Vizier First Steps Assignment

This homework's purpose is to familiarize you with the Vizier system and make sure you have a working Vizier setup. We will give each group Successfully completing the assignment by the deadline some small bonus on their data curation project score. The assignment consists of a set of tasks described below. Upload the screenshots to your group's git repository.

- **Deadline: 09/28**

Install docker, download the vizier image, and start a Vizier container

See here for instructions https://github.com/IITDBGroup/cs520/blob/master/vizier/README.md

Create a project

From the main UI create a project called *"public_schools"* by pressing the "+" button on the landing page.

Load the public school dataset

A project (or notebook) in Vizier is initially empty. In contrast to Jupyter, datasets are first-class citizens in Vizier. To be able to work on a dataset, you have to first load it into your notebook. This is done using a *"Load Dataset"*. Press the plus button to add a new cell at the end of the notebook and select the load dataset cell.

You can either drop a local file onto the cell or load a dataset from an URL. For this example, we will load the public schools dataset from an URL. Select *"From the Internet"* and put in the following URL: `https://raw.githubusercontent.com/IITDBGroup/cs520/master/vizier/Chicago_Public_Schools_-_Progress_Report_Cards__2011-2012_.csv` and give the dataset a name *"schools"*. Press *"Submit"* to save the edits to this cell. Vizier will now execute the cell and load the dataset, try to determine the datatypes of columns automatically and try to guess whether the dataset has column headers or not.

Note that you can explore any dataset in Vizier also through a spreadsheet interface by selecting the dataset from the menu on top. This allows you to apply spreadsheet-style edits to the dataset.

- **Task 1**: load a dataset and take a screenshot of the result

Explore the datasets and caveats

Dataset statistics

Notice how Vizier shows the loaded dataset below the cell and has correctly identified the data types for the columns. You can switch between 3 views on the top right:

- **compact view**: this is what you are looking at now
- **detail view**: this view shows data distributions for each column
- **column view**: this view shows more statistics about each column, but not the actual data in the spreadsheet
- **Task 2**: Select the detail view and look at the distributions of some columns. Then look at the column view and take a screenshot of the distribution for column `Teachers_Score`.

Note that the `Teachers_Score` has a lot of null values.

Caveats

Let's explore this further. Switch back to the compact view and scroll to the `Teachers_Score` column. Vizier has identified (correctly) that this column is of type `short` (small integer values). However, there are some values that are missing and have a question mark beside them. Such values are called **Caveats** in Vizier. Intuitively a caveat indicates that there is some potential problem with a value. By clicking on the question mark, you can see why Vizier has marked a particular value with a Caveat.

- **Task 3**: Click on one of the question marks for values in the teachers column and take a screenshot.

In this case, some of the values in this column cannot be interpreted as integer values. Vizier has replaced these with null values, but has kept track of this problem in the caveat message for the value. Vizier propagates Caveats through most operations to inform you which analysis results you have derived depend on potentially problematic data. Let's see this in action.

SQL cells and caveat propagation

Let's run some basic analysis over the dataset. In addition to Python and Scala cells, Vizier also supports SQL natively. Let's start by focusing on two particular columns.

- **Task 4**: Create a SQL cell and write a query that returns columns `Teachers_Score` and `Community_Area_Name`. SQL results can be stored as new datasets in Vizier. Call the result dataset `score_and_community`. And take a screenshot of the result.

Now let's add another SQL cell to compute the average `Teachers_Score` per community area (attribute `Community_Area_Name`) over the `score_and_community` dataset and order the result in descending order of average scores.

- **Task 5**: Create a SQL cell and write a query over the over the `score_and_community` dataset that computes the result as described above. Call the result dataset `community_teacher_scores`. And take a screenshot of the result.

Note that some result values are highlighted in red and marked with caveats. These are aggregation results that were computed from cells with caveats and, thus, may not be trustworthy.

Plotting data

To get a better overview of the results, let's plot them! For simple plots, Vizier has a dedicated cell type *"Simple Chart"*.

- **Task 6**: Create a line chart of the aggregation result by creating a plot cell and take a screenshot of the result.

Lenses

Vizier also supports special cell types for common data cleaning and integration operations. These cell types are called *"Lenses"*. Let's use the missing value imputation lens to impute missing values in the `Teachers_Score` column.

- **Task 7**: Insert a new cell above the SQL cell that computes the average teacher scores (notebooks in Vizier are executed top down) by pressing the three bars below the cell number. Select *"Impute Missing Values"*, select the `score_and_community` dataset and `Teachers_Score` as the column to be imputed, and select mean as the imputation method and take a screenshot of the updated line chart.

Vizier automatically refreshes the result of a cell when a cell whose results it depends on is updated. Observe how your SQL query result and plot are updated.

Using Python

Furthermore, as in Jupyter, you can have cells with Python code. However, in contrast to Jupyter cells in Vizier are isolated from each other. That means the common pattern of declaring variables / functions in one cell and then using them in a subsequent cell is not possible. To enable modularization, Vizier allows you to export and import artifacts (functions, variables, datasets) through its API. For example, to make a function declared in one cell available in another cell, you have to call Vizier's Python API to export the function to be able to use it. For more details about the API see https://github.com/VizierDB/vizier-scala/wiki/Cell-Python

Exporting and importing functions and values and accessing datasets

Let's start by defining a function that prints the average teacher scores from the `community_teacher_scores` dataset.

- **Task 8** Create a Python cell at the end of the notebook and create a function called `print_avg_teachers` that uses Vizier's API to get a handle for this dataset and print all values of the `avg_teacher_score` column. *Hint: use the "Show Code Examples" buttom to see example Vizier API usage and see here for the API documention*. Then use `vizierdb.export_module` to export the function. Then create a second Python cell and use `vizierdb.get_model("print_avg_teachers")` for importing the function and then call it. Take a screenshot of the result.

Get Vizier dataset as a Pandas DataFrame

You can also access a Vizier dataset as a Pandas DataFrame.

- **Task 9** Create another Python cell and use Vizier's API to access the dataset `community_teacher_scores` as a DataFrame, then filter out rows where the `avg_teacher_score` is larger than or equal to `30.0` and then print the remaining rows and take a screenshot.