645 Final Project Report
Project 3: SeeDB: efficient data-driven visualization recommendations to support visual analytics

Team: Dhruvi Tahilramani, Devyani Mishra, Harshita Loganathan
GitHub:- https://github.com/dhruvitahil/SeeDB.git

Overview of the Paper:

This paper "SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics" discusses SeeDB which is a visualization recommendation engine that was developed to help data analysts in choosing the most informative visualizations automatically. The biggest issue is with the scale and utility of high dimensional dataset, while SeeDB has a deviation-based utility metric that helps in identifying far-off visualizations from a reference dataset. Hence, it is able to identify possibly interesting data trends that other systems are not able to recommend. Key optimization techniques that are used by the system are: pruning and sharing. These techniques ensure effective management of computational resources while considering interactive response times. It is highly flexible and powerful, and this makes it a useful tool to enhance productivity and discovery in the analysis of the data-driven insights the tool offers.

Introduction:

We work on replicating some of the results following the methodology explained in the SeeDB paper. The authors discuss an intermediate level visualization recommendation engine over and above any arbitrary database system. It can conduct speedy visual analysis by exploring all possible visualizations of a data set and computing the 'utility' of each visualization before recommendations are made. Implementation of the proposed algorithms includes query rewriting for shared-based optimizations and the use of the Hoeffding-Serfling inequality for pruning optimizations. We have used K-L Divergence as the utility metric for this experiment on the census dataset. The following sections explain the problem statement, dataset details and preprocessing steps, algorithms with detailed steps of implementation and assumptions, implementation details, and the findings of the project.

Project Scope and Objectives:

In this project, we want to implement and evaluate a few algorithms which realize the functionality suggested in the "SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics" paper. The primary goal would be to identify the top 5 aggregate views from the census dataset which use K-L Divergence as the utility measure. We discuss two optimization strategies suggested by SeeDB:

- **Shared-based Optimization through Query Rewriting:** This is reusing computation results within and among many queries in order to minimize the processing time and maximize the efficiency of the system.

- **Pruning-based Optimization Using Hoeffding-Serfling Inequality:** Less-promising visualizations get pruned away early in the computation, allowing the resources to be used optimally toward the more promising candidates.

For this evaluation, we developed the user-informed query to examine the married, and the reference query to look at the unmarried people, from our census dataset collection. The effectiveness of the visualization is then measured as the difference (K-L Divergence) between the data distributions of those two sets, highlighting the compound views which exhibit the greatest discrepancies, and therefore possess the greatest usefulness.

The specific tasks include:

- Understanding the general method of approach and assessment as presented in the paper.
- Implementation of the algorithm according to the definitions in the paper, with all the mathematical formulations including the aggregate views and the utility function.
- Apply the SeeDB algorithms on the census dataset to derive the top-5 visualizations that maximize the utility measure.

About the dataset:

The Census Income dataset, also known as the Adult dataset, was sourced from the 1994 Census database by Ronny Kohavi and Barry Becker from the Data Mining and Visualization group at Silicon Graphics. It's designed to predict whether an individual's income exceeds $50,000 per year based on census data. The dataset comprises 48,842 instances, each with 14 features that combine both categorical and continuous types.

Key Attributes:

- Age: Continuous. Represents the age of the individual.
- Workclass: Categorical. Includes private, federal government, local government, etc.
- Education: Categorical. Lists levels of education from preschool to doctoral degree.
- Marital Status: Categorical. Includes married, divorced, never-married, etc.

- Occupation: Categorical. Represents the individual's job role, such as tech-support, craft-repair, etc.
- Relationship: Categorical. Specifies familial relationships like wife, own-child, husband, etc.
- Race: Categorical. Includes White, Asian-Pac-Islander, Amer-Indian-Eskimo, Black, and Other.
- Sex: Categorical. Male or Female.
- Capital Gain and Capital Loss: Continuous. Investment income data.
- Hours per Week: Continuous. Represents the number of hours worked per week.

The rich variety of attributes in this dataset, which includes demographic, employment, and income data, provides a comprehensive view of the socio-economic status of individuals. This makes it an ideal dataset for SeeDB, which can leverage this variety to recommend insightful visualizations for quick and efficient data analysis.

Methodology and Implementation Strategy:

Data Preprocessing and Assumptions:

The dataset of the project was sent as a single CSV file. To make sure it is easier for further analysis, we set up a PostgreSQL database and further created a schema that replicates the original data file structure. The columns with missing values were handled so that the integrity of the data would be maintained when loading them into the database.

Assumptions:

Our project's key assumption was the need to compare characteristics for the married and the unmarried; hence, preprocessing for marital status data. The 'marital-status' field in data has multiple categories, which we make into two groups for clarity and in order to manage a balanced comparison:

Married Group:

- Married-civ-spouse
- Married-spouse-absent
- Married-AF-spouse
- Separated (included as these individuals are technically still married)

Unmarried Group:

- Divorced
- Never-married
- Widowed (included as they are currently unmarried)

This classification helps isolate and picture the data in terms of marital status, in concordance with our project objectives for finding patterns that can distinguish these two groups from each other. As we mentioned before, we constructed a new database in which we had a single major table 'adult'. For the analysis purposes, we divided the 'adult' table into two separate tables, for married and unmarried people.

Development Environment:
The project development was carried out using Python 3.6, with several packages to support database connectivity and data manipulation:

- psycopg2: For connecting to the PostgreSQL database.
- numpy: For numerical computations.
- pandas: For data manipulation and retrieval.
- scipy: Specifically for calculating K-L divergence, supporting our analysis of the utility of different visualizations based on the defined groups.

The aggregate functions considered in our analysis include minimum, maximum, sum, average, and count, which were utilized to explore various aspects of the data across the defined groups.

Comprehensive Search Approach:
We designed our approach with the defined methodology, which consists of three sets: attributes (a), measures (m), and functions (f) to the aggregation. We then executed SQL queries for each possible combination of the three sets over the views of the target and reference, followed by storing the results in structured data formats.

For example, the challenges we went through during this alignment process were column discrepancies across the views, like a number of missing rows or dissimilar column values. For example, if a 'native country' column has a value of 'Yugoslavia' in some view and not in others, this results in unbalanced results because the insertion of a column value with an aggregate value of zero leads to a driving K-L divergence in the method used to estimate utility, since inclusions of zeros may drive the divergence result to infinity.

To ensure correctness of our results, we smoothed out alpha. That is, if the aggregates were zero or some missing value for a column, we replaced the value by a very small alpha, typically 10^-10 in our experiments. We normalized the results after querying such that each results was scaled between 1 and 0, from which we computed the view's utility. We retained the views that had the best k utility scores to proceed with further analysis based on the optimization techniques the original authors recommended.

Shared-Based Optimization:
For our example, a large number of queries have been executed, which is 2×f×a×m, because of multiple groupings on the same attribute and having to compute different aggregates on different columns. One important aspect of our sharing-based optimization was that it aims at minimizing this by grouping all aggregate computations together into the minimum number of queries. For every combination, we developed a single query that aggregated values of the columns for all of the aggregates (m × f) and traversed different attributes in 'a' for grouping instead of submitting a different query. In this way, the number of queries reduced considerably that were getting submitted and accordingly the response time of execution also reduced.

Further, for the furtherance of our experiment, we experimented with combining multiple 'group by' attributes with multiple aggregates. We developed a single general query that contained all possible aggregates, and that grouped by all attributes in 'a'. This, however, caused some problems when extracting the results for the individual values of a column. For instance, we may obtain as a result of grouping the attributes A and B the following two tuples: (1, 2) and (1, 3), where the first number represents an aggregated value for A and the second one represents an aggregated value for B. This has given rise to multiple tuples for the same A value.

To address this, we designed an aggregation plan, which first grouped together the tuples corresponding to a column value and afterward computed a composite aggregate, say max of max or sum of sums. Clarifying a bit more about aggregated average: the counts of each of the categories are summed up and used to calculate the total sum and overall average for each of the categories of 'a'.
Although this approach produced results similar to those with our previous methods, it did have a very large negative impact on performance in terms of execution time. In spite of the fact that the total number of queries was made less, the extensive computation rendered this approach unfavorable.

These results underscore the challenge alluded to by the authors in the summary of problem 4.1 of needing to find an optimal grouping of attribute sets, such that groups can be effectively combined in a way that does not push memory utilization past some point where it begins to cause performance to degrade to non-acceptable levels. For practical implementations, we developed the core of our new database, being an adult table, and further categorized this table as either married or unmarried. This allowed the chance to carry narrow scopes of analysis over marital statuses, hence deriving better intuitions over the impacts of demographics on the attributes in the dataset.

Pruning-Based Optimization

Based on the above pruning optimization techniques, we have implemented the Confidence-Interval based pruning in SeeDB. It approximates a range for the utilities of different views using worst-case statistical confidence intervals. For this, we divided the whole dataset, based on the row numbers, into 20 partitions. For this, we first add all the views and then apply sharing-based optimizations, along with minimizing the database scans of each partition. Utility scores for each view is calculated using K-L Divergence and will be discussed under shared-based optimizations in Section 4.iii.

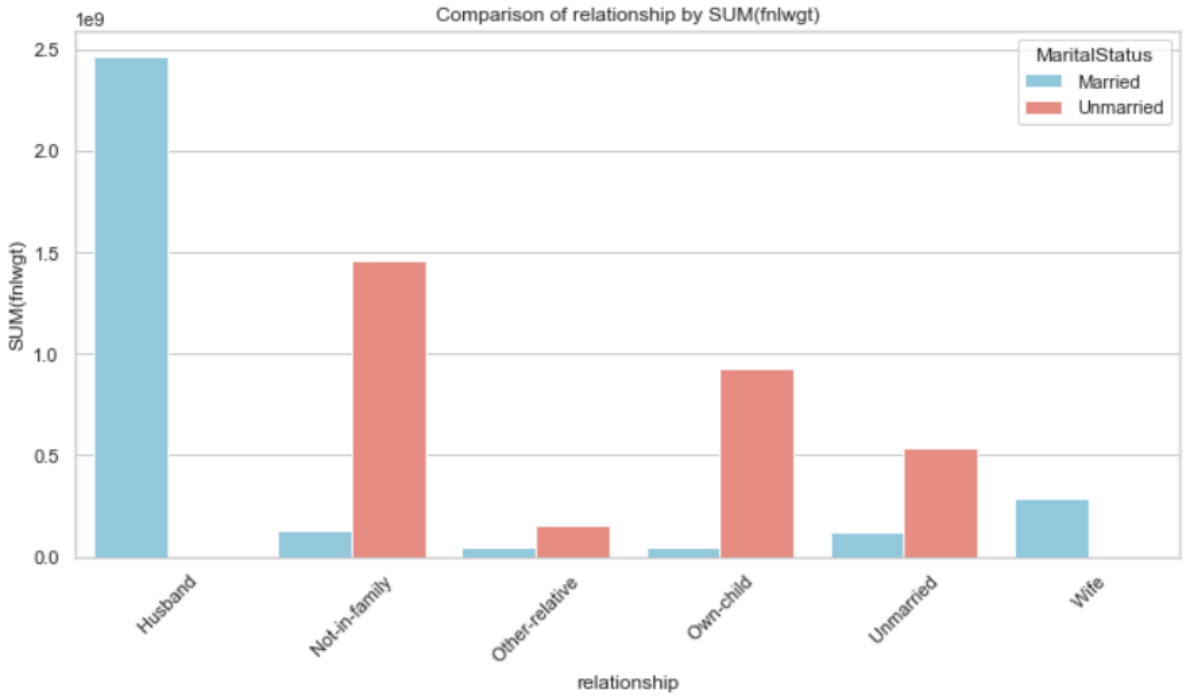
We have performed data processing using a dictionary that has stored each view with current utility scores. A further statistics data structure was maintained to keep records of mean utility scores and the corresponding lower and upper bounds for each of the views after each iteration. The utility scores for the sharing-optimized queries on a partition are appended to the dictionary and the mean utility scores and their confidence intervals are updated in the statistics structure. The confidence intervals are calculated using the Hoeffding-Serfling inequality, with an m = 1 modification that assumes no pruning in the first iteration, since in the absence of confidence intervals, the epsilon term would go towards infinity. This means utility score and confidence intervals are then updated iteratively for each of the views.

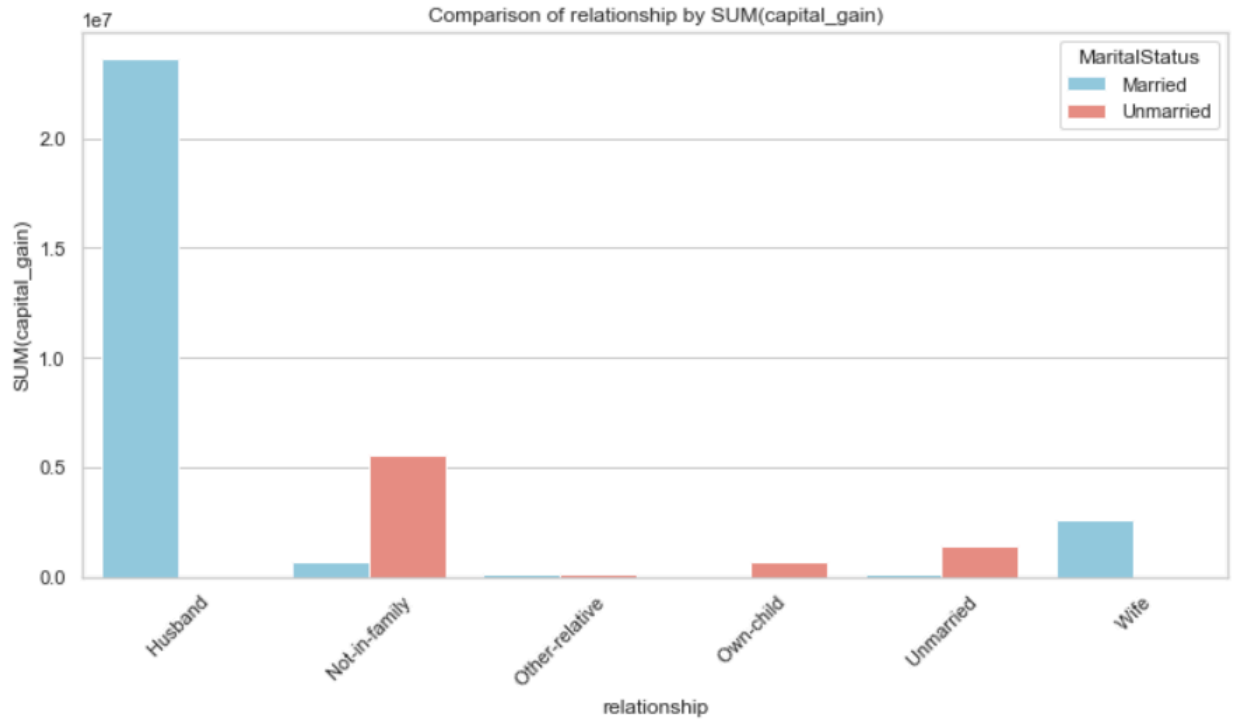
With the exception of the first, after each calculation of utility score, the low-utility visualizations are pruned.The authors explain that this pruning strategy drops any view where the upper utility bound is less than the current iteration lower bound of the 5th highest visualization among the top 5 visualizations in that iteration. This works out to be effective because the authors state, "Very likely that any view for which UB is below the lower of the top 5 will not be a top-5 view". This process of iterative view removal, combined with recomputation of utility scores, mean utilities, and confidence intervals, would slowly peel off the number of views until only the top k views are left after 20 iterations, where k is much less relative to the total number of initial views. For analysis, we first designed a core database table named 'adult'. Further, we subdivided the core table into two other tables: 'married' and 'unmarried', based on marital status, then further analyzed it by comparison and graphics of the effect of demographic attributes on the utility scores of the data set.

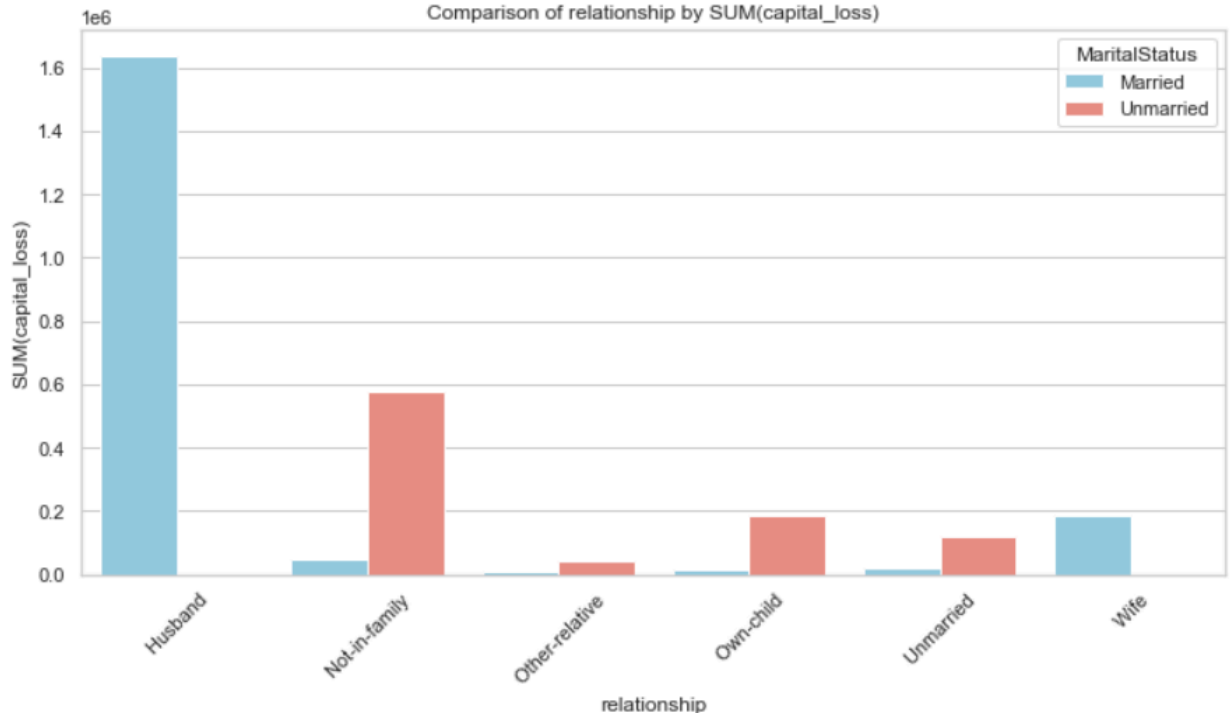
Visualization:

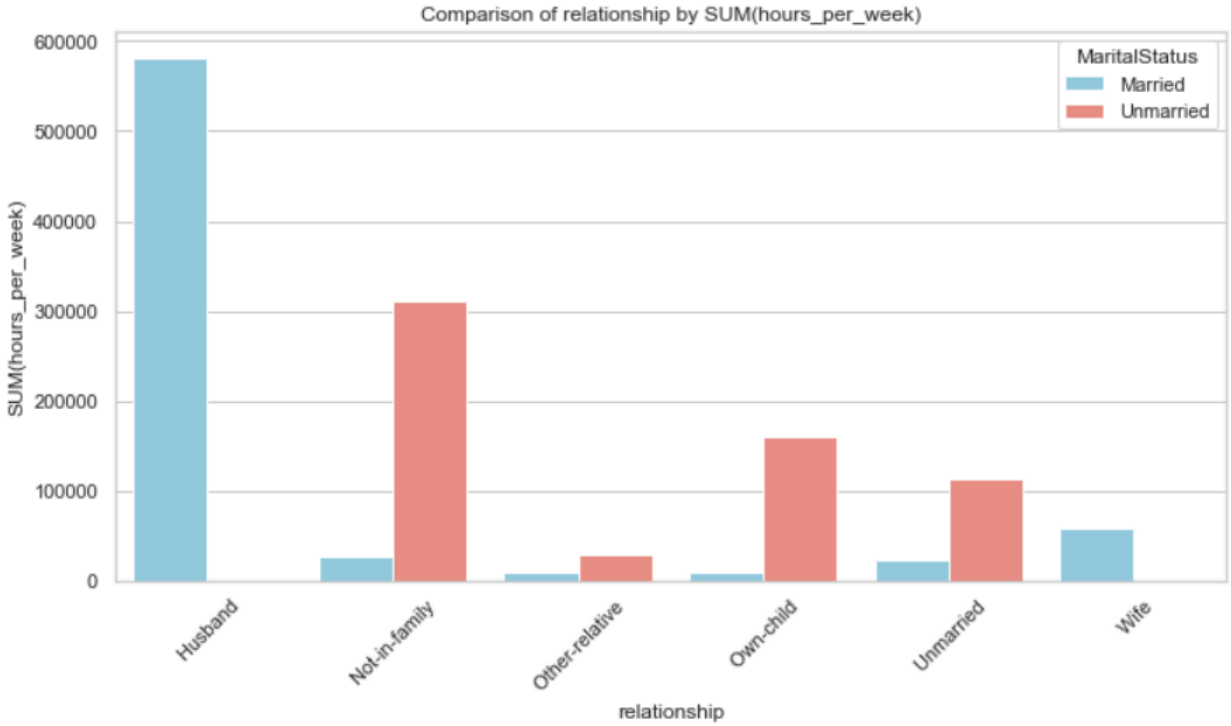
Here we see the top 5 views plotting striking differences between the target data (married adults, displayed in blue) and the reference data (unmarried adults, displayed in green). K-L divergence was then used to quantitatively compare differences between the results obtained by the queries applied to the visualized data. These visualizations

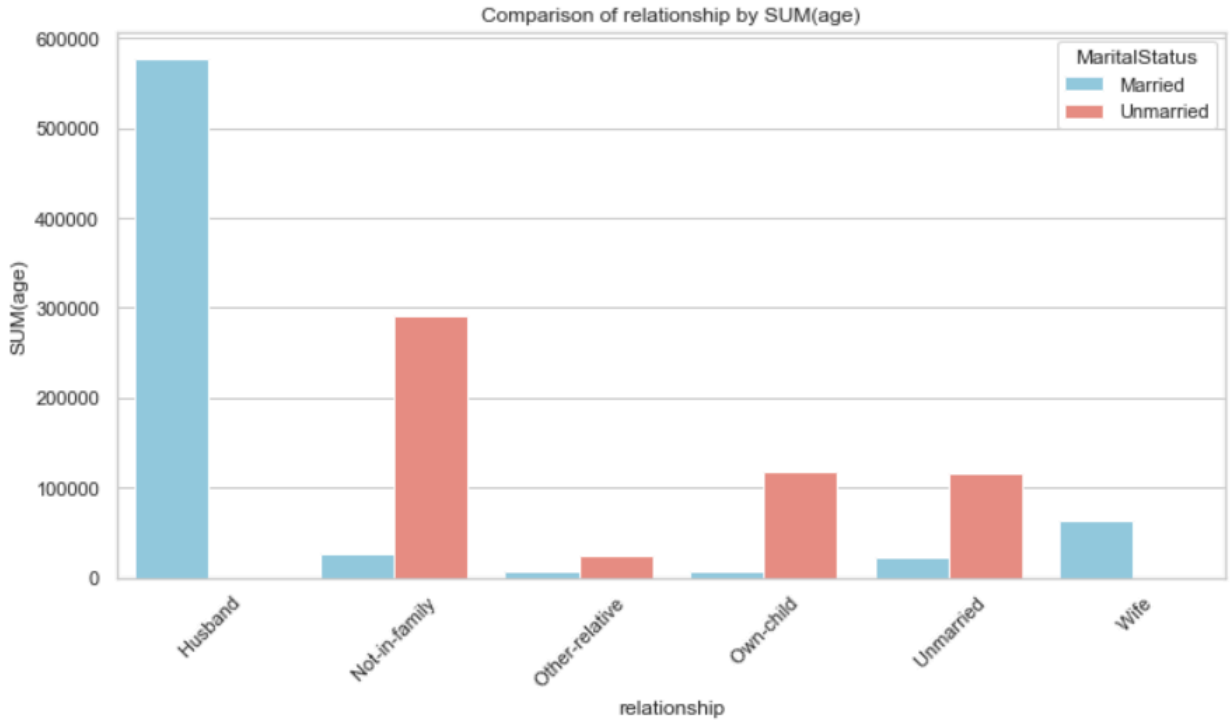
were chosen as the top 5 from all of the visualizations possible. This was because they had high K-L Divergence, which is a measure of entropy. SciPy has calculated that these results have major variations and may be quite informative for the analyst. The following images show the top five visualizations, with the first being the most highly recommended and the fifth being the last in the series.






Comparison of relationship by SUM(capital_loss)
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
SUM(capital_loss)
MaritalStatus
Married
Unmarried
Husband
Not-in-family
Other-relative
Own-child
Unmarried
Wife
relationship


Comparison of relationship by SUM(hours_per_week)
600000
500000
400000
300000
200000
100000
0
SUM(hours_per_week)
MaritalStatus
Married
Unmarried
Husband
Not-in-family
Other-relative
Own-child
Unmarried
Wife
relationship



Conclusion:

We have presented the top 5 aggregate views with respect to K-L Divergence as the utility metric that these methodologies define in SeeDB. This has enabled us to optimally consider a large space of potential visualizations for subsets of data, while avoiding computing with less meaningful visualizations. Our implementation covered Shared-based optimization through query rewriting and resulted in a remarkable decrease in database querying in relation to reduced utilization of shared computational resources. We then exploited the Pruning-based optimizations, which leverages Hoeffding-Serfling inequality, to iteratively prune redundant views after the completion of each phase.