Final Report - SeeDB

Group Members: Akhila Jetty, Sree Nidhi Kanala, Nikita Masanagi

1. Problem Statement

This paper aims at implementing a visualization recommendation engine known as 'SeeDB' that produces promising or interesting visualizations on a subset of data. It serves as a middleware on top of any SQL-compliant DBMS. In order to achieve desired functionality, this project addresses the two major challenges namely, scale - evaluating a large number of visualizations within interactive time scales and utility - finding appropriate metric to assess the interestingness of various visualizations in order to produce/recommend the best visualizations. Two important techniques such as pruning optimizations and sharing optimizations are employed in this project to overcome the challenges mentioned above. Furthermore, this project uses deviation-based metric as a utility measure to evaluate the interestingness of visualizations.

This project focuses on database D with snowflake based schema. It uses two types of attributes namely dimension attributes denoted by A and measure attributes denoted by M. The former attributes to perform group-by operations whereas the latter set of attributes perform aggregation. In addition, there is a set of aggregate functions F over the measure attributes. This idea assumes that D can be grouped along any of the dimension attributes A and aggregated on any of the measure attributes M resulting in a two column table visualization. Furthermore, SeeDB can recommend visualizations on user-specified queries as well, denoted by Q (subset of D).

SeeDB visualization also known as aggregate view or view denoted by Vi is a function represented by a triple (a, m, f), where m ∈ M, a ∈ A, f ∈ F. An aggregate view performs a group-by on a and applies the aggregation function f to measure attribute m. The utility of the visualization is measured using a deviation metric. Interesting visualizations are those Visualizations that show different trends on Query Dataset, denoted by Dq when compared to reference dataset, denoted by Dr. The reference dataset Dr may be defined as the entire underlying dataset (D), the complement of Dq (D - Dq) or data selected by any arbitrary query Q0 (DQ0). View Vi is applied on both Dq (Qt view) and Dr datasets (Qr view) (see Figure 1 and Figure 2) and the results are summarized in two columns a and f(m). These values are normalized using a probability distribution. The distance between these distributions is computed using K-L divergence distance. The higher the distance, the more interesting the visualization is.

$$Q_T = \texttt{SELECT } a, f(m) \texttt{ FROM } D_Q \texttt{ GROUP BY } a$$

Figure 1

$$Q_R = \texttt{SELECT } a, f(m) \texttt{ FROM } D_R \texttt{ GROUP BY } a$$

Figure 2

2.Methodology:

We want to recommend visualizations. There are two things that must be answered. The first one is how to define relevance or utility and how to do it efficiently so that we can scale it to a large number of rows, also avoiding the curse of dimensionality. We want to do it with interactive time scales as visual analytics. We don't want high wait times.

2.1 Visualization Utility

To the question of what makes a visualization interesting , there are many answers like data distribution, schema , the number of distinct values, the query asked to evaluate etc. Utility depends on the distribution of data, the metadata of query, context, aesthetics etc.We will be focusing on data, query and metadata. For example, in barcharts, we can represent the visualizations by V_i= (d:dimension(x-axis), m:measure(y-axis), f:aggregate).We can represent it in sql by using aggregate and group by queries. The sql query would be " SELECT d,f(m) from table GROUP BY d where selection_predicate".We want to rank our visualization based on deviation and surface the ones with highest deviation.We have to make the recommendation efficiently as well.

2.2 Architecture

The architecture is simple and sits on the top of the database. All the data processing is delegated to the Dbms. Let suppose we have d dimensions, m measures and f number of aggregate functions. The number of potential visualizations would be d*m*f. The number of queries issued to the database would be 2*d*m*f. Every query potentially scans the full dataset which is very inefficient and therefore there is a wastage of computation on the views which has low-utility.We mitigate these challenges by using some optimizations that will be explained later in the section.
Our main Architecture f consists of two components:

a. **View generator:**
   View generator generates a list of visualization queries from the input query.First the input query is parsed ,then we query the system metadata.The output is used to generate the list of visualization queries which we can evaluate further for our use case.
b. **Execution engine:**
   Execution engine evaluates the collection of queries using optimizations on top of the underlying DBMS. The selected aggregate views are the recommended visualizations.

2.3 Execution Engine

The execution engine provides the top-k aggregates best suited for the given data evaluation. To identify the top-k best aggregate views, the following steps has to be followed:

a. For each aggregate view, we have to generate a sql query with relation to the target and the reference view distributions.
b. Identify the visualizations with highest utility.

For larger dataset, this implementation can produce high latency. To reduce the latency, two kinds of optimizations are used : Sharing based optimization and pruning based optimization. In sharing based optimization, the aggregate view queries are combined so that the computation will be shared to the

maximum extent and in pruning based optimization, the aggregate view queries related to low utility visualizations are dropped from consideration without scanning the whole dataset. To obtain maximum benefit from both kinds of optimizations, a phased execution framework is developed where each phase is operated on a subset of data.In each phase, sharing and pruning optimization are applied sequentially. The two optimization strategies are explained below:

2.3.1 Sharing based optimization

We focus to minimize the total execution time. We can do this by reducing the total number of queries that will be issued to the database and also reducing the total number of scans of the underlying tables in dbms. We apply the following optimizations:

a. **Combine Multiple Aggregates:** Rewrite all the aggregate queries with the same group-by attribute as a single combined query. For example: if (a1, m1, f1), (a1, m2, f2) . . .(a1, mk, fk) are independent queries ,then we combine them into a single view by (a1, {m1, m2 . . . mk}, {f1, f2 . . . fk}).
b. **Combine Multiple Group-bys:**
   We will be left with a number of queries with a group-by on single aggregate after the application of multiple aggregate optimization. We can combine them into a single query but this will only lead to a slow overall performance as the number of groups increases in size. We want to combine the multiple single attribute group-by queries into multi attribute group-by queries. This is isomorphic to the NP-hard bin-packing problem. In this paper, the authors use the standard first-fit algorithm for finding the optimal grouping of the dimension attributes.

2.3.2 Pruning based optimization

We need pruning methods to save some computational resources by cutting down on low-utility visualisations.At the end of every stage,the execution engine discards some aggregate views using pruning optimizers.The two pruning methods used in SeeDB are the confidence-interval techniques to bound utilities of views and multi-armed bandit allocation strategies to find top-utility views.

a. **Confidence Interval- based pruning**
   This type of method used the worst-case statistical confidence intervals (CI) to bound view utilities. It is similar to top-k based pruning algorithms. This pruning scheme works in the following way-
   We keep an estimate of the mean utility for every aggregate view (Vi) and a confidence interval around that mean.At the end of a phase, if the upper bound of the utility view Vi is less than the lower bound utility of k or more views, then Vi is discarded.
b. **Multi-armed bandit Pruning**
   In Multi-armed Bandit(MAB) pruning, an online algorithm chooses from a set of alternative arms over a sequence to maximize reward.We adapt the Successive Accepts and Rejects algorithm to find the arms with the highest mean reward.At the end of every phase, the views are still being considered are ranked in order of their utility means.

We compute two differences, Δ1 the difference between the highest mean and the k+1st highest mean, and Δn is the difference between the lowest mean and kth highest mean.
If Δ1 is greater than Δn, the view with highest mean is added to the top-k and no longer participates in pruning computations.If Δn is higher, the view with the lowest mean is discarded. This helps to identify the top-k arms with highest probability.

3. Implementation - Main approaches, Data Preprocessing:

3.1 Data preprocessing:

3.1.1 For Census Dataset:

The census dataset has 32661 rows and 15 columns. There were many '?' present in many columns of the dataset. We replaced '?' with average for continuous attributes, mode for categorical attributes over the column values. The '?' in columns: age, education_num, capital_gain, capital_loss integer, hours_per_week in the dataset were replaced with the mean of their respective columns. The '?' in columns workclass, education, marital_status, occupation, relationship, race, sex, native_country, salary_range are replaced with the mode of the respective columns. We dropped column 'fnlwgt' as it is not contributing to good visualizations and might be too complex.

3.1.2 For Dblp Dataset:

There are 4 tables present in the dataset. It consists of four tables: (1) Authors table: contains the authors names, (2) Venue table : contains information about conferences or journals and where they are presented, (3) Papers table: information about the papers is present (4) Paperauths table : author_id and paper_id are associated here.

Step 1: Joining of the tables to create a bigger dataset: These tables are joined by the condition:

- papers.id=paperauths.paperid,
- papers.venue =venue.id

Step 2: We dropped some not usable columns like volume, number, paper_name, venue_name and added some new columns. We have a column called pages which contains the range of page numbers in the volume where the paper is. From this column, we calculated the number of pages and added a new column called page_count. We added a new column called decade whose value is calculated from the years column. The values below 2000 are considered to be "20" and remaining which are above year 2000 to be "21".

Step 3: Replacing Nan Values:

We replaced Nan with average for continuous attributes, mode for categorical attributes over the column values.

3.2 Implementation :

3.2.1 For Census Dataset:

- We have two groups: **1. Married group** : Consists of [Separated, Married-AF-spouse, Married-spouse-absent, Married-civ-spouse] and **2. Unmarried group:** consists of [Never-married, Divorced, Widowed]
- **User-defined Query** - married people
- **Reference Query -** unmarried people
- **Measures -** age, education_num, hours_per_week, capital_gain, capital_loss
- **Dimensions -** workclass, education, occupation, relationship, race, sex, native_country, economic_indicator
- **Functions -** avg, sum, min, max, count
- We implemented sharing based optimization by the combination of multiple aggregates and group-by using KL-Divergence as the utility metric. We implemented a Pruning based method : confidence interval based as our base idea and then implemented multi-armed bandit pruning for this dataset as one of our extension ideas. For confidence interval pruning, we took 95% CI.

3.2.2 For Dblp dataset:

- We have two groups based on the type column: **1. Type_a**: Consists of [0], and **2. Type_b**: consists of [1,3]. We merged the type 1 and type 3 kind of papers into type_b group.
- **User-defined Query** - type_a
- **Reference Query -** type_b
- **Measures -** num_authors, page_count
- **Dimensions -** years, venue, decade
- **Functions -** avg, sum, min, max, count.
- We implemented sharing based optimization by the combination of multiple aggregates and group-by using KL-Divergence as the utility metric. We implemented a Pruning based method : confidence interval based as our extension.

4. Experimental Results for Census dataset:

4.1. Graphs reproduced as per Figure 1 in the paper:

Sample visualization of queries on the dataset.

Visualization 1:

target_view_query_1 = "SELECT sex, avg(capital_gain) as avg_capital_gain FROM married GROUP BY sex;"

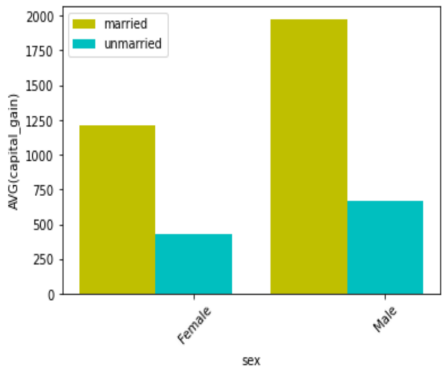
Reference query 1: "SELECT sex, avg(capital_gain) as avg_cap_gain FROM unmarried GROUP BY sex;"

Visulation 2:

target_view_query_2 = "SELECT sex, avg(age) as avg_age FROM married GROUP BY sex;"

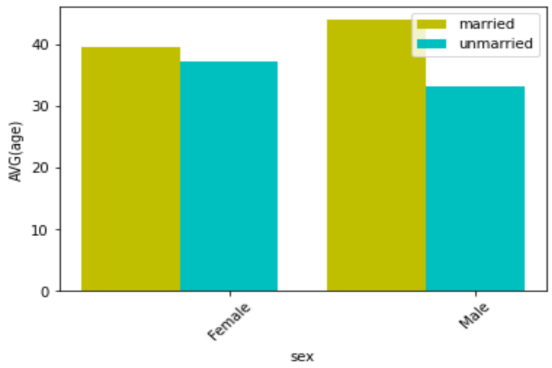
reference_view_query_2 = "SELECT sex, avg(age) as avg_age FROM unmarried GROUP BY sex;"

Dimension - sex, measure - capital_gain, aggregate function - AVG



Visualization 1

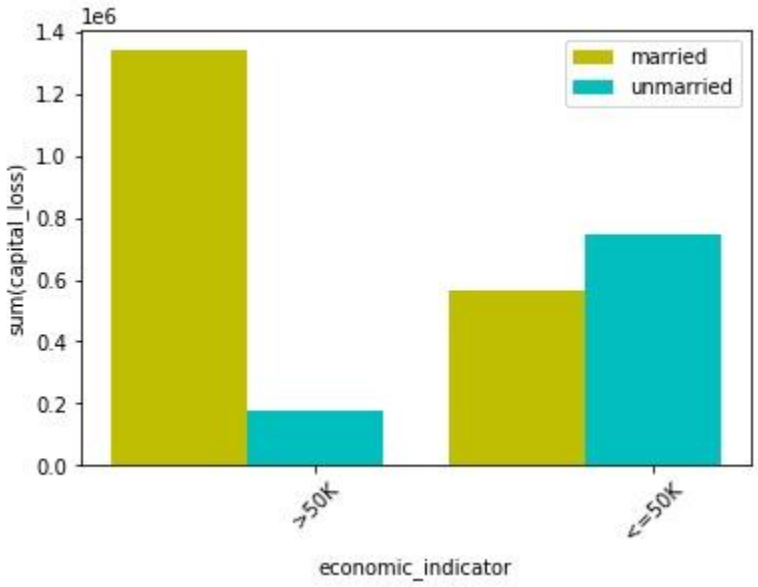
Dimension - sex, measure - age, aggregate function - AVG



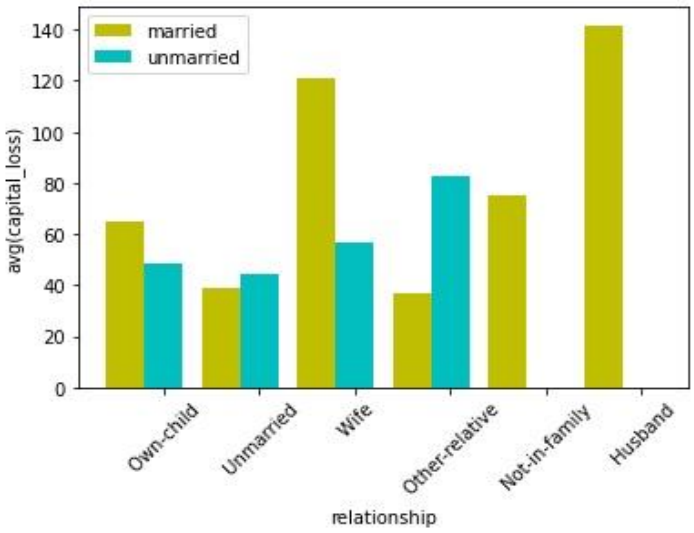
Visualization 4

Produced Top-5 aggregate views and Pruning based optimization Algorithm using Confidence Interval based pruning on Census dataset. The top-5 graphs after phased execution:

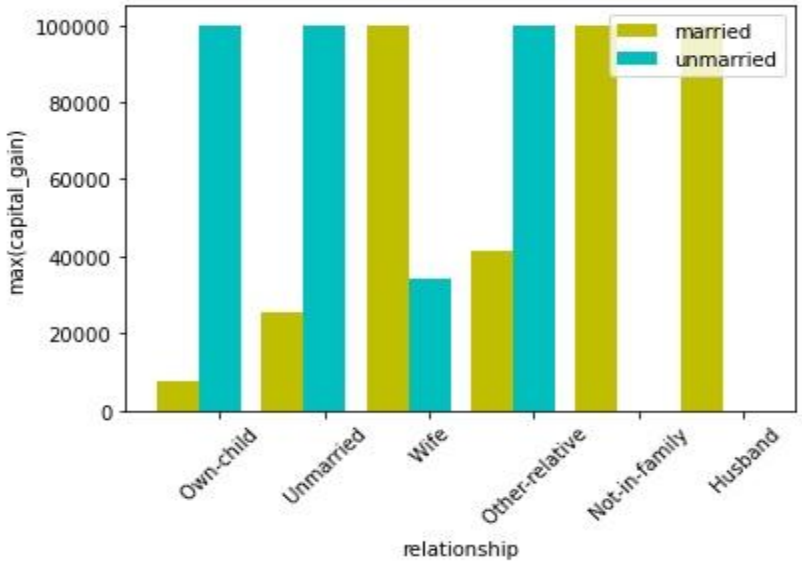
1. **Dimension - economic_indicator, measure - capital_loss, aggregate function - sum**



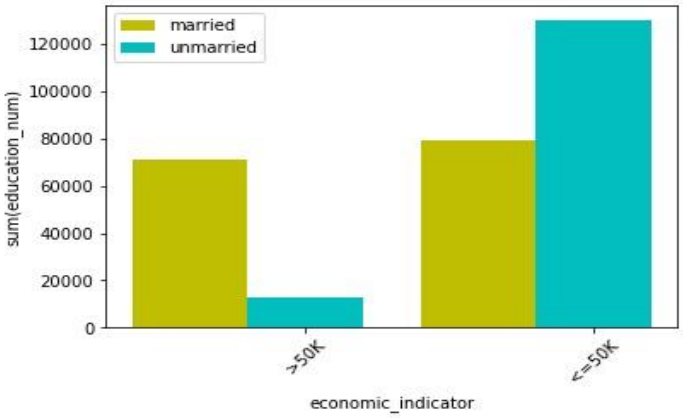
2. **Dimension - relationship, measure - capital_loss, aggregate function - avg**



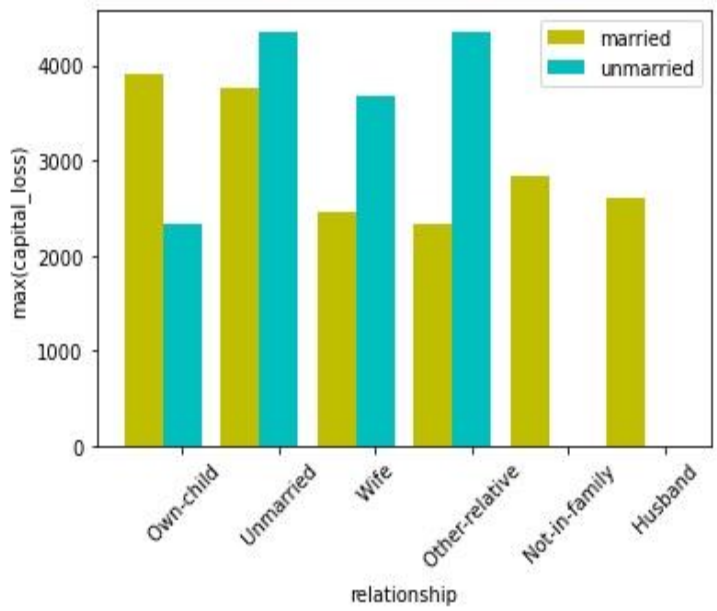
3. **Dimension - relationship, measure - capital_gain, aggregate function - max**



4. **Dimension - economic_indicator, measure - education_num, aggregate function - sum**



5. **Dimension - relationship, measure - capital_loss, aggregate function - max**

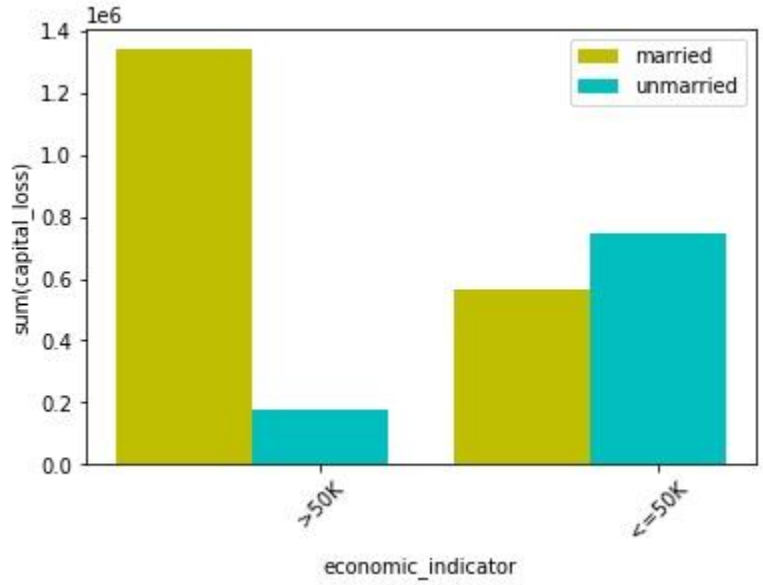


5. Extension and Results :

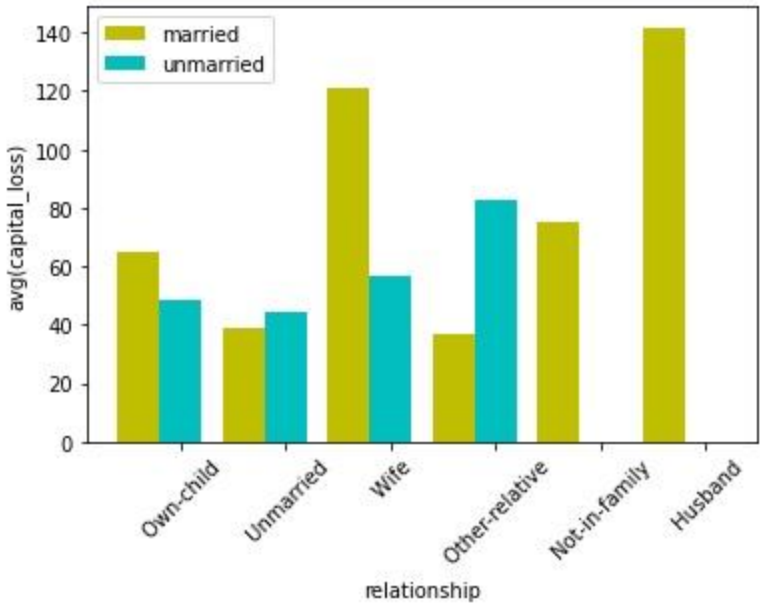
5.1 MAB Pruning:

The top-5 views after implementing MAB pruning:

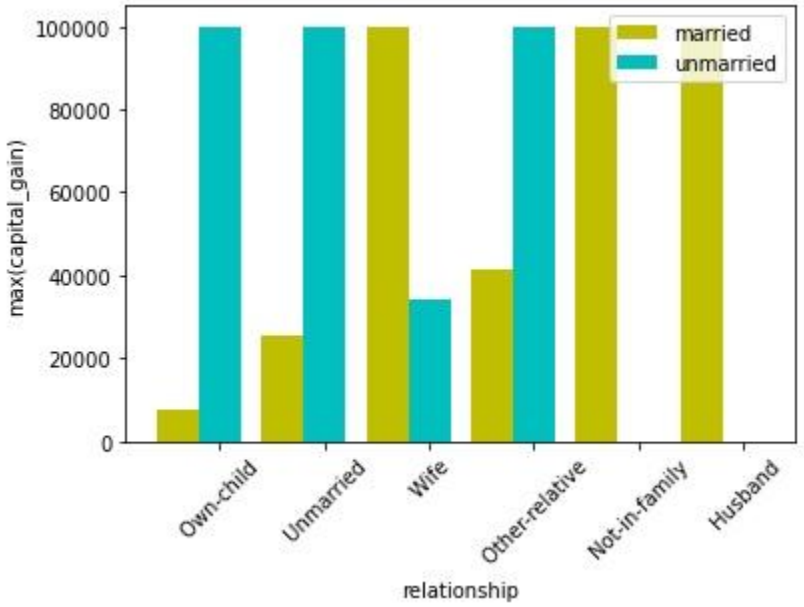
1. **Dimension - economic_indicator, measure - capital_loss, aggregate function - sum**



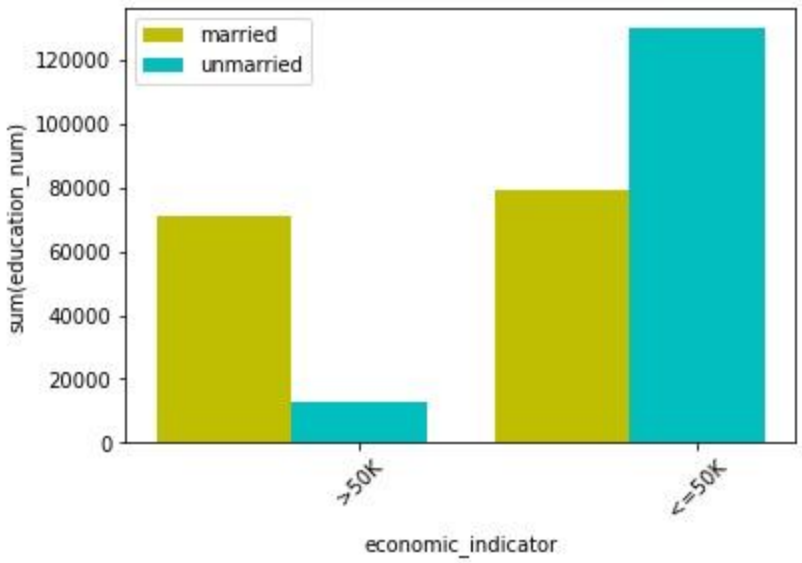
2. **Dimension - relationship, measure - capital_loss, aggregate function - avg**



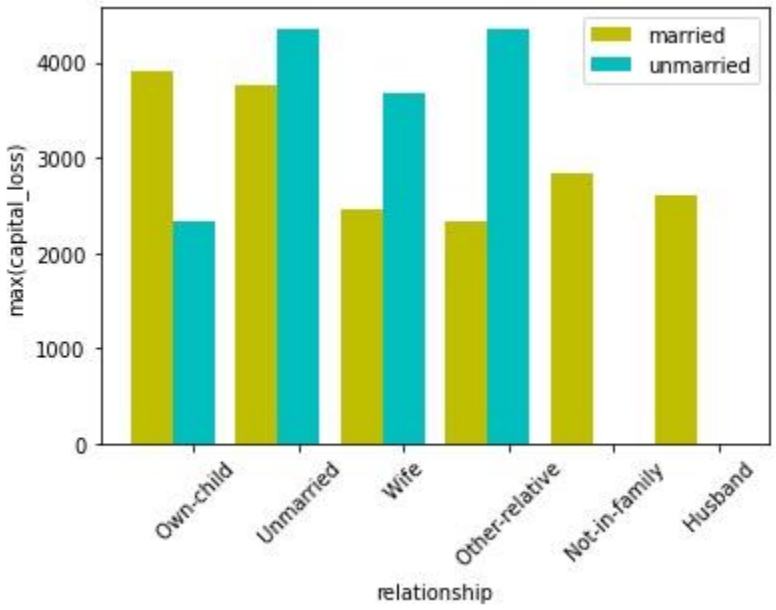
3. **Dimension - relationship, measure - capital_gain, aggregate function - max**



4. **Dimension - economic_indicator, measure - education_num, aggregate function - sum**



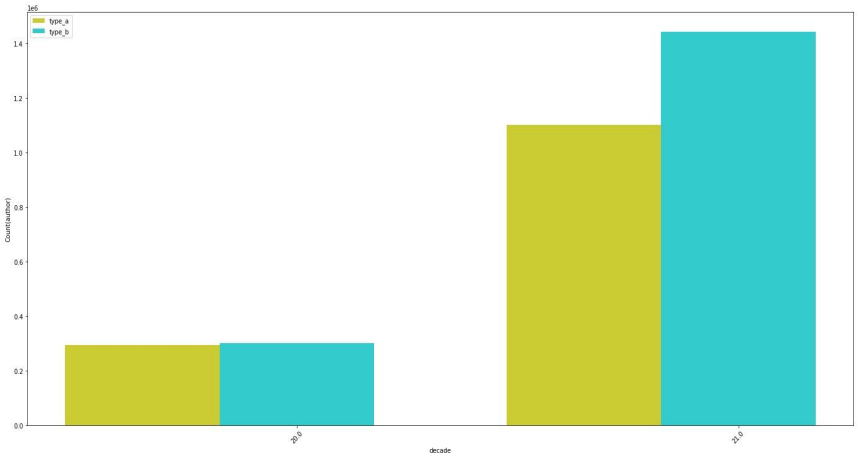
5. **Dimension - relationship, measure - capital_loss, aggregate function - max**



5.2 DBLP dataset:

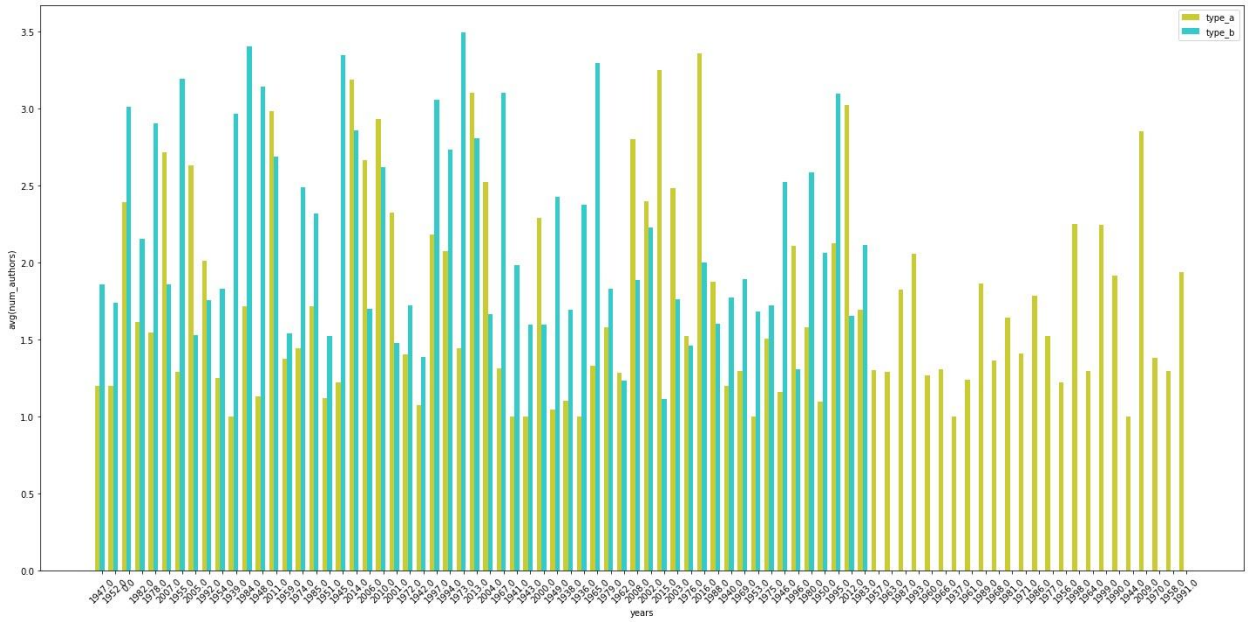
Sample visualization of query :
Target query = "SELECT decade, count(num_authors) FROM type_a GROUP BY decade ;"

Reference query = "SELECT decade,count(num_authors) FROM type_b GROUP BY decade;"
Result :
Dimension - decade, measure - author, aggregate function - Count

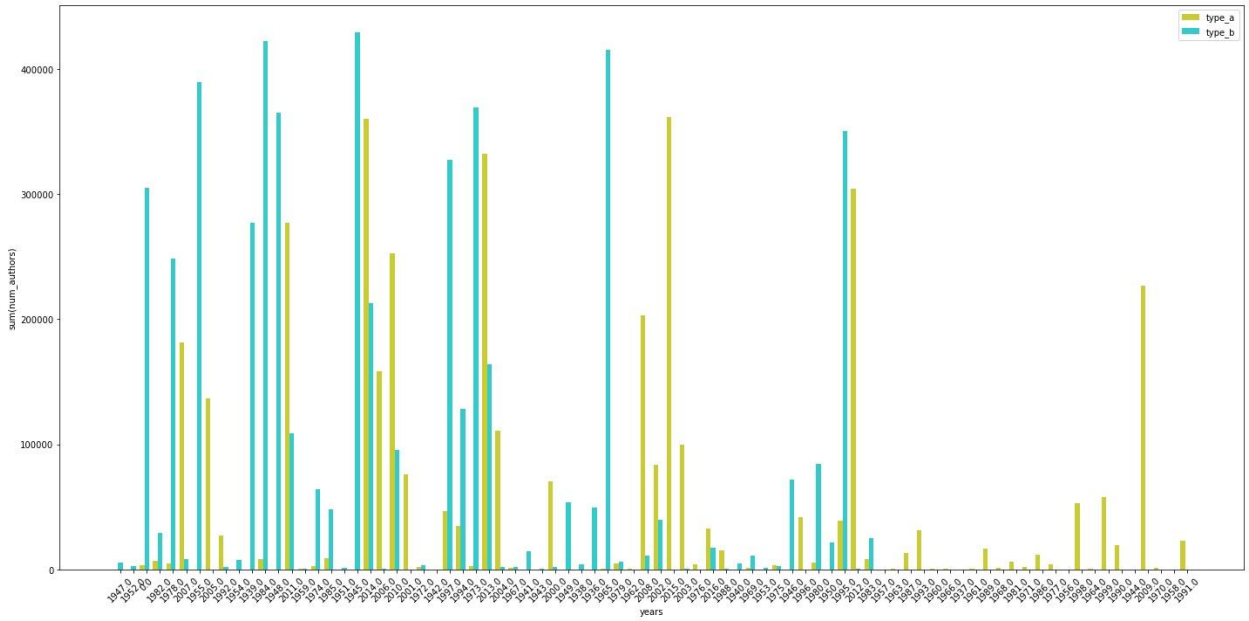


Top K=5 views produced in shared optimization:

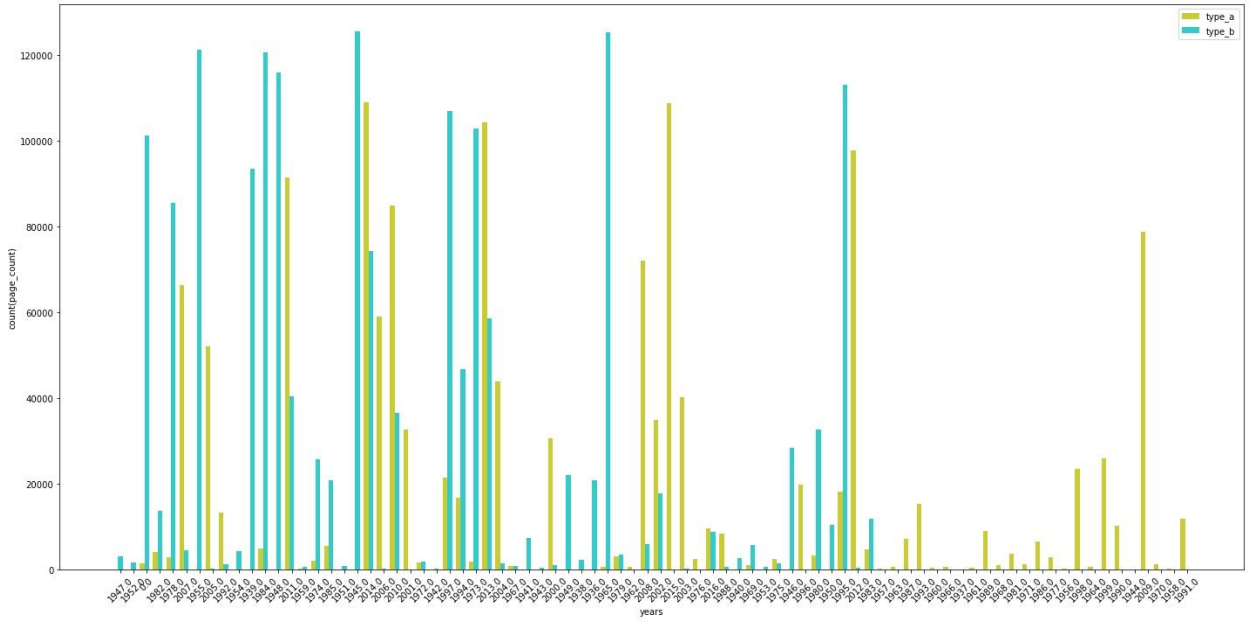
1. **Dimension - years, measure - num_authors, aggregate function - avg**



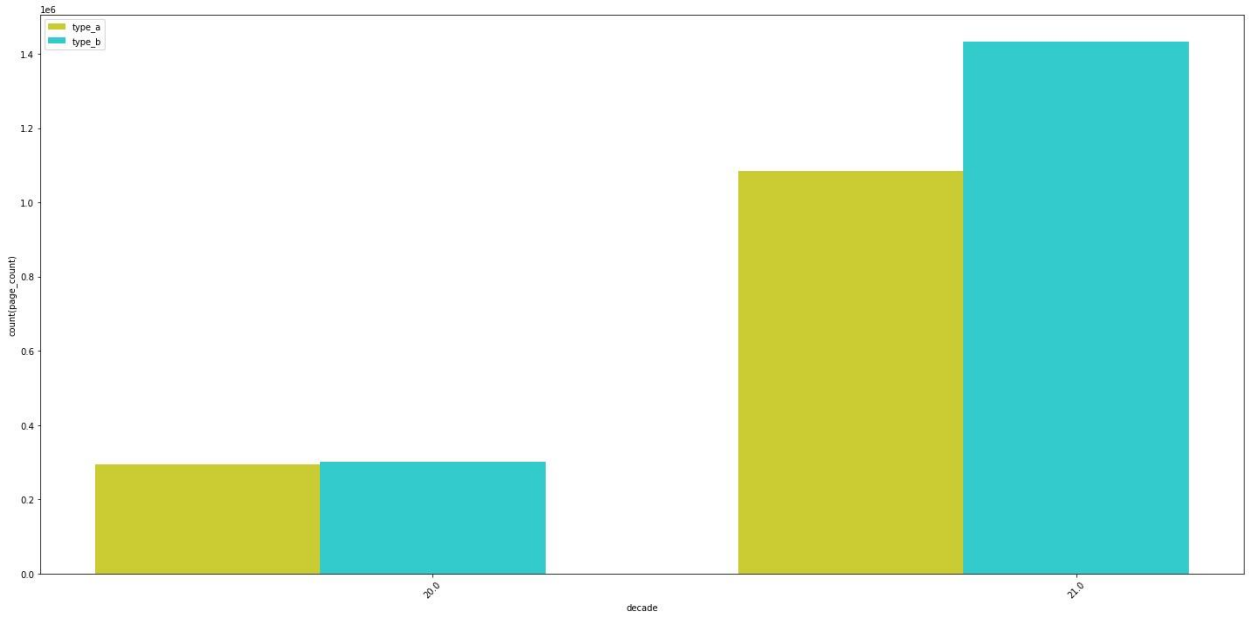
2. **Dimension - years, measure - num_authors, aggregate function - sum**



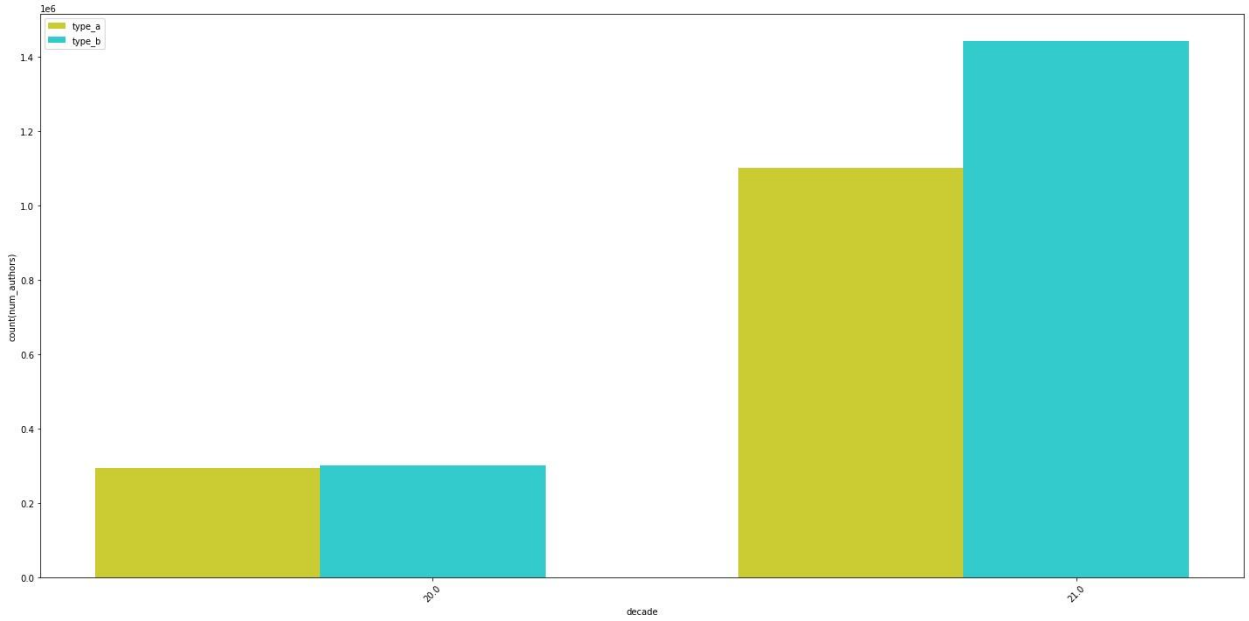
3. **Dimension - years, measure - page_count, aggregate function - count**



4. **Dimension - decade, measure - page_count, aggregate function - count**



5. Dimension - decade, measure - num_authors, aggregate function - count



6. Conclusion

We have presented the results for the top 5 aggregate views, using K-L Divergence as the utility measure, by implementing the algorithms provided in SeeDB, which allows us to explore the vast space of visualizations, for a given data subset, by using pruning optimizations for reducing the rendering or querying of less important/useful visualizations. We have implemented sharing based query optimization for DBLP dataset. We observed that when the number of attributes is less, not many interesting views can be interpreted even if we have large amounts of data. Pruning based optimization was not possible for

DBLP dataset because of limited number of visualizations.There was a large number of missing data in DBLP dataset which has led to some incorrect or irrelevant visualizations.

7. Team Contribution:

1. Akhila Jetty:
    a. Implemented Sharing based optimization for DBLP
    b. Data preprocessing for Census, DBLP Dataset.
    c. Worked on Target, ref query for DBLP
2. Sree Nidhi Kanala:
    a. Implemented Pruning: Confidence Interval.
    b. Data preprocessing for Census Dataset
    c. Worked on shared optimization of Census
3. Nikita Masanagi:
    a. Implemented MAB pruning
    b. Created the Schemas, views and Tables
    c. Consolidated results in report and conclusion