Kushal Gevaria (kgg5247)
Akshay Pudage (ap7558)

CSCI 621: Database System Implementation
H2 DATABASE
Assignment 3

As a part of this assignment we have created a new aggregate feature, PIVOT, which selects the returns the pivot or the median of a column within a set of records. Unlike the built-in MEDIAN aggregate function which works only with integer data type, this function is designed such that it works with any data type. Thus, it can be considered as a generic version of MEDIAN. It can be used to in algorithms like quick sort and merge sort if sorting needs to be implemented in the database.

Implementation Steps

1. Additions in **AggregateDataDefault** class:
   This is a built-in class which is used for storing data while calculating any aggregate function. In case of PIVOT, this class is used for storing pivots tables of a database in an Arraylist.
   a. We created an Arraylist named pivot for storing the pivots which holds objects of Value class.
   b. Inside add function, we added a case for the pivot function in the switch case. This function is called by the Aggregate class while computing various aggregate functions. Within this pivot case, for add, there are two possibilities, if the instance variable value is null, we convert the incoming value object into the necessary data type and then add it into the pivot Arraylist. If the instance variable is not null, we convert the incoming value object to the datatype of the value object stored as an instance variable in the class and then add it to the Arraylist.
   c. Next, we modified the getValue function. We added a case for pivot in the switch case within the getValue function. This case basically gets the middle element of the pivot Arraylist and stores it in the Value object which is then converted to the necessary data type and returned.

2. Additions in the **Aggregate** class:
   This class holds the implementation of various aggregate functions available within the H2 database. We have added the implementation of our pivot function within this class.
   a. We added a new entry to the AggregateType enum for pivot named PIVOT.

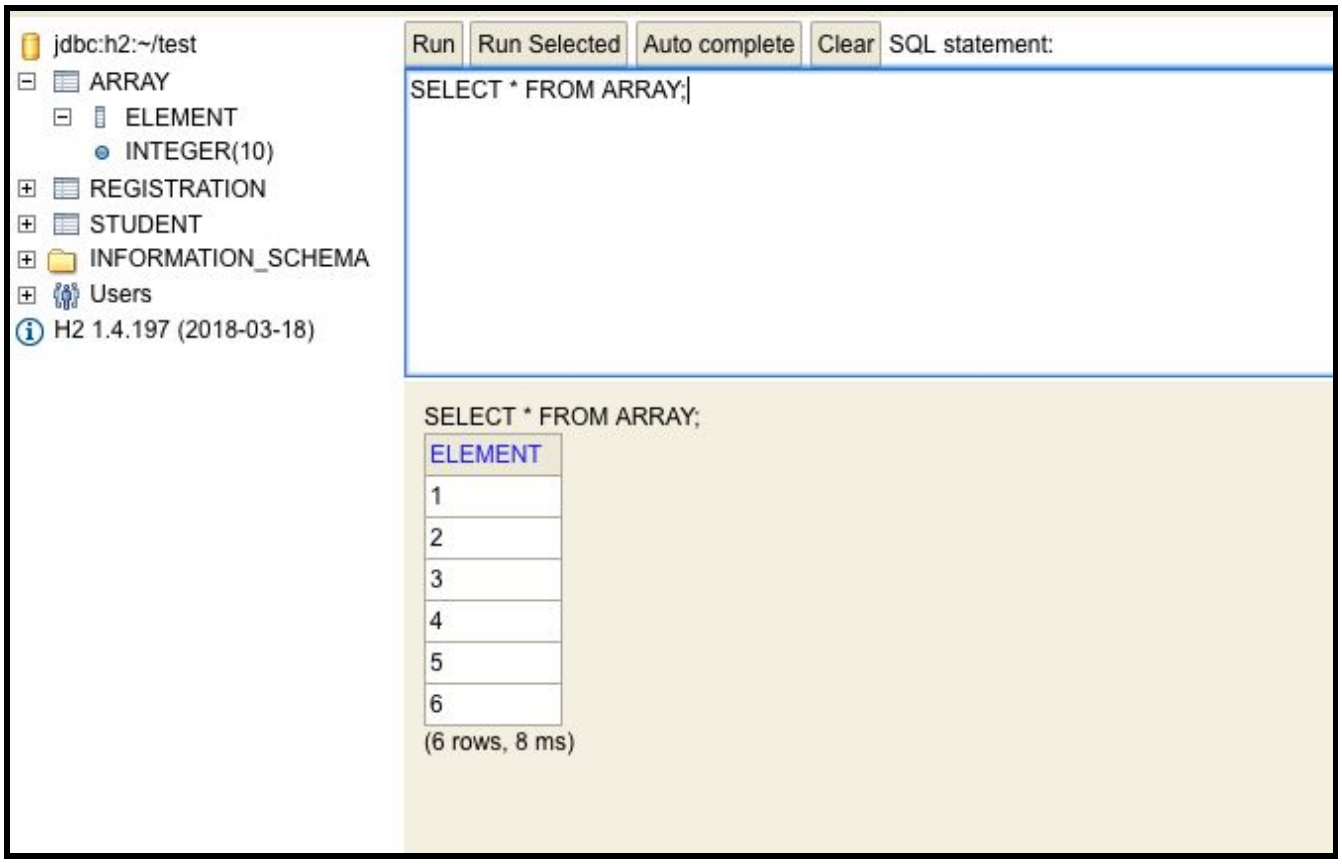
b. We then added a line to add the newly created aggregate function to the AGGREGATES map which holds the name of the aggregate function as key and the AggregateType enum as the value within a static block of the Aggregates class. This ensures the map is always initialized with our user defined function, whenever the Aggregate class is instantiated.
c. Next, we added a case for pivot within the switch case of the getValueQuick function. This case basically gets the rows of the table and uses the Cursor object's getSearchRow function to get the middle row of the table and return the value of that row as the pivot for the table. This function makes use of the index on the column that is available on the table to quickly find the required element. If the index is not available on the column, we manually iterate through the values to find the pivot value. If the pivot value for the column is null, we return a null instance of the Value object, otherwise we return the middle or the pivot element of the column. In case of even number of records, we return the last value of the middle two values as pivot.
d. Within the optimize function switch case we add a case for the pivot. Any future optimizations for the pivot function can be made here.
e. Next, we add a case for the switch case within the getSQL function. This is basically used by the SQL parser to parse the PIVOT keyword and call appropriate pivot functions within the code.
f. We add a case for the switch case within the isEverything function. This basically checks if an index is available on the column for which we are to find the pivot.

3. Creating a **Pivot** class:
   We create a pivot class which implements the built-in AggregateFunction class. Here we override and implement the necessary classes as follows:
   a. **getType**:
      This method simply returns the unique identifier of the data type of the pivot element.
   b. **add**:
      This method is called whenever a new value is added to the column for a table. It maintains the pivot element of the column within the result instance variable of the Pivot class. This stored pivot can be used for a quick retrieval of the pivot element.
   c. **getResults**:
      This method simply returns the stored pivot element which is calculated in the above add function.

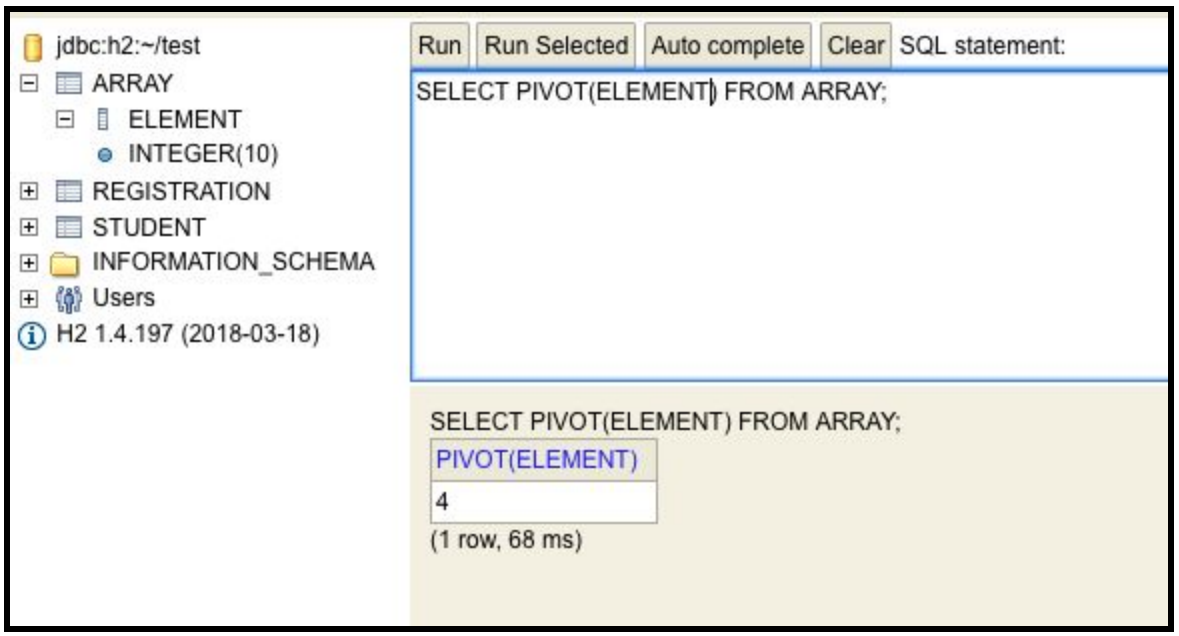
Kushal Gevaria (kgg5247)
Akshay Pudage (ap7558)

Demo Screenshots

Viewing data added within the table

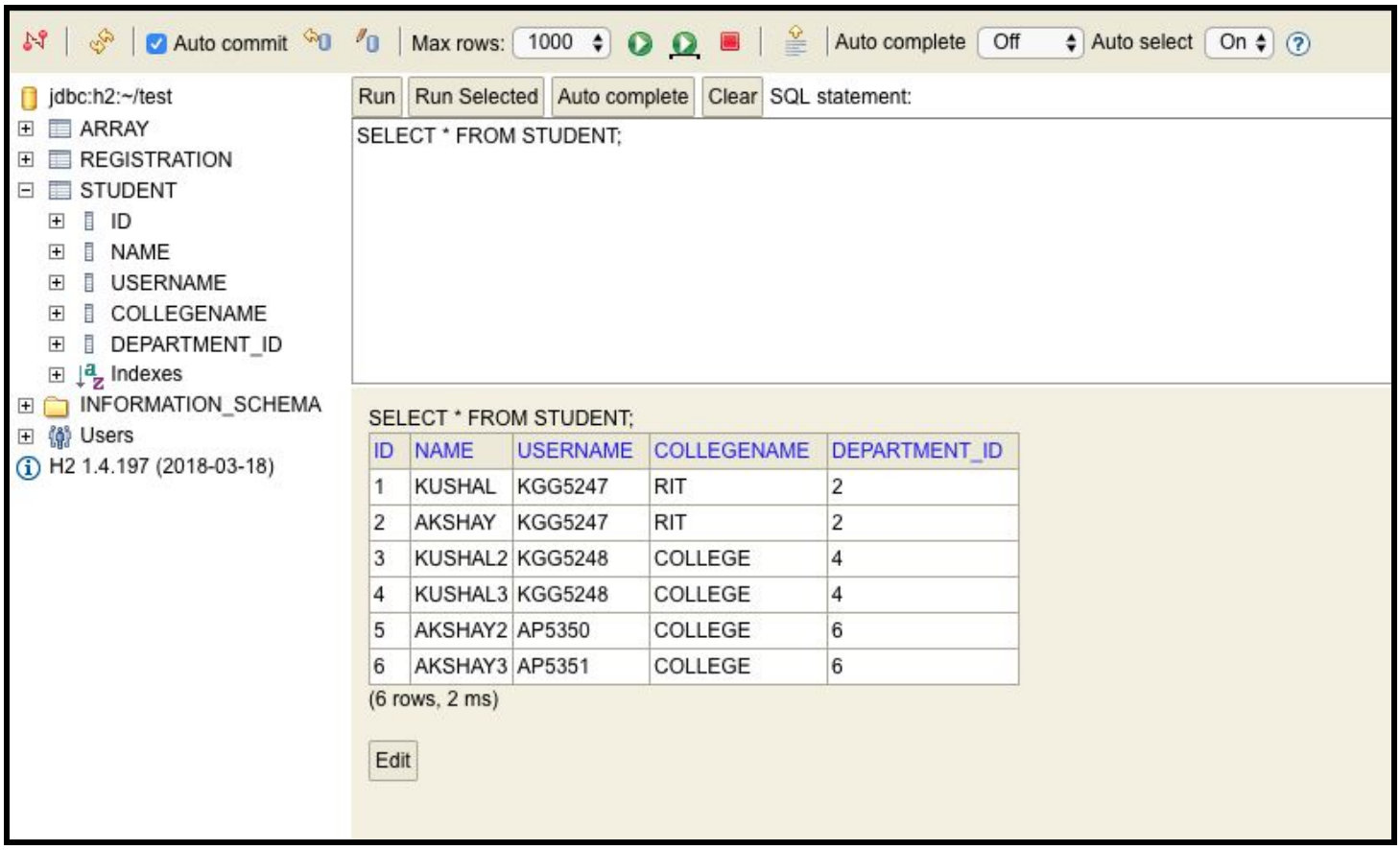


Viewing the pivot of the Elements column



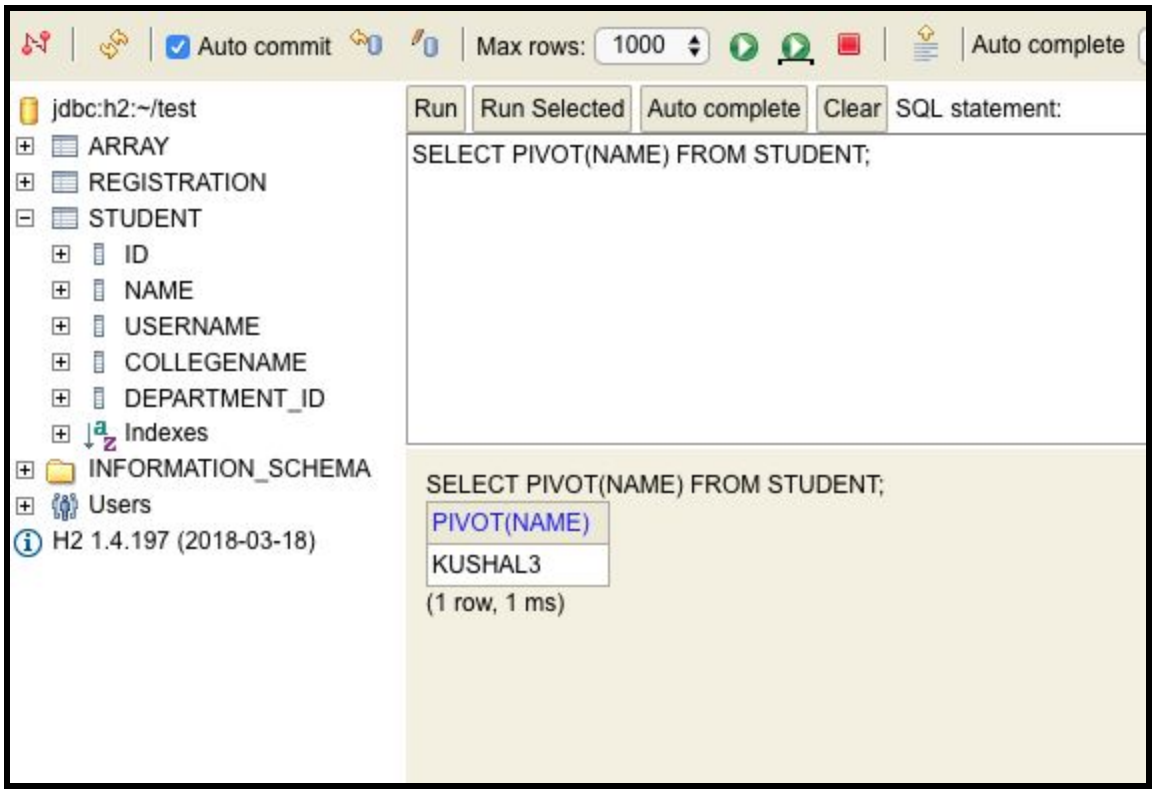
Kushal Gevaria (kgg5247)
Akshay Pudage (ap7558)

Viewing the data within the Student table

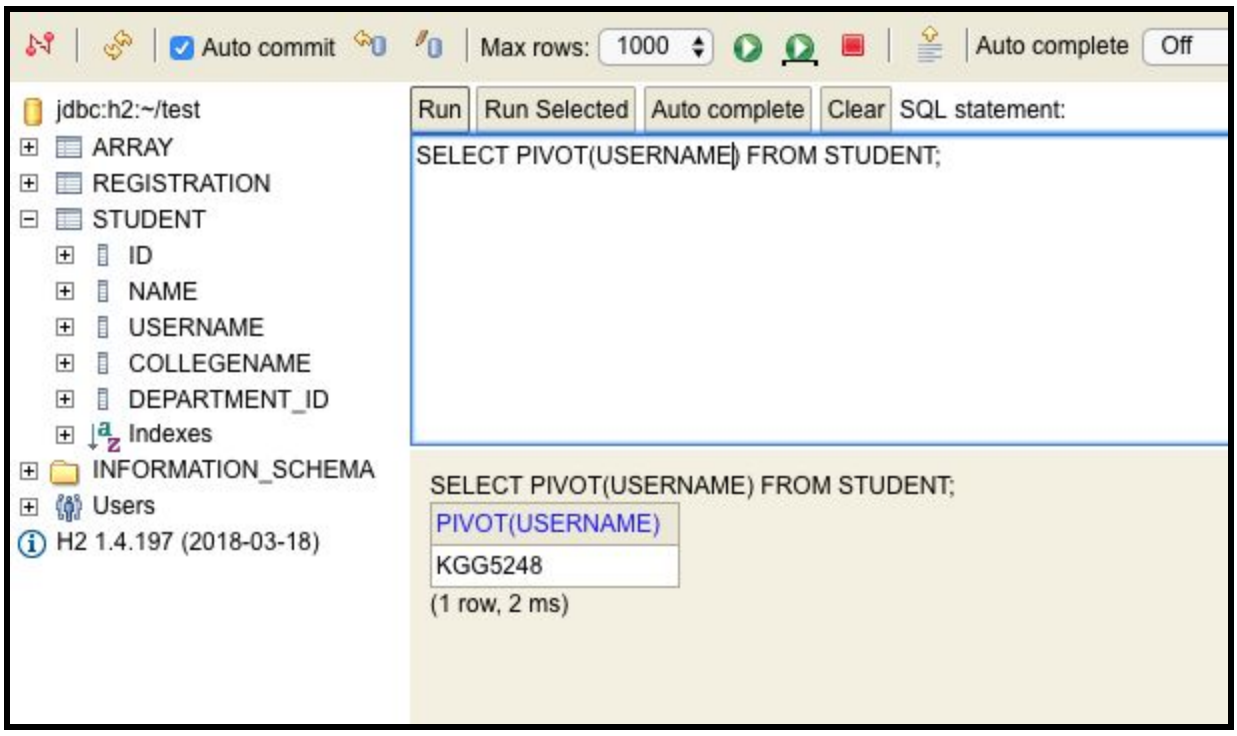


Kushal Gevaria (kgg5247)
Akshay Pudage (ap7558)

Viewing the pivot of the name column of the Student table



Viewing the pivot of the Username column of the Student table



Kushal Gevaria (kgg5247)
Akshay Pudage (ap7558)

Viewing the pivot of all columns correctly generates error as the pivots can only be computed for a column at a time.