APPENDIX

For better reading experience, we recommend the online version.

A.Gallery

Basic corpus

Overview

Overview

Aa class	Specification	Status
Row Oriented Unidimensional Table By 1to1 Functions without nestingS	(Paper), () ⇒ (Author)	Yes
Row Oriented Unidimensional Table By 1toN Functions without nesting	(First Name), () ⇒ (Last Name)	Yes
Row Oriented Unidimensional Table By Nto1 Functions without nesting	(product), () ⇒ (total_sales)	Yes
Row Oriented Unidimensional Table By 1to1 Functions with nesting	(Name), () ⇒ (total)	Yes
Row Oriented Unidimensional Table By 1toN Functions with nesting	(First Name), () ⇒ (Last Name)	Yes
Row Oriented Unidimensional Table By Nto1 Functions with nesting	(product), () ⇒ (total_sales)	Yes
Row Oriented Multidimensional Table By 1to1 Functions without nesting	(Year * State), () ⇒ (Avg_GDP)	Yes
Row Oriented Multidimensional Table By 1toN Functions without nesting	(Year * Month * State), () ⇒ (GDP)	Yes

Aa class	Specification	Status
Row Oriented Multidimensional Table By Nto1 Functions without nesting	(Year * State), () ⇒ (Crime_rate + Key)	Yes
Row Oriented Multidimensional Table By 1to1 Functions with nesting	(Year * State), () ⇒ (Avg_GDP)	Yes
Row Oriented Multidimensional Table By 1toN Functions with nesting	(Year * Month * State), () ⇒ (GDP)	Yes
Row Oriented Multidimensional Table By Nto1 Functions with nesting	(Year * State), () ⇒ (Crime_rate + Key)	Yes
All column oriented table	transposition operation	Yes
Cross-Tabulation Unidimensional Table by 1to1 Functions without nesting	(Semester), (Course) ⇒ (Amount)(Time), (State) ⇒ (Key)	Yes
Cross-Tabulation Unidimensional Table by 1toN Functions without nesting	(State), (Year) ⇒ (GDP)	Yes
Cross-Tabulation Unidimensional Table by Nto1 Functions without nesting	(Time), (State) ⇒ (Key)	Yes
Cross-Tabulation Unidimensional Table by 1to1 Functions with nesting	(Month), (Year) ⇒ (Avg_GDP)	Yes
Cross-Tabulation Unidimensional Table by 1toN Functions with nesting	(Month), (Year) ⇒ (Avg_GDP)	Yes
Cross-Tabulation Unidimensional Table by Nto1 Functions with nesting	(Year), (State) ⇒ (Key)	Yes
Cross-Tabulation Multidimensional Table by 1to1 Functions without nesting	(Course * Course_type), (Semester) ⇒ (Avg_score)	Yes
Cross-Tabulation Multidimensional Table by 1toN Functions without nesting	(Year * Month), (State) ⇒ (GDP)	Yes
Cross-Tabulation Multidimensional Table by Nto1 Functions without nesting	(Product * Country), (Date) ⇒ (total_sales)	Yes
Cross-Tabulation Multidimensional Table by 1to1 Functions with nesting	(Course * Course_type), (Semester) ⇒ (count_score_above_70)	Yes
Cross-Tabulation Multidimensional Table by 1toN Functions with nesting	(Year * Month), (State) ⇒ (GDP)	Yes
Cross-Tabulation Multidimensional Table by Nto1 Functions with nesting	(Product * Country), (Date) ⇒ (total_sales)	Yes

Row oriented Table

- **Generate a Row Oriented Unidimensional Table By 1to1 Functions without nesting**
  - Input Table

Paper	Author
D3 data-driven documents	Michael Bostock
D3 data-driven documents	Vadim Ogievetsky
D3 data-driven documents	Jeffrey Heer
Lyra: An interactive visualization design environment	Arvind Satyanarayan
Lyra: An interactive visualization design environment	Jeffrey Heer
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Arvind Satyanarayan
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Ryan Russell
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Jane Hoffswell
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Jeffrey Heer

Vega-lite: A grammar of interactive graphics	Arvind Satyanarayan
Vega-lite: A grammar of interactive graphics	Dominik Moritz
Vega-lite: A grammar of interactive graphics	Kanit Wongsuphasawat
Vega-lite: A grammar of interactive graphics	Jeffrey Heer

- Instructions

  Count the amount of authors of each paper.

- Output Table

Paper	Count_authors_per_paper
D3 data-driven documents	3
Lyra: An interactive visualization design environment	2
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	4
Vega-lite: A grammar of interactive graphics	4

- **Generate a Row Oriented Unidimensional Table by 1toN Functions without nesting**
  - Input Table

Name
John Smith
Peter Wallen
Kevin Micheal
Robin Anderson

  - Instructions

    Split the name into First Name and Last Name.

  - Output Table

First Name	Last Name
John	Smith
Peter	Wallen
Kevin	Micheal
Robin	Anderson

- **Generate a Row Oriented Unidimensional Table Do Nto1 Functions without nesting**
  - Input Data

```
Date: 2023-01-01
Order_ID: 10001
Order: 14 apples
Unit_Price: 7
Order_ID: 10002
Order: 10 bananas
Unit_Price: 5

Date: 2023-01-02
Order_ID: 10003
Order: 3 apples
Unit_Price: 8
Order_ID: 10004
Order: 12 bananas
Unit_Price: 4
```

  - Instruction

Calculate the total_sales by Product.

- Output Table

	total_sales
apple	122
banana	98

- **Generate a Row Oriented Unidimensional Table by 1to1 Functions with nesting**
  - Input Table

Name	Course	Score
John	English	93
John	Mathematics	78
John	Physics	84
Kevin	English	71
Kevin	Mathematics	69
Kevin	Physics	60
Peter	English	87
Peter	Mathematics	100
Peter	Physics	95

  - Instructions

    Get the total score of each student, and sort from highest to lowest.

  - Output Table

Name	Total
John	255
Kevin	200
Peter	282

  - Refined Instruction

    (click the Total) sort by descending order.

  - Output Table

Name	Total
Peter	282
John	255
Kevin	200

- **Generate a Row Oriented Unidimensional Table by 1toN Functions with nesting**
  - Input Table

Name
John Smith
Peter Wallen
Kevin Michael
Robin Anderson

  - Instruction

    Split the name into first name and last name and then sort in dictionary order.

- Output Table

First Name	Last Name
John	Smith
Peter	Wallen
Kevin	Michael
Robin	Anderson

- Refined Instruction

  (click “Last Name”) Sort in dictionary order.

- Output Table

	last_name_sorted
Robin	Anderson
Kevin	Michael
John	Smith
Peter	Wallen

- **Generate a Row Oriented Unidimensional Table by Nto1 Functions with nesting**
  - Input Data

```
Date: 2023-01-01
Order_ID: 10001
Order: 14 apples
Unit_Price: 7
Order_ID: 10002
Order: 10 bananas
Unit_Price: 5
Order_ID:10003
Order: 8 oranges
Unit_Price: 6

Date: 2023-01-02
Order_ID: 10004
Order: 3 apples
Unit_Price: 8
Order_ID: 10005
Order: 12 bananas
Unit_Price: 4
Order_ID:10006
Order: 13 oranges
Unit_Price: 5
```

  - Instruction

    Calculate the total sales of each fruit.

  - Output Table

	total_sales
apple	122
banana	98
orange	111

  - Refined Instruction

    (click the total_sales) Sort in descending order.

  - Output Table

	total_sales

apple	122
orange	111
banana	98

- **Generate a Row Oriented Multidimensional Table by 1to1 Functions without nesting**
  - Input Table

Year	Month	State	GDP
2022	July	Alabama	3201.5
2022	January	Alabama	3177.7
2021	July	Alabama	3101.5
2021	January	Alabama	3055.3
2022	July	Alaska	2799.1
2022	January	Alaska	2745.5
2021	July	Alaska	2665.9
2021	January	Alaska	2587.1

  - Instructions

    Calculate the average GDP by Year and State.

  - Output Table

		Avg_GDP_by_Year_and_State
2021	Alabama	3078.4
2021	Alaska	2626.5
2022	Alabama	3189.6
2022	Alaska	2772.3

- **Generate a Row Oriented Multidimensional Table by 1toN Functions without nesting**
  - Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

  - Instruction

    Show GDP based on Year, Month and State.

  - Output Table

			GDP
2021	January	Alabama	3055.3
2021	January	Alaska	2587.1
2021	July	Alabama	3101.5
2021	July	Alaska	2665.9

2022	January	Alabama	3177.7
2022	January	Alaska	2745.5
2022	July	Alabama	3201.5
2022	July	Alaska	2799.1

- **Generate a Row Oriented Multidimensional Table by Nto1 Functions without nesting**
  - Input Table

State	Year	Crime rate
Alabama	2004	4029.3
Alabama	2005	3900
District of Columbia	2004	4852.8
District of Columbia	2005	4490
California	2004	3423.9
California	2005	3321
New Jersey	2004	2433
New Jersey	2005	2337

  - Instructions

    add a key for each record.

  - Output Table

State	Year	Crime rate	Key
Alabama	2004	4029.3	Alabama2004
Alabama	2005	3900	Alabama2005
District of Columbia	2004	4852.8	District of Columbia2004
District of Columbia	2005	4490	District of Columbia2005
California	2004	3423.9	California2004
California	2005	3321	California2005
New Jersey	2004	2433	New Jersey2004
New Jersey	2005	2337	New Jersey2005

- **Generate a Row Oriented Multidimensional Table by 1to1 Functions with nesting**
  - Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

  - Instruction

    Get the average GDP based on Year and State.

  - Output Table

		Avg_GDP_by_Year_and_State
2021	Alabama	3078.4
2021	Alaska	2626.5
2022	Alabama	3189.6
2022	Alaska	2772.3

- **Generate a Row Oriented Multidimensional Table by 1toN Functions with nesting**
  - Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

  - Instruction

    Show GDP based on Year, Month and State.

  - Output Table

			GDP
2021	January	Alabama	3055.3
2021	July	Alabama	3101.5
2022	January	Alabama	3177.7
2022	July	Alabama	3201.5
2021	January	Alaska	2587.1
2021	July	Alaska	2665.9
2022	January	Alaska	2745.5
2022	July	Alaska	2799.1

  - Refined Instruction

    Sort by descending order according to the GDP.

  - Output Table

			GDP
2022	July	Alabama	3201.5
2022	January	Alabama	3177.7
2021	July	Alabama	3101.5
2021	January	Alabama	3055.3
2022	July	Alaska	2799.1
2022	January	Alaska	2745.5
2021	July	Alaska	2665.9
2021	January	Alaska	2587.1

- **Generate a Row Oriented Multidimensional Table by Nto1 Functions with nesting**

- Input table

State	Year	Crime rate
Alabama	2004	4029.3
Alabama	2005	3900
District of Columbia	2004	4852.8
District of Columbia	2005	4490
California	2004	3423.9
California	2005	3321
New Jersey	2004	2433
New Jersey	2005	2337

- Instructions

  add a key for each record.

- Output Table

		Crime rate	Key
Alabama	2004	4029.3	Alabama2004
Alabama	2005	3900	Alabama2005
District of Columbia	2004	4852.8	District of Columbia2004
District of Columbia	2005	4490	District of Columbia2005
California	2004	3423.9	California2004
California	2005	3321	California2005
New Jersey	2004	2433	New Jersey2004
New Jersey	2005	2337	New Jersey2005

- Refined Instruction

  (choose the Crime_rate) sort in descending order.

- Output Table

		Crime rate	Key
District of Columbia	2004	4852.8	District of Columbia2004
District of Columbia	2005	4490	District of Columbia2005
Alabama	2004	4029.3	Alabama2004
Alabama	2005	3900	Alabama2005
California	2004	3423.9	California2004
California	2005	3321	California2005
New Jersey	2004	2433	New Jersey2004
New Jersey	2005	2337	New Jersey2005

Column Oriented Table

Since the column-oriented table is symmetric to the row-oriented table, we only need to prove that NL2Rigel is able to do the transposition operation. Here is a example.

- **Generate a Column Oriented Unidimensional Table by Transposing the Row-Oriented Table**
  - Input Table

Name
John Smith

Peter Wallen
Kevin Micheal
Robin Anderson

- Instructions

  Split the name into First Name and Last Name.

- Output Table

First Name	Last Name
John	Smith
Peter	Wallen
Kevin	Micheal
Robin	Anderson

- Refined Instruction

  Transpose the table.

- Output Table

First Name	John	Peter	Kevin	Robin
Last Name	Smith	Wallen	Micheal	Anderson

Cross-Tabulation Table

- **Generate a Cross-Tabulation Unidimensional Table by 1to1 Functions without nesting**
  - Input Table

Semester	Course	Student_ID	Score
Spring	CS101	12345	90
Spring	CS101	67890	85
Spring	CS101	23456	60
Spring	CS102	12345	95
Spring	CS102	34567	69
Spring	CS102	56789	69
Autumn	CS101	54321	80
Autumn	CS101	76543	75
Autumn	CS101	11451	98
Autumn	CS102	54321	88
Autumn	CS102	76543	65
Autumn	CS102	11451	73

  - Instruction

    Count the amount of students based on Semester and Course.

  - Output Table

	CS101	CS102
Spring	3	3
Autumn	3	3

- **Cross-Tabulation Unidimensional Table Do 1toN Functions without nesting**
  - Input Table

Info	GDP
Alabama2021	3055.3
Alabama2022	3177.7
Alaska2021	2587.1
Alaska2022	2745.5

- Instruction

  Show GDP based on State and Year.

- Output Table

	2021	2022
Alabama	3055.3	3177.7
Year	2587.1	2745.5

- **Generate a Cross-Tabulation Unidimensional Table by Nto1 Functions without nesting**
  - Input Table

Date	Product	Quantity	Unit Price
2023-01-01	Apple	5	7
2023-01-01	Banana	6	4
2023-01-02	Apple	4	6.5
2023-01-02	Banana	5	4

  - Instruction

    Caulate the total sales grouped by Date and Product.

  - Output Table

	Apple	Banana
2023-01-01	35	24
2023-01-02	26	20

- **Generate a Cross-Tabulation Unidimensional Table Do 1to1 Functions with nesting**
  - Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

  - Instructions

    Calculate the average GDP of states based on Year and Month.

  - Output Table

	2021	2022
January	2821.2	2961.6

July	2883.7	3000.3

- **Generate a Cross-Tabulation Unidimensional Table by 1toN Functions with nesting**

  This case is same as the “Generate a Cross-Tabulation Unidimensional Table by 1to1 Functions with nesting”, so it will not be repeated.

- **Generate a Cross-Tabulation Unidimensional Table by Nto1 Functions with nesting**
  - Input Table

Time	State
January 2021	Alabama
July 2021	Alabama
January 2022	Alabama
July 2022	Alabama
January 2021	Alaska
July 2021	Alaska
January 2022	Alaska
July 2022	Alaska

  - Instruction

    Combine the State with Year as the Key and Show the Key based on Year and State.

  - Output Table

	Alabama	Alaska
2021	Alabama2021	Alaska2021
2022	Alabama2022	Alaska2022

- **Generate a Cross-Tabulation Multidimensional Table by 1to1 Functions without nesting**
  - Input Table

Semester	Course	Course_Type	Student_ID	Score
Spring	CS101	Honor	12345	90
Spring	CS101	Medium	67890	85
Spring	CS101	Medium	23456	81
Spring	CS101	Honor	34567	58
Spring	CS102	Honor	12345	85
Spring	CS102	Honor	34567	69
Spring	CS102	Medium	56789	82
Spring	CS102	Medium	67890	88
Autumn	CS101	Honor	54321	78
Autumn	CS101	Honor	76543	99
Autumn	CS101	Medium	11451	98
Autumn	CS101	Medium	23456	88
Autumn	CS102	Medium	54321	88
Autumn	CS102	Medium	76543	65
Autumn	CS102	Honor	11451	73

  - Instruction

    Calculate the average scores, group by semester and course and course_type.

- Output Table

		Spring	Autumn
CS101	Honor	74	88.5
CS101	Medium	83	94
CS102	Honor	77.5	73
CS102	Medium	85	76.5

- **Generate a Cross-Tabulation Multidimensional Table by 1toN Functions without nesting**
  - Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

  - Instruction

    Show GDP based on Year, Month and State.

  - Output Table

		Alabama	Alaska
2021	January	3055.3	2587.1
2021	July	3101.5	2665.9
2022	January	3177.7	2745.5
2022	July	3201.5	2799.1

- **Generate Cross-Tabulation Multidimensional Table by Nto1 Functions without nesting**
  - Input Table

Date	order	product	country	quantity	unit_price
2023/1/1	101	apple	USA	10	3
2023/1/1	103	banana	USA	2	4
2023/1/1	104	apple	CHN	5	3
2023/1/1	106	banana	CHN	3	4
2023/1/2	201	apple	CHN	14	4
2023/1/2	203	banana	USA	7	4
2023/1/2	204	apple	USA	8	4
2023/1/2	206	banana	CHN	5	4

  - Instruction

    calculate the total sales grouped by Date, country and product.

  - Output Table

		2023/1/1	2023/1/2

Apple	CHN	15	56
Apple	USA	30	32
Banana	CHN	12	20
Banana	USA	8	28

- **Generate a Cross-Tabulation Multidimensional Table by 1to1 Functions with nesting**
  - Input Table

Semester	Course	Course_Type	Student_ID	Score
Spring	CS101	Honor	12345	90
Spring	CS101	Medium	67890	85
Spring	CS101	Medium	23456	81
Spring	CS101	Honor	34567	58
Spring	CS102	Honor	12345	85
Spring	CS102	Honor	34567	69
Spring	CS102	Medium	56789	82
Spring	CS102	Medium	67890	88
Autumn	CS101	Honor	54321	78
Autumn	CS101	Honor	76543	99
Autumn	CS101	Medium	11451	98
Autumn	CS101	Medium	23456	88
Autumn	CS102	Medium	54321	88
Autumn	CS102	Medium	76543	65
Autumn	CS102	Honor	11451	73

  - Instruction

    Count the number of scores greater than 70, group by semester, course and course type.

  - Output Table

		Spring	Autumn
CS101	Honor	1	1
CS101	Medium	2	2
CS102	Honor	1	0
CS102	Medium	2	1

- **Generate a Cross-Tabulation Multidimensional Table by 1toN Functions with nesting**
  - Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

  - Instruction

Show the GDP by Year, Month and State.

- Output Table

		Alabama	Alaska
2021	January	3055.3	2587.1
2021	July	3101.5	2665.9
2022	January	3177.7	2745.5
2022	July	3201.5	2799.1

- Refined Instruction

  Only keeps the GDP of 2022.

- Output Table

		Alabama	Alaska
2022	January	3177.7	2745.5
2022	July	3201.5	2799.1

- **Generate a Cross-Tabulation Multidimensional Table by Nto1 Functions with nesting**
  - Input Table

Date	order	product	country	quantity	unit_price
2023/1/1	101	apple	USA	10	3
2023/1/1	103	banana	USA	2	4
2023/1/1	104	apple	CHN	5	3
2023/1/1	106	banana	CHN	3	4
2023/1/2	201	apple	CHN	14	4
2023/1/2	203	banana	USA	7	4
2023/1/2	204	apple	USA	8	4
2023/1/2	206	banana	CHN	5	4

  - Instruction

    Calculate the total sales of each product grouped by Date and country.

  - Output Table

		2023/1/1	2023/1/2
Apple	CHN	15	56
Apple	USA	30	32
Banana	CHN	12	20
Banana	USA	8	28

  - Refined Instruction

    Only Keep the records including USA.

  - Output Table

		2023/1/1	2023/1/2
Apple	USA	30	32
Banana	USA	8	28

Advanced corpus

We have designed an advanced corpus with 2 cases from real-world scenarios to demonstrate the generalizability of NL2Rigel in various contexts. Naturally, NL2Rigel can be potentially implemented as a web plug-in to construct tables from webpages by using the HTML document or raw text of the webpage as the input data. Therefore, case 1 illustrates an example of constructing tables from a book review website, which we have discussed in the main text. Additionally, since NL2Rigel is able to facilitate table generation and improvement with only a few efforts, it is also potentially extensible to open-ended exploration on big-scale datasets and data presentation. We depict how Bob, an average data analyst generates series of tables from an insurance dataset (570 records × 8 entities) with NL2Rigel in a few minutes for his presentation slides.

The included entities for the dataset are *id*, *age*, *sex*, *bmi*, *children*, *smoker*, *region* and *charges*. A sample of the dataset is shown in the following table.

id	age	sex	bmi	children	smoker	region	charges
1	19	female	27.90	0	yes	southwest	16884.92
2	18	male	33.77	1	no	southwest	1725.55
3	33	male	22.70	0	no	northwest	21984.47
4	37	female	27.74	3	no	northwest	7281.51
	……	……	……	……	……	……	……

Initially, Bob imports the data to NL2Rigel and provides the instruction "**Compute the average charges by sex**". NL2Rigel then derives the target table in seconds.

Table A

	avg_charges
female	12332.66
male	14210.77

Obviously, the table shows that males tend to pay more charges than females. Bob then decides that the theme of the table gallery he is going to make is "why males need more insurance". To dig into this question, he randomly chooses two interested entities (*smoker* and *children*) and plans to build a cross-tab, so he refines the table by the instruction "**Keep the males and show whether the charges relate to smoking and the children number**" and gets the following table.

Table B

	yes	yes	yes	yes	yes	yes	no
	0	1	2	3	4	5	0
male	33093.33	32525.31	32790.62	34444.66	17942.11		7672.44

After inspecting the table, Bob has two insights. First, apparently the smokers have far higher charges than the non-smokers. Second, the children seems to have little influence of the charges. He decides to generate a table showing clearly the relationship between *sex* and *smoker*. So he refines the table by "**remove the children**":

Table C

	yes	no
male	32897.86	8152.40

Next, Bob wants some specific examples to emphasize his points. So he says "Show the top males smokers", which leads to the following table:

Table D

		charges
male	yes	39611.7577, 36837.467, …

Although this table meets his demand to some extent, he doesn't want the charges to be put in a single cell. So he says "show the ID also" and selects a table from the recommended results:

Table E

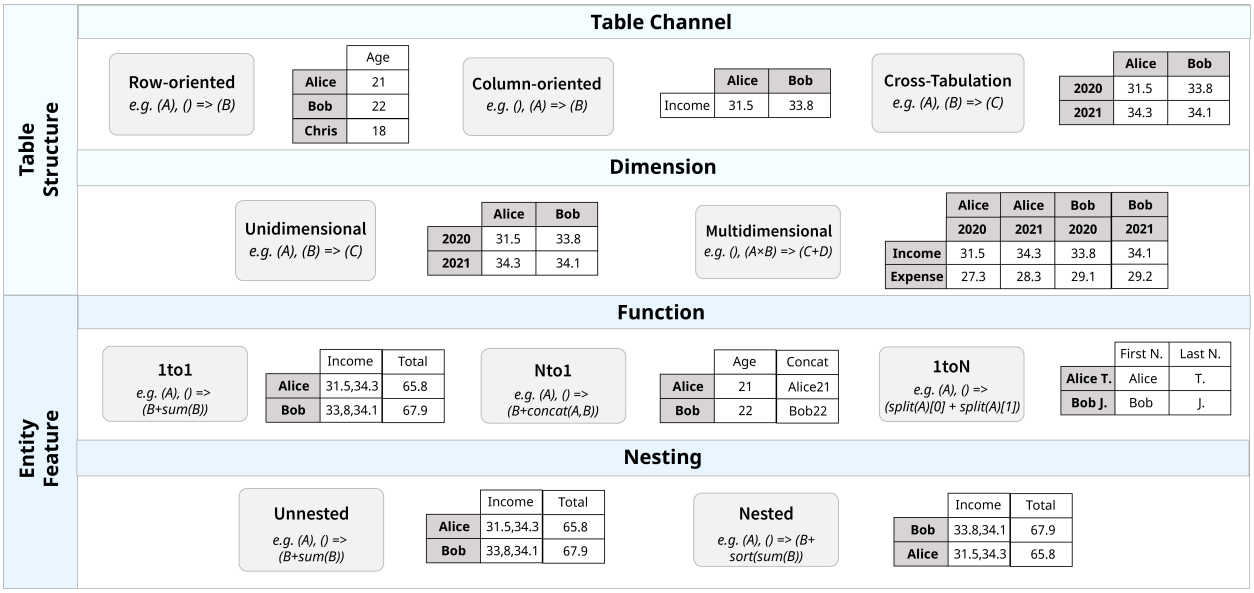
			charges
35	male	yes	51194.56
531	male	yes	48675.52
282	male	yes	48549.18
……	……	……	……

Throughout the process, Bob efficiently explores the dataset and easily generates three tables (A, C, E) for his presentation with the assistance of NL2Rigel.

B. The design space on tabular data transformation

An overview of the design space can be found in section 5 of the main contents. We provide detailed definitions of the classes and illustrate examples in this section.

- Row-oriented: Tables with an empty column header.
- Column-oriented: Tables with an empty row header.
- Cross-Tabulation: Tables that have both a non-empty row header and column header.
- Unidimensional: Tables with at most one entity mapped to a single table channel (row/column/cell).
- Multidimensional: Tables with at least two entities mapped to a single table channel (row/column/cell).
- 1to1: Entities with functions that take one parameter as input and output one entity.
- Nto1: Entities with functions that take multiple parameters as input and output one entity.
- 1toN: Entities with functions that take one parameter as input and can potentially output multiple entities.
- Unnested: Entities with at most a single data function exerted.
- Nested: Entities with at least two data functions exerted.



C. Tasks for User Study

Task 1

- Input Table

Paper	Author
D3 data-driven documents	Michael Bostock
D3 data-driven documents	Vadim Ogievetsky
D3 data-driven documents	Jeffrey Heer
Lyra: An interactive visualization design environment	Arvind Satyanarayan
Lyra: An interactive visualization design environment	Jeffrey Heer
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Arvind Satyanarayan
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Ryan Russell
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Jane Hoffswell
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Jeffrey Heer
Vega-lite: A grammar of interactive graphics	Arvind Satyanarayan
Vega-lite: A grammar of interactive graphics	Dominik Moritz
Vega-lite: A grammar of interactive graphics	Kanit Wongsuphasawat
Vega-lite: A grammar of interactive graphics	Jeffrey Heer

- Output Table

Paper	Author
D3 data-driven documents	Michael Bostock, Vadim Ogievetsky, Jeffrey Heer,
Lyra: An interactive visualization design environment	Arvind Satyanarayan,J effrey Heer,,
Reactive vega: A streaming dataflow architecture for declarative interactive visualization	Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, Jeffrey Heer
Vega-lite: A grammar of interactive graphics	Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer

Task 2

- Input Table

State	Year	Crime rate
Alabama	2004	4029.3
Alabama	2005	3900
Alabama	2006	3937
Alabama	2007	3974.9
Alabama	2008	4081.9

District of Columbia	2004	4852.8
District of Columbia	2005	4490
District of Columbia	2006	4653.9
District of Columbia	2007	4916.3
District of Columbia	2008	5104.6
California	2004	3423.9
California	2005	3321
California	2006	3175.2
California	2007	3032.6
California	2008	2940.3
New Jersey	2004	2433
New Jersey	2005	2337
New Jersey	2006	2278.4
New Jersey	2007	2205.5
New Jersey	2008	2293.4

- Output Table

State	Year	Crime rate	Key
Alabama	2004	4029.3	Alabama2004
Alabama	2005	3900	Alabama2005
Alabama	2006	3937	Alabama2006
Alabama	2007	3974.9	Alabama2007
Alabama	2008	4081.9	Alabama2008
District of Columbia	2004	4852.8	District of Columbia2004
District of Columbia	2005	4490	District of Columbia2005
District of Columbia	2006	4653.9	District of Columbia2006
District of Columbia	2007	4916.3	District of Columbia2007
District of Columbia	2008	5104.6	District of Columbia2008
California	2004	3423.9	California2004
California	2005	3321	California2005
California	2006	3175.2	California2006
California	2007	3032.6	California2007
California	2008	2940.3	California2008
New Jersey	2004	2433	New Jersey2004
New Jersey	2005	2337	New Jersey2005
New Jersey	2006	2278.4	New Jersey2006
New Jersey	2007	2205.5	New Jersey2007
New Jersey	2008	2293.4	New Jersey2008

Task 3

- Input Table

Name	Type	Number	Usage
Niles C.	Tel	(800)645-8397	home
Niles C.	Tel	(800)645-8398	work

Niles C.	Fax	(907)586-7252	work
Jean H.	Tel	(918)781-4600	home
Jean H.	Tel	(918)781-4601	work
Jean H.	Fax	(918)781-4603	home
Jean H.	Fax	(918)781-4604	work
Bach J.		781-4605	work
Bach J.		(918)781-4604	work

- Output Table

		Niles C.	Jean H.
Tel	home	(800)645-8397	(918)781-4600
Tel	work	(800)645-8398	(918)781-4601
Fax	home		(918)781-4603
Fax	work	(907)586-7252	(918)781-4604

Task 4

- Input Table

State	Year	Crime rate
Alabama	2004	4029.3
Alabama	2005	3900
Alabama	2006	3937
Alabama	2007	3974.9
Alabama	2008	4081.9
District of Columbia	2004	4852.8
District of Columbia	2005	4490
District of Columbia	2006	4653.9
District of Columbia	2007	4916.3
District of Columbia	2008	5104.6
California	2004	3423.9
California	2005	3321
California	2006	3175.2
California	2007	3032.6
California	2008	2940.3
New Jersey	2004	2433
New Jersey	2005	2337
New Jersey	2006	2278.4
New Jersey	2007	2205.5
New Jersey	2008	2293.4

- Output Table

Alabama	Alabama	Alabama	Alabama	Alabama	District of Columbia	District of Columbia	District of Columbia
2004	2005	2006	2007	2008	2004	2005	2006
4029.3	3900	3937	3974.9	4081.9	4852.8	4490	4653.9

Task 5

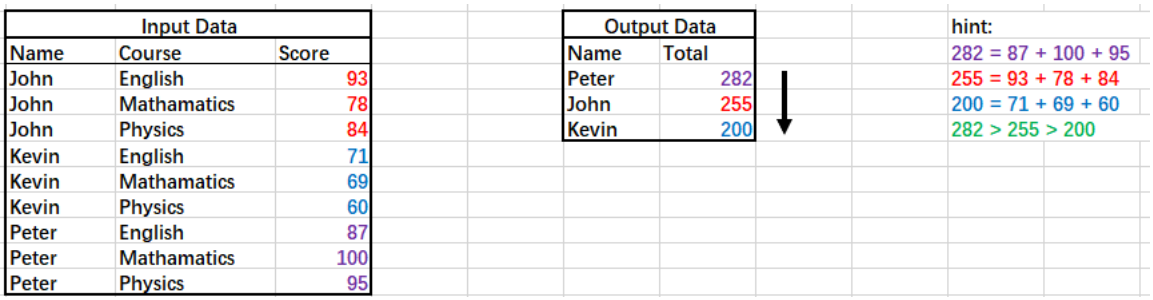
- Input Table

Name	Course	Score
John	English	93
John	Mathematics	78
John	Physics	84
Kevin	English	71
Kevin	Mathematics	69
Kevin	Physics	60
Peter	English	87
Peter	Mathematics	100
Peter	Physics	95

- Output Table

Name	Total
Peter	282
John	255
Kevin	200

- Screenshot of the task description file



Input Data		
Name	Course	Score
John	English	93
John	Mathamatics	78
John	Physics	84
Kevin	English	71
Kevin	Mathamatics	69
Kevin	Physics	60
Peter	English	87
Peter	Mathamatics	100
Peter	Physics	95

Output Data	
Name	Total
Peter	282
John	255
Kevin	200

hint:
282 = 87 + 100 + 95
255 = 93 + 78 + 84
200 = 71 + 69 + 60
282 > 255 > 200

Task 6

- Input Table

Name
John Smith
Peter Wallen
Kevin Micheal
Robin Anderson

- Output Table

First Name	Last Name
John	Smith
Peter	Wallen
Kevin	Michael
Robin	Anderson

Task 7

- Input Table

Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

- Output Table

	2021	2022
January	2821.2	2961.6
July	2883.7	3000.3

- Screenshot of the task description file

Input Data		
Time	State	GDP
January 2021	Alabama	3055.3
July 2021	Alabama	3101.5
January 2022	Alabama	3177.7
July 2022	Alabama	3201.5
January 2021	Alaska	2587.1
July 2021	Alaska	2665.9
January 2022	Alaska	2745.5
July 2022	Alaska	2799.1

Output Data		
	2021	2022
January	2821.2	2961.6
July	2883.7	3000.3

hint:
2821.2 = (3055.3 + 2587.1) /2
2883.7 = (3101.5 + 2665.9) /2
2961.6 = (3177.7 + 2745.5) /2
3000.3 = (3201.5 + 2799.1) /2

Task 8

- Input Table

Name	Course	Score
John	English	93
John	Mathamatics	78
John	Physics	84
Kevin	English	71
Kevin	Mathamatics	69
Kevin	Physics	60
Peter	English	87
Peter	Mathamatics	100
Peter	Physics	95

- Output Table

	<70	>=70
English	0	3
Mathmatics	1	2
Physics	1	2

- Screenshot of the task description file

Input Data		
Name	Course	Score
John	English	93
John	Mathamat	78
John	Physics	84
Kevin	English	71
Kevin	Mathamat	69
Kevin	Physics	60
Peter	English	87
Peter	Mathamat	100
Peter	Physics	95

Output Data		
	<70	>=70
English	0	3
Mathmatics	1	2
Physics	1	2

D. Prompt templates

We provide example prompt templates of the system that we used in our implementation below. Note that the prompts displayed here are only for reference and do not represent the best performance. Engineering efforts such as optimizing the prompts or using tuning methods are welcome and we encourage future research to contribute relevant datasets for fine-tuning the model, which is likely to yield better results.

Extract entities

- Input
  - `data`: String. The text of the sampled raw data.

```
Please identify and extract all existing entities(e.g. date, address) and list their values and sources(the rows or the string fragments of
(output in the form: e.g. [{entity_name: "entity_1", values: [value_1, value_2, ...], source: [source_1, source_2, ...]}, ...], where value
${data}
```

- Output
  - The extracted entities.
  - `entity_name` refers to the name of the entity.
  - `values` includes the values of this entity. (Only the sampled data is considered, so this field will be recalculated in subsequent steps.)
  - `source` is the source rows of the raw data from which the entity is extracted. (Only the sampled data is considered, so this field will be recalculated in subsequent steps.)

Generate the relational table

- Input
  - `data`: String. The text of the sampled raw data.
  - `entity_names`: String. The names of the extracted entities separated by commas, e.g. `date, order_id, order`
  - `eg`: String. The first record of the relational table, which can be composed by combining the `entity_name` and `values` of the extracted entities. E.g. `{"date": "2023-01-01", "order_id": "10001", ...}`

```
/*
Javascript
data = \`${data}\`

Write code to transform the data into a relational table including the following entities: ${entity_names}
Example output:
[${JSON5.stringify(eg)}, ...]
*/
```

- Output
  - The scripts for transforming raw data into the relational table. Note that the script is likely to contain a variable `data` indicating the sampled raw data, and its value should be replaced by the real raw data.

Derive entities

- Input
  - `data`: String. The text of the sampled raw data.
  - `instruction`: String. The user instruction.
  - `extracted_entities`: String. The extracted entities in the “name=[values]” form, e.g. `date=["2023-01-01", "2023-01-02"]`

```
Assume a grammar offers the following functions to derive entities from existing entities:

Function name: sum
Format: sum(E), where E is a numerical entity
Description: Computing the sum of values to aggregate them for summarizing rows.
Example: E = [2, 3, 4], sum(E) = 9
Function name: average
Format: average(E), where E is a numerical entity
Description: Computing the average of the values for some entity.
Example: E = [2, 3, 4], average(E) = 3
Function name: mul
Format: mul(E1, E2), where E1, E2 are numerical entities
Description: Perform a value-wise multiplication on two entities.
Example: E1 = [2, 3, 4], E2 = [2, 3, 4], mul(E1, E2) = [4, 9, 16]
Function name: split
Format: split(E, pattern)[index], where E is a categorical entity, "pattern" is a string, "index" is an integer.
Description: Split all values for an entity E by a given pattern string into several substring arrays, and pick the substrings with given i
Example: E = ["a-h", "b-w", "c-e"], you can derive 2 entities here: split(E, '-')[0] = ['a', 'b', 'c'], and split(E, '-')[1] = ['h', 'w', '
Function name: count
Format: count(E), where E is any entity
Description: Count the number of different values for the entity.
Example: E = [1, 2, 1, 2], count(E) = 2
Function name: ascsort
Format: ascsort(E), where E is any entity
Description: Sort the values of E in ascending order.
Example: E = [1, 2, 1, 2], ascsort(E) = [1, 1, 2, 2]
Function name: descsort
Format: descsort(E), where E is any entity
Description: Sort the values of E in descending order.
Example: E = [1, 2, 1, 2], descsort(E) = [2, 2, 1, 1]
Function name: concat
Format: concat(E), where E is one of the extracted entities, which will be listed later.
Description: Concatenate all values of some entity. Please DO NOT USE IT if not clearly specified!
Example: E = ["a", "b", "c"], concat(E) = ["abc"]
Function name: concat
Format: concat(E1, E2), where E1, E2 are extracted entities, which will be listed later.
Description: Concatenate the values of E1 and E2 correspondingly to derive a new entity. Note this should only be used if expressed explici
Example: E1 = ["a", "b", "c"], E2 = ["a", "b", "c"], concat(E1, E2) = ["aa", "bb", "cc"]
Function name: filterByValue
Format: filterByValue(E, value1, value2, ...), where E is any entity and value1, value2, ... are values.
Description: Filter the values of some entity and keep the values in value1, value2, ... Note that function "filterByValue" should only be
Example: E = ["a", "b", "c"], filterByValue(E, "b") = ["b"]
Function name: filterByBound
Format: E = filterByBound(E, lowerBound, upperBound), where E is any entity and lowerBound and upperBound are numerical values.
Description: Filter the values of some entity and keep the values >= lowerBound and values < upperBound. Note this should only be used if e
Example: E = [10, 20, 30], filterByValue(E, 15, 25) = [20]

Note that you can only use the functions described above. Please do not use other functions.

The grammar also offers a way of combining functions to get composite function. Here is the definition of composite function:
Given a function "g" and a series of function "h_1", "h_2", "h_3", ..., "h_k", where "g", "h_1", "h_2", "h_3", ..., "h_k" are all functions

Note that if you get composite functions, you should derive it step by step. For example, given extracted entities "name", "gender" and "ag
Avg_age = average(age)
Ranked_avg_age = ascsort(Avg_age)
Please use different names when you derive a new entity.

Note that if the users wants to perform a group by operation (e.g. "for A, compute B", "compute A group by B", "pivot A by B", ...), you do

Question: Given the following data and entities extracted from this data, can you derive entities in order to "${instruction}"? If the extr
```

```
Data:\n
${data}
Extracted entities:
${extracted_entities}
```

- Output
  - Derived entities in the `name = specification` form, as shown in the task-specific examples in the prompt. Note that noise may exist in the output so extra engineering is required to extract the needed information, such as using regular expressions.

Construct table specifications

- Input
  - `data`: String. The text of the sampled raw data.
  - `instruction`: String. The user instruction.
  - `extracted_entities`: String. The extracted entities in the “name=[values]” form, e.g. `date=["2023-01-01", "2023-01-02"]`
  - `derived_entities`: String. The derived entities in the “name=specification” form, e.g. `Quantity = split(Order, ' ')[0]`

```
Rigel is a declarative table grammar. It describes table as "(row), (column) => (cell)" where "row", "column", and "cell" are entity expres
1. E = A, where A is an entity (i.e. a column) in the given relational table;
2. E = A + B, where values in entities A and B are concatenated, but '+' can only be used in cell channel;
3. E = A * B, where * denotes the Cartesian product of entities A and B, but '*' can only be used in row and column channel;
4. E can be empty.


Note that in a valid specification, "row" and "column" CANNOT be empty at the same time. For example, "(state), (year) => (crime)", "(state

Example: Consider a specification (state * year), () => (crime), given state, year and crime as extracted entities in order. After transpos


When multiple entities are in the cell channel, an example is (Alabama,"4029,3900",7929\nAlaska,"3900,3615",7515) with specification (state

When multiple entities are in the cell channel, you can change different permutations to generate multiple specifications.


Question: Given the following data, extracted and derived entities from the data, please generate some valid Rigel specifications using app

Rank the specifications and put the most likely ones at the top of your output.

Data:
${data}

Extracted entities:
${extracted_entities}

Derived entities:
${derived_entities}
```

- Output
  - The generated specifications. Note that noise may exist in the output so extra engineering is required to extract the needed information, such as using regular expressions.

Input suggestion

- Input
  - `data`: String. The text of the sampled raw data.

```
Recommend 3 prompts that can be used to transform the following data into a target table. The prompts should be abstract natural language i
Data:
${data}
```

- Output
  - The recommended natural language instructions. Note that noise may exist in the output so extra engineering is required to extract the needed information, such as using regular expressions.

Explain entities

- Input
  - `entity_name`: String. The name of the entity to be explained.
  - `instruction`: String. The user instruction.
  - `extracted_entities`: String. The extracted entities in the “name=[values]” form, e.g. `date=["2023-01-01", "2023-01-02"]`
  - `derived_entities`: String. The derived entities in the “name=specification” form, e.g. `Quantity = split(Order, ' ')[0]`

```
Assume a grammar offers the following functions to derive entities from existing entities:

Function name: sum
Format: E = sum(E), where E is a numerical entity
Description: Computing the sum of values to aggregate them for summarizing rows.
Example: E = [2, 3, 4], sum(E) = 9
Function name: average
Format: E = average(E), where E is a numerical entity
Description: Computing the average of the values for some entity.
Example: E = [2, 3, 4], average(E) = 3
Function name: mul
Format: E = mul(E1, E2), where E1, E2 are numerical entities
Description: Perform a value-wise multiplication on two entities.
Example: E1 = [2, 3, 4], E2 = [2, 3, 4], mul(E1, E2) = [4, 9, 16]
Function name: split
Format: E = split(E, pattern)[index], where E is a categorical entity, pattern is a string, index is an integer
Description: Split all values for an entity E by given pattern string into several substring arrays, and pick the substrings with given ind
Example: E = ["a-h", "b-w", "c-e"], split(E, '-')[0] = ['a', 'b', 'c'], split(E, '-')[1] = ['h', 'w', 'e']
Function name: count
Format: E = count(E), where E is any entity
Description: Count the number of different values for the entity.
Example: E = [1, 2, 1, 2], count(E) = 2
Function name: ascsort
Format: E = ascsort(E), where E is any entity
Description: Sort the values of E in ascending order.
Example: E = [1, 2, 1, 2], ascsort(E) = [1, 1, 2, 2]
Function name: descsort
Format: E = descsort(E), where E is any entity
Description: Sort the values of E in descending order.
Example: E = [1, 2, 1, 2], descsort(E) = [2, 2, 1, 1]
Function name: concat
Format: E = concat(E), where E is any entity
Description: Concatenate all values of some entity.
Example: E = ["a", "b", "c"], concat(E) = ["abc"]
Function name: concat
Format: E = concat(E1, E2), where E1, E2 are any entities
Description: Concatenate the values of E1 and E2 correspondingly to derive a new entity.
Example: E1 = ["a", "b", "c"], E2 = ["a", "b", "c"], concat(E1, E2) = ["aa", "bb", "cc"]
Function name: filterByValue
Format: E = filterByValue(E, value1, value2, ...), where E is any entity and value1, value2, ... are values.
Description: Filter the values of some entity and keep the values in value1, value2, ... Note this should only be used if expressed explici
Example: E = ["a", "b", "c"], filterByValue(E, "b") = ["b"]
Function name: filterByBound
Format: E = filterByBound(E, lowerBound, upperBound), where E is any entity and lowerBound and upperBound are numerical values.
Description: Filter the values of some entity and keep the values >= lowerBound and values < upperBound. Note this should only be used if e
Example: E = [10, 20, 30], filterByValue(E, 15, 25) = [20]

Now the user wants to ${instruction}. Consider there is a derived entity ${entity_name}, please infer what it represents and explain the ca

Extracted entities:
${extracted_entities}

Derived entities:
${derived_entities}
```

- Output
  - The explanation for the given entity.

Refine entities

- Input
  - `entity`: String. The user selected entity in the “spec=values” form, e.g. `year=[2003, 2004]`
  - `instruction`: String. The user instruction.
  - `extracted_entities`: String. The extracted entities in the “name=[values]” form, e.g. `date=["2023-01-01", "2023-01-02"]`
  - `derived_entities`: String. The derived entities in the “name=specification” form, e.g. `Quantity = split(Order, ' ')[0]`

```
Existing entities:
${extracted_entities}
${derived_entities}

Assume a grammar offers the following functions to derive entities from existing entities:

Function name: sum
Format: E = sum(E), where E is a numerical entity
Description: Computing the sum of values to aggregate them for summarizing rows.
Example: E = [2, 3, 4], sum(E) = 9
Function name: mul
Format: E = mul(E1, E2), where E1, E2 are numerical entities
Description: Perform a value-wise multiplication on two entities.
Example: E1 = [2, 3, 4], E2 = [2, 3, 4], mul(E1, E2) = [4, 9, 16]
Function name: split
Format: E = split(E, pattern)[index], where E is a categorical entity, pattern is a string, index is an integer
Description: Split all values for an entity E by given pattern string into several substring arrays, and pick the substrings with given ind
Example: E = ["a-h", "b-w", "c-e"], split(E, '-')[0] = ['a', 'b', 'c'], split(E, '-')[1] = ['h', 'w', 'e']
Function name: count
Format: E = count(E), where E is any entity
Description: Count the number of different values for the entity.
Example: E = [1, 2, 1, 2], count(E) = 2
Function name: ascsort
Format: E = ascsort(E), where E is any entity
Description: Sort the values of E in ascending order.
Example: E = [1, 2, 1, 2], ascsort(E) = [1, 1, 2, 2]
Function name: descsort
Format: E = descsort(E), where E is any entity
Description: Sort the values of E in descending order.
Example: E = [1, 2, 1, 2], descsort(E) = [2, 2, 1, 1]
Function name: concat
Format: E = concat(E), where E is any entity
Description: Concatenate all values of some entity.
Example: E = ["a", "b", "c"], concat(E) = ["abc"]
Function name: concat
Format: E = concat(E1, E2), where E1, E2 are any entities
Description: Concatenate the values of E1 and E2 correspondingly to derive a new entity.
Example: E1 = ["a", "b", "c"], E2 = ["a", "b", "c"], concat(E1, E2) = ["aa", "bb", "cc"]
Function name: filterByValue
Format: E = filterByValue(E, value1, value2, ...), where E is any entity and value1, value2, ... are values.
Description: Filter the values of some entity and keep the values in value1, value2, ... Note this should only be used if expressed explici
Example: E = ["a", "b", "c"], filterByValue(E, "b") = ["b"]
Function name: filterByBound
Format: E = filterByBound(E, lowerBound, upperBound), where E is any entity and lowerBound and upperBound are numerical values.
Description: Filter the values of some entity and keep the values >= lowerBound and values < upperBound. Note this should only be used if e
Example: E = [10, 20, 30], filterByValue(E, 15, 25) = [20]

Question: To replace the entity ${entity}, the user wants to ${instruction}. Please output the entity that can most fulfill the user's inte

Output in the format: entityName = specification (Only in 1 line, no other information should be output.)
If the entity is a new one that you derived just now, the "entityName" should be replaced by a name(letters only, CANNOT be the same as any
Note the specification is not the value list.

Examples:
1. The existing entities are "state", "crime" and "year", and to replace the entity "year=[2003, 2004]", the user wants to "change it to st
2. The existing entities are "state", "crime" and "year", and to replace the entity "sum(crime)=[3305, 3046]", the user wants to "sort it i
```

- Output

- The specification for the entity after the refinement. Note that noise may exist in the output so extra engineering is required to extract the needed information, such as using regular expressions.

Refine specifications

- Input
  - `data`: String. The text of the sampled raw data.
  - `entity`: String. The user selected entity in the “spec=values” form, e.g. `year=[2003, 2004]`
  - `newInstruction`: String. The user instruction for refinement.
  - `current_spec`: String. The specification of the current table.
  - `extracted_entities`: String. The extracted entities in the “name=[values]” form, e.g. `date=["2023-01-01", "2023-01-02"]`
  - `derived_entities`: String. The derived entities in the “name=specification” form, e.g. `Quantity = split(Order, ' ')[0]`

```
Rigel is a declarative table grammar. It describes table as "(row), (column) => (cell)" where "row", "column", and "cell" are entity expres
1. E = A, where A is an entity chosen from extracted entities or derived entities, which will be listed later;
2. E = A + B, where values in entities A and B are concatenated, but '+' can only be used in cell channel;
3. E = A * B, where * denotes the Cartesian product of entities A and B, but '*' can only be used in row and column channel;
4. E can be empty.


Note that in a valid specification, "row" and "column" CANNOT be empty at the same time. For example, "(state), (year) => (crime)", "(state

Example: Consider a specification (state * year), () => (crime), given state, year and crime as extracted entities in order. After transpos


When multiple entities are in the cell channel, an example is (Alabama,"4029,3900",7929\nAlaska,"3900,3615",7515) with specification (state

When multiple entities are in the cell channel, you can change different permutations to generate multiple specifications.


Question: Given the following data, extracted and derived entities from the data, and the user wants to ${instruction}, you initially gener

Rank the specifications and put the most likely ones at the top of your output.
Make sure the specifications you output are in the form (...), (...) => (...)

Data:
${data}

Extracted entities:
${extracted_entities}

Derived entities:
${derived_entities}
```

- Output
  - The specifications for the current table after the refinement. Note that noise may exist in the output so extra engineering is required to extract the needed information, such as using regular expressions.