H2 Database Phase 3

Akash Desai ad3059@rit.edu
Nihal Parchand np9603@rit.edu
Viraj Chaudhari vc6346@rit.edu

INTRODUCTION:

For implementing an enhanced feature for H2 Database in terms of query processing and optimization we have decided to implement a new aggregate function called GreaterThanAverage and LessThanAverage. Aggregate functions are functions that combines multiple records and then perform the function and return a single value as result.

- **GreaterThanAverage** is an aggregate function that calculates the average of all the values of a column in the table and returns the count of records that have a value greater than the average. The initial input is given in integer data type format and the aggregate function returns the result in integer format.
- **LessThanAverage** is an aggregate function that calculates the average of all the values of a column in the table and returns the count of records that have a value less than and equal to the average. The initial input is given in integer data type format and the aggregate function returns the result in integer format.

For performing the same operation in H2, we have to use a subquery.

LessThanAverage

Example: Select count(*) from employee where salary <= (Select avg(salary) from employee);

Using implemented **LessThanAverage** aggregate function:
Select **LT_AVG(salary)** from employee;

GreaterThanAverage

Example: Select count(*) from employee where salary > (Select avg(salary) from employee);

Using implemented **GreaterThanAverage** aggregate function:
Select **GT_AVG(salary)** from employee;

IMPLEMENTATION:

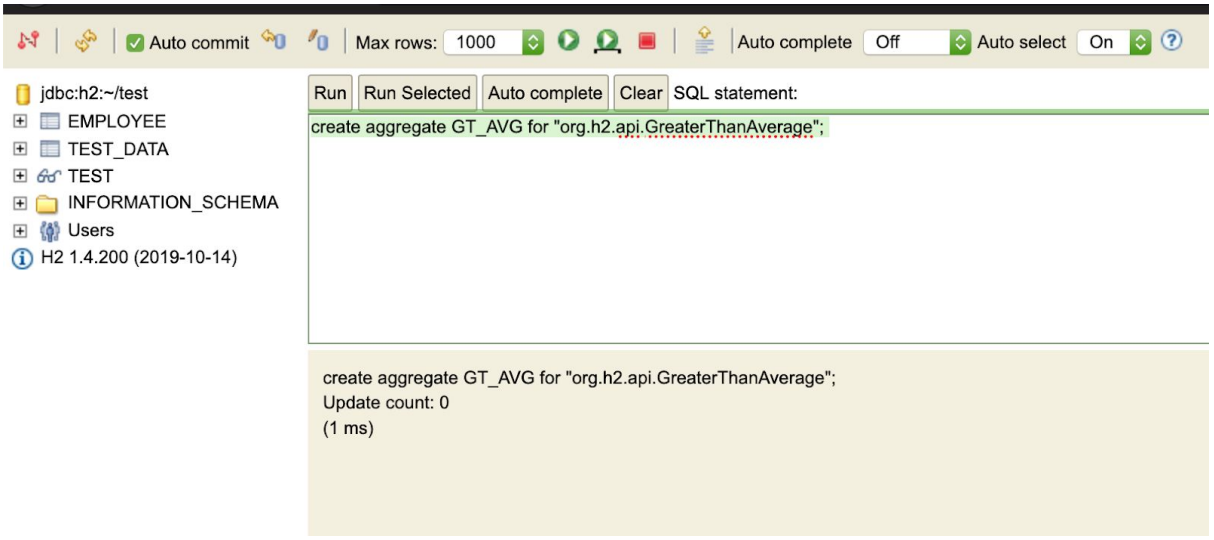
H2 allows users to create their own aggregate functions by implementing the minimal required AggregateFunction interface. To use this feature, the GreaterThanAverage.java and LessThanAverage.java file should implement all the abstract methods defined in the AggregateFunction interface (add, getResult , getType , init methods). The created java file needs to be added to the org.h2.api package.

```
    */
    @Override
    public void add(Object o) throws java.sql.SQLException {
        Object value = o;

        if (count == 0) {
            //Initialize all the variable on processing the first row
            sum = (Integer)value;
            max = value;
            min = value;
            numbers.add((Integer)value);
        }
        else
        {

            if ((Integer)value < (Integer)min) {
                min = value; //Stores the min value
            } else if ((Integer)value > (Integer)max) {
                max = value; //Stores the max value
            }
            sum += (Integer) value;  //Add  to the  sum
            numbers.add((Integer)value);  //Add number to the list
        }
        count++;
    }

    /**
     * This method returns the computed aggregate value. This method must
     * preserve previously added values and must be able to reevaluate result if
     * more values were added since its previous invocation.
     *
     * @return the aggregated value
     */
    @Override
    public Object getResult() throws java.sql.SQLException {

        Integer avg = sum/numbers.size(); //Average of values

        int count = 0; //Number of values greater than average

        for(int i = 0;i < numbers.size();i++){
            if(numbers.get(i) > avg)
                count++;
        }
        return count;
    }

}
```

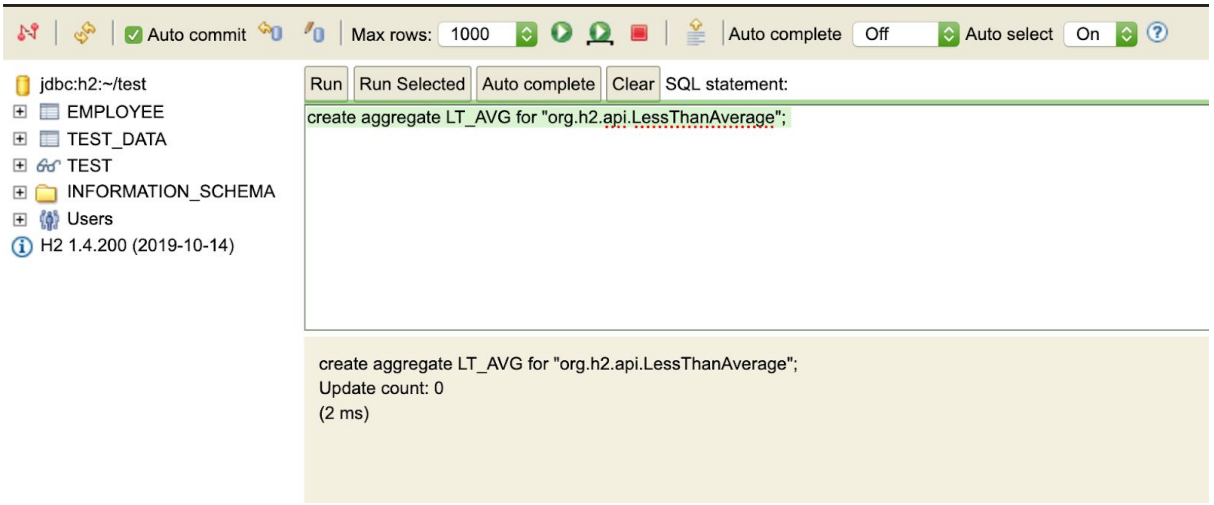
Arraylist is used to store the values of a column. In the implementation of add function we are finding minimum and maximum value of a column. Each value is added to the arraylist and sum is incremented by value. The purpose of storing the value in arraylist is to get the total number of records. This will be used for calculating average. In the implementation of getResult function the average value is calculated using the sum and size of the arraylist. Each value in the arraylist is compared with the calculated average. In LessThanAverage aggregate function count will be incremented if the value is less than or equal to the calculated average. In GreaterThanAverage aggregate function count will be incremented if the value is less than or equal to the calculated average.

For adding the aggregate function in H2 database, we need to execute the following command in the H2 interface.

Create aggregate <function_name> for “org.h2.api.<file_name>”;



Create aggregate GT_AVG for “org.h2.api.GreaterThanAverage”;



Create aggregate LT_AVG for “org.h2.api.LessThanAverage”;

DEMO:

1. Creating a table employee

Create a table employee with attributes ID INT, name VARCHAR2(10) and salary INT.

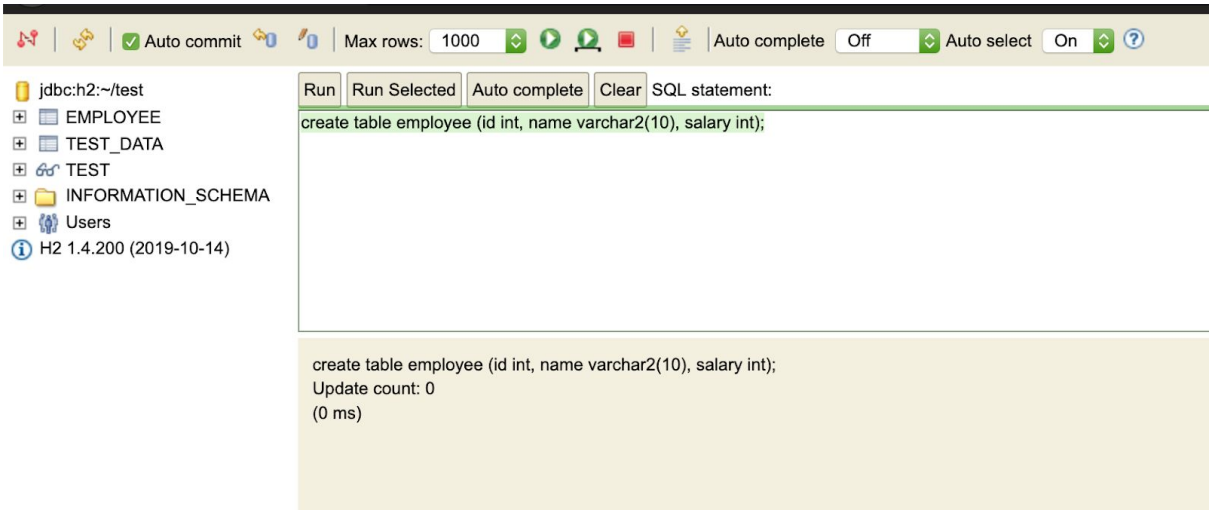
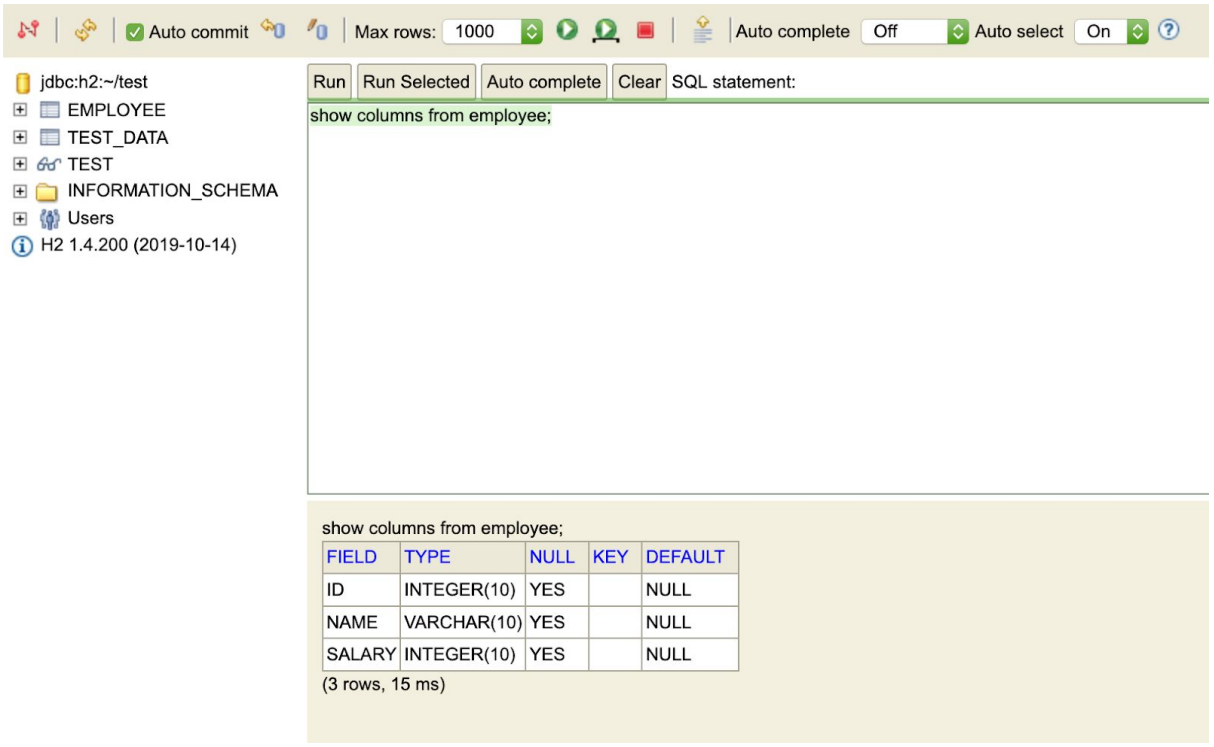
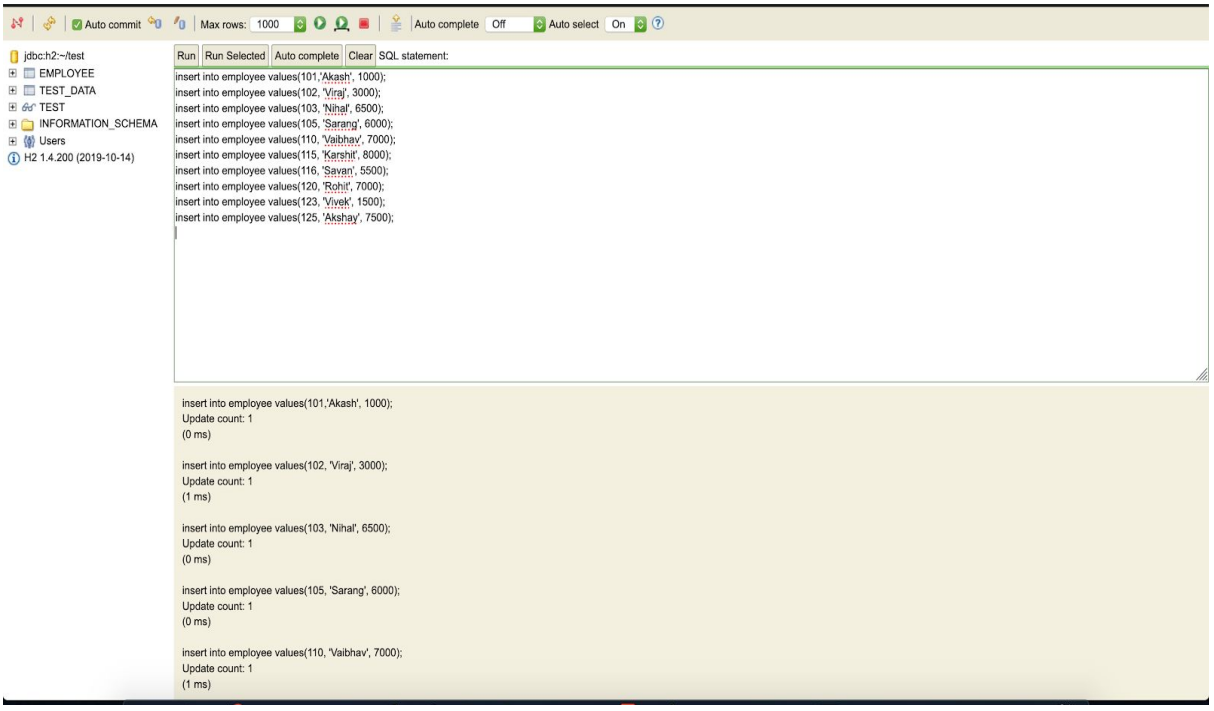


Table employee is successfully created with attributes ID INT, name VARCHAR2(10) and salary INT.

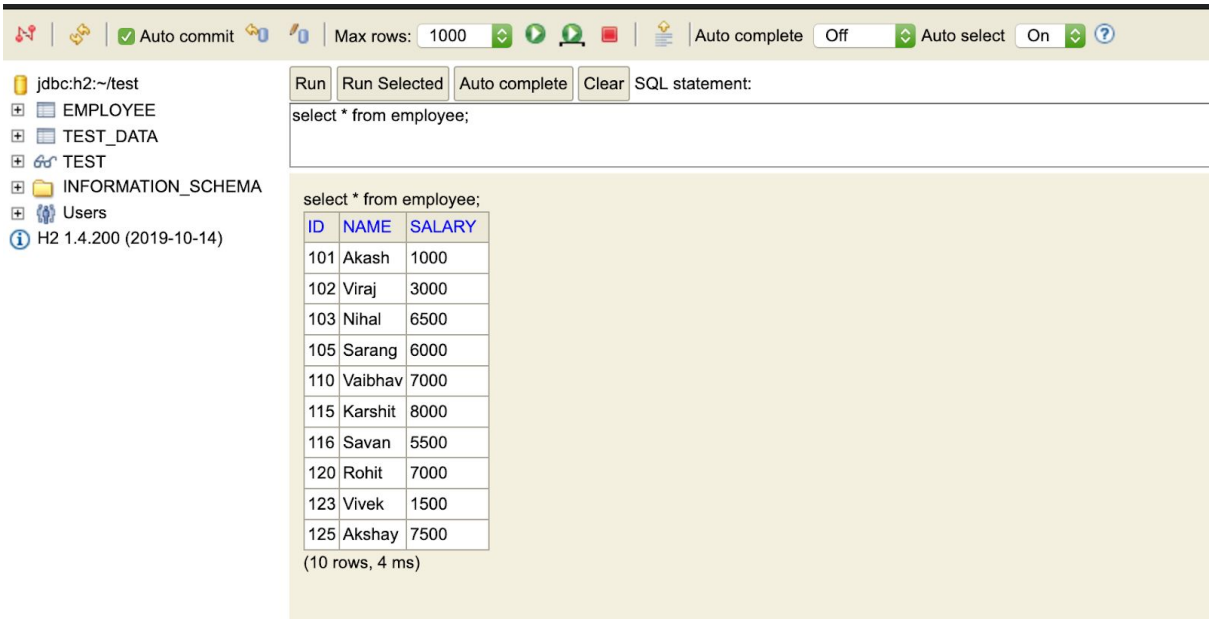
2. Displaying the column information for the created table

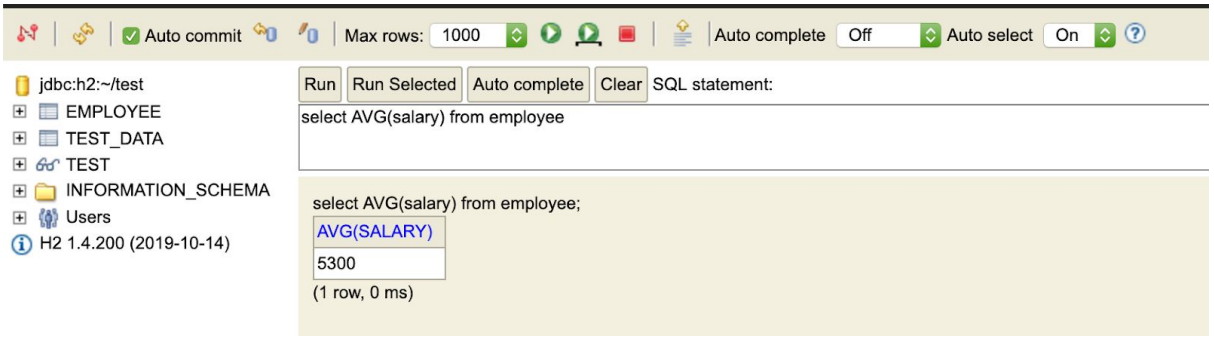


3. Inserting new records in the user info table

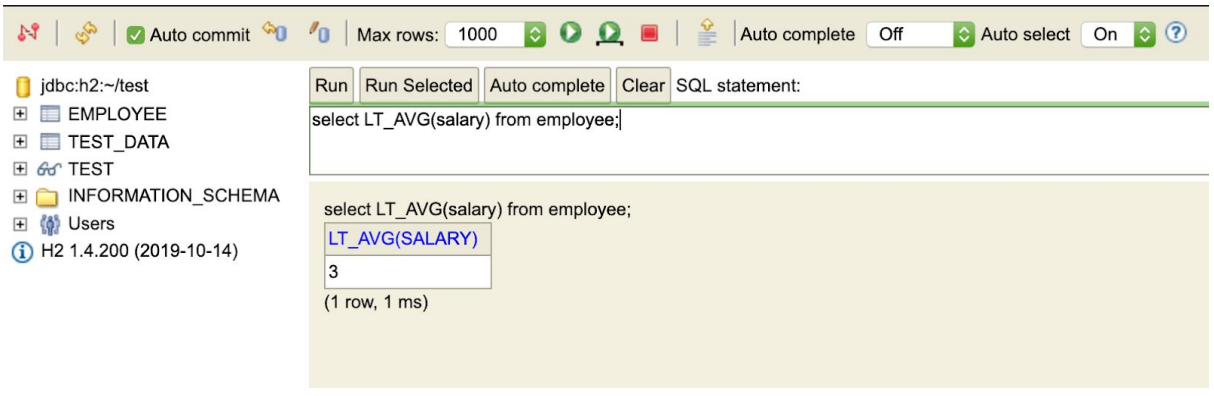


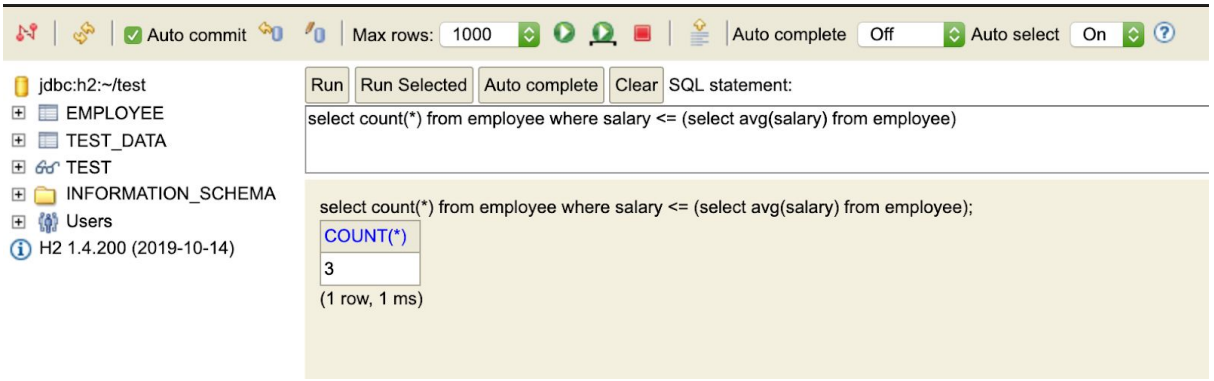
4. Displaying records of the user info table

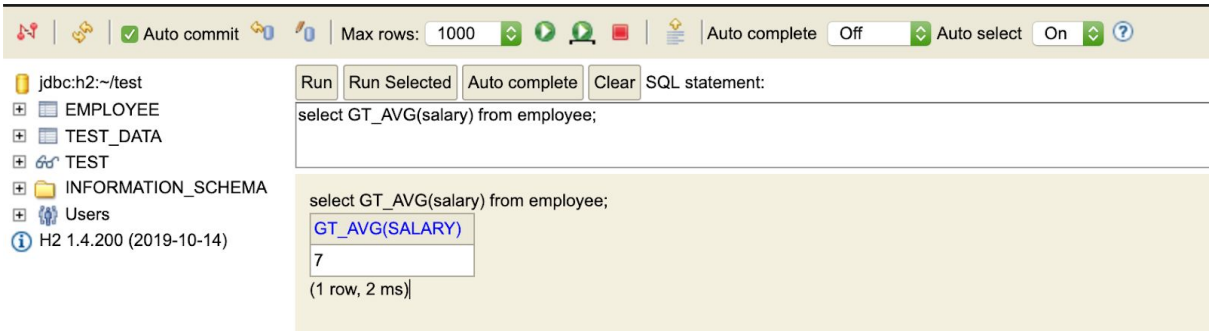


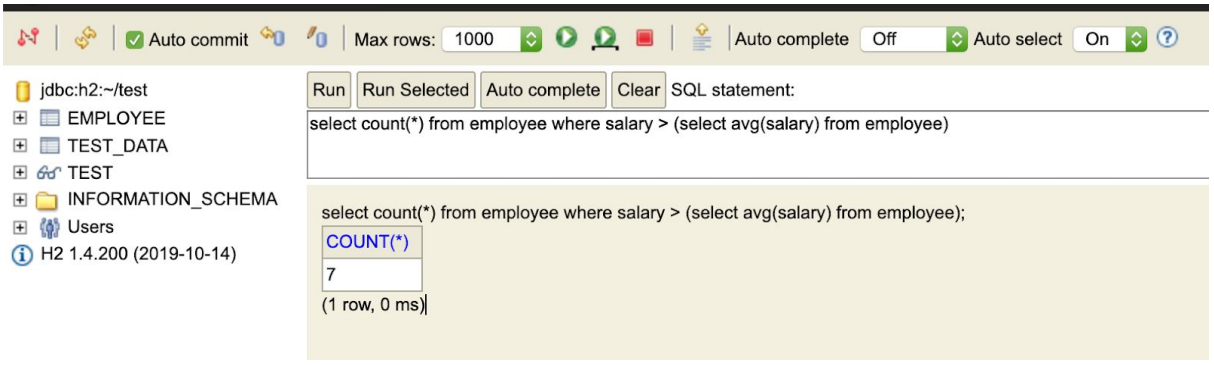


5. Displaying the aggregate function results









REFERENCES:

1. https://github.com/h2database/h2database

2.https://github.com/h2database/h2database/blob/master/h2/src/main/org/h2/api/AggregateFunction.java

3.http://h2-database.66688.n3.nabble.com/user-defined-aggregate-function-FIRST-and-LAST-td4031866.html