5] Cameron Browne, Edward Jack Powley, et al. 2012. A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games 4 (2012), 1–43.
[6] Yiru Chen and Eugene Wu. 2020. Monte Carlo Tree Search for Generating Interactive Data Analysis Interfaces. In AAAI Intelligent Process Automation Workshop.
[7] Yiru Chen and Eugene Wu. 2021. PI2: End-to-end Interactive Visualization Interface Generation from Queries. CoRR abs/2107.08203 (2021). arXiv:2107.08203
[8] B Granger and J Grout. 2016. JupyterLab: Building blocks for interactive computing. Slides of presentation made at SciPy (2016).
[9] Donald Ervin Knuth. 1984. Literate programming. The computer journal 27, 2 (1984), 97–111.
[10] Doris Jung-Lin Lee, Dixin Tang, Kunal Agarwal, et al. 20. Lux: Always-on Visualization Recommendations for Exploratory Data Science. VLDB (20).
[11] SDSS. 2021. Sloan Digital Sky Survey, 2017. http://www.sdss.org/.
[12] Edward Segel and Jeffrey Heer. 2010. Narrative visualization: Telling stories with data. IEEE transactions on visualization and computer graphics 16, 6 (2010).