Finding Comparison Insights in Multidimensional Datasets

Claire Antoine[1,†], Alexandre Chanson[1], Nicolas Labroche[1] and Patrick Marcel[2,*,‡]

[1]*LIFAT, université de Tours, France*
[2]*LIFO, Université d'Orléans, France*

Abstract
What are the best comparisons that can be found in a multidimensional dataset? In this paper, we propose to support exploratory data analysis by answering this question, resorting to both input space sampling and output space sampling. We sample both the dataset and the set of aggregate queries that a user would have to ask to find significant comparisons that are frequent in the data cube of the dataset. Our tests show that our approach is effective in retrieving significant comparisons, and that comparison insights can be computed in seconds even for fairly large datasets, if the number of values compared is small.

Keywords
Exploratory data analysis, Comparison queries, Data cube, Statistical tests, Sampling

1. Introduction

Exploratory Data Analysis (EDA) is the notoriously tedious activity of Data Science consisting of interactively analyzing a dataset to gain insights. According to De Bie et al. [1] EDA poses the greatest challenges for automation, since background knowledge and human judgment are the keys to success. Recently, many approaches were proposed to support EDA, including approaches to automatically generate EDA sessions, often defined as maximization problems (see, e.g., [2, 3, 4, 5]).

In this work, we target a specific type of insights, comparisons, that are very popular among data workers (see e.g., [6, 7, 8]). For a given multidimensional dataset R with a categorical attribute A and a numerical attribute M, a comparison insight is noted $a < b$, where a and b are values in the active domain of A, and is such that there are enough evidences that the value of M for $A = a$ is less than the value of M for $A = b$.

Example 1.1. *Consider the excerpt of the flight dataset displayed in Table 1, loaded in a relational database. Suppose the analyst is interested in comparing airlines (the categorical attribute A) to each other regarding the average departure delay (the numerical attribute M). To do so, the analyst could for instance run a SQL query selecting two airlines, say WN and OO (the values a and b), grouping by airline and some other attributes (e.g., departure airport), and aggregating measure departure delay. This implies using a full table scan, which may be prohibitively costly for large tables, and results in a very summarized view of the data, which may hide some "local" effects at more granular level of details. To obtain interesting comparisons, the analyst would have to run enough aggregate queries to check that the same insights for WN and OO are found at various levels of detail.*

departure airport	date	departure hour	airline	departure delay
DCA	1-1-2015	945	WN	-5
LAX	1-1-2015	1951	OO	-4
DCA	1-1-2015	2233	AA	93
SEA	2-1-2015	547	WN	-3
LAX	2-1-2015	2143	OO	63
LAX	3-1-2015	1143	OO	206
LAX	3-1-2015	927	AA	-3
LAX	3-1-2015	1346	OO	-9
SEA	3-1-2015	1654	OO	54
SEA	4-1-2015	155	AA	85

Table 1
Excerpt of table flight

Our goal is to efficiently extract such insights in a given multidimensional dataset. To do so, we resort to sampling the dataset and use existing optimization structures. Resorting to sampling the dataset allows to obtain a candidate comparison insight quickly, using statistical tests to check its significance. Besides, the candidate insights should be validated against the possible group-bys that can be computed over the multidimensional dataset. To validate the insight, we evaluate aggregate queries against the real data, using a sample of queries over existing materialized views.

Our approach to extract comparison insights from a dataset contributes with:

- a robust way of obtaining candidate comparison insights by devising a statistics to choose between a parametric and a non parametric test of significance, based on the distribution of values in the sample of the dataset,
- an algorithm for generating queries over existing materialized views to validate candidate comparison insights,
- a series of experiments on artificial and real datasets to asses the effectiveness and efficiency of the approach.

The outline of the paper is as follows. Next section presents the formal background. Section 3 introduces our approach to extract comparisons. Section 4 details how candidate comparisons are obtained while Section 5 presents the validation of candidates. Experiments are reported in Section 6. Section 7 discusses related work and Section 8 concludes by outlining perspectives.

DOLAP 2025: 27th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2025, March 25, 2025, Barcelona, Spain
*Corresponding author and main contributor.
†The work was done when the author was an intern at LIFAT.
‡This work was partially supported by JUNON Program (DATA/LIFO) with financial support of Région Centre-Val de Loire, France.
claire.antoine56@gmail.com (C. Antoine); Alexandre.Chanson@univ-tours.fr (A. Chanson); Nicolas.Labroche@univ-tours.fr (N. Labroche); Patrick.Marcel@univ-orleans.fr (P. Marcel)
0000-0003-3171-1174 (P. Marcel)

2. Formal background

We give the definition of the comparison queries we consider. This definition extends that of [4, 9] by considering more than one attribute in the group-by set. We consider an instance of relation R of schema $R[A_1, \ldots, A_n, M_1, \ldots, M_m]$. The A_i's are categorical attributes and the M_j's are numerical attributes, called *measures* in what follows. The active domain of attribute A is noted $adom(A)$. As usual for multidimensional data, we require R to be in Boyce-Codd Normal Form.

Definition 2.1 (Comparison queries). *Comparison queries are extended relational queries of the form: $\gamma_{G,agg(M)\rightarrow val}(\sigma_{A=a}(R)) \bowtie \gamma_{G,agg(M)\rightarrow val'}(\sigma_{A=b}(R))$ where A is a categorical attribute in $\{A_1, \ldots, A_n\}$, G is a group-by set in $2^{\{A_1,\ldots,A_n\}}$, M is a measure attribute in $\{M_1, \ldots, M_m\}$, agg is an aggregate function, and $a, b \in adom(A)$. γ denotes the grouping/aggregation operator.*

For a group-by set G of relation R, we call a cuboid of R for function agg and measure M the result of query: $\gamma_{G;agg(M)}(R)$. The cuboids of R form the data cube [10] of R. The number of cuboids for R and $agg(M)$ is $2^{|\{A_1,\ldots,A_n\}|}$, and the number of cuboids for R and $agg(M)$ whose group-by includes attribute A is $2^{|\{A_1,\ldots,A_n\}|-1}$.

For two values a, b of an attribute A for measure M, a comparison insight $a < b$ indicates that, on average, values of M for a are statistically lower than values of M for b. The result of a comparison query can present evidences (if the value of M for a is indeed lower than the value of M for b), or violations (otherwise) of the insight. An actual insight must have enough evidences that support it. If this is not known, we call it a candidate comparison insight.

Definition 2.2 (Candidate comparison insight). *A candidate comparison insight $a < b$ where a, b are in $adom(A)$ for attribute A, measure M of R, aggregation function agg, and instance g of a group-by set G over R including A is a pair of tuples (g, a, m^a), (g, b, m^b) of the result of a comparison query over R such that $m^a < m^b$.*

A candidate comparison insight can be violated depending on the group-by set instance, where a violation of candidate $a < b$ is when $m^a \geq m^b$.

	airline		
departure airport	AA	OO	WN
DCA	11.92	-1.67	8.13
LAX	10.15	3.18	19.09
SEA	6.22	31.0	4.5

Table 2
Cuboid by airline and departure airport (crosstab view) over the flight table

Example 2.3. *Assume Table 2 shows the result of the aggregate query $q = \gamma_{airline,departure\_airport;avg(departure\_delay)}(flight)$ over the flight table of the previous Example. The candidate comparison insight AA> OO holds for DCA and LAX airports, while SEA airport shows a violation of AA> OO. This candidate insight can be obtained from the comparison query: $\gamma_{G,avg(delay)\rightarrow AA}(\sigma_{airline=AA}(flight)) \bowtie \gamma_{G,avg(delay)\rightarrow OO}(\sigma_{airline=OO}(flight))$ where G denotes all the attributes of table flight (see the previous example).*

Definition 2.4 (Validated comparison insight). *Let $a < b$ be a candidate insight, and C be a cuboid over R including attribute A. We say that C supports $a < b$ if the ratio of violations of $a < b$ is below a user defined threshold τ. Finally, we say that the insight is validated if the ratio of cuboids supporting the insight is above a user defined threshold ρ.*

3. Approach overview

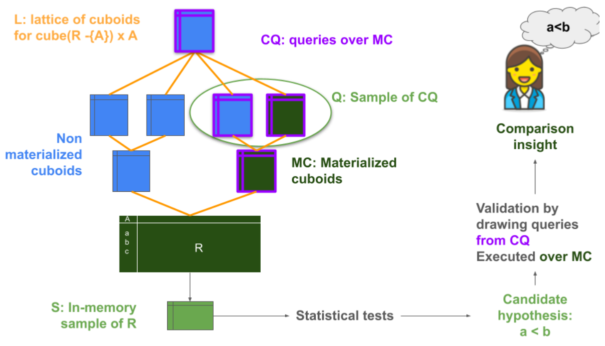

Figure 1: Approach overview

The computation of comparison insights requires to explore the data cube of R which may be too computationally costly if R is large. We developed an approach based on sampling to tackle those cases of discovering comparison insights from large relations. The principle is to compute candidate comparison insights on a sample of R and validate them over a sample of the cuboids of R.

Our approach is illustrated in Figure 1. We first sample R, so that this sample fits in main memory. Let S be the sample of R. Over S, we compute a candidate comparison insight $a < b$, for the comparison query $\gamma_{G,agg(M)\rightarrow val}(\sigma_{A=a}(S)) \bowtie \gamma_{G,agg(M)\rightarrow val'}(\sigma_{A=b}(S))$ where G includes all categorical attributes of R. To check if the comparison is significant, we run a statistical test assessing whether the mean for a is significantly greater than that for b over the sample.

To validate the candidate comparisons found with the statistical tests over S, we sample the set L of cuboids over R as follows. From this set L, we assume that some cuboids are materialized on disk under the form of materialized views (MC, dark green cuboids in Figure 1). From this set MC of materialized views (excluding R), we consider the set CQ of all comparison queries that can be answered using MC (purple bordered in Figure 1). Generally, $|CQ| > |MC|$. We sample this set CQ. This sample is called Q. Over Q, we approximate the ratio of queries of CQ satisfying the user threshold τ. We postulate that this approximation is close enough to the ratio of queries of L satisfying the user threshold.

We now detail these two steps.

4. Statistical test to obtain candidate comparisons

As we do not have all of R but a sample S, the pairwise comparison of a, b is made using a statistical test over S. The null hypothesis of the test should be that the mean for a is greater than that for b. The choice of accepting or rejecting the hypothesis is made based on the p-value of the chosen test. As we will be making many comparisons, the p-values should be corrected, using e.g., the Benjamini-Hochberg FDR correction [11].

We could use a non-parametric test, as in [4] since such tests make few or no assumptions about the data distribution. However, the statistical power of non parametric tests is generally lower and some tests, e.g., permutation tests, require more computation time, as they estimate the distribution by performing numerous re-samplings.

We choose to use the parametric Welch's one-sided t-test that has the advantage of not making assumptions about the variance or sample size, which is well-suited to a context where there is no prior knowledge of the data. However Welch's test still assumes normality of the data and is mainly impacted by skewness and sample size. Indeed, type I error tends to increase with the rise in skewness of the distributions [12].

The question becomes: how to build a decision model based on a dedicated statistic to verify whether the samples used in the comparison are sufficiently normal and decide whether to use Welch's test or a non parametric test. We therefore present our methodology relying on the parametric Welch-t test to devise potentially discriminating candidate statistics (Section 4.1) and then train a simple decision model based on these candidates (Section 4.2).

4.1. Devising candidate statistics

To devise a statistics for deciding which test to use, we considered 8 candidate statistics, devised based on the factors cited in the literature as influencing Welch's test robustness, namely sample size (n) and skewness ($skew$) for each sample, a and b: $(n_a \times n_b), (n_a + n_b), (skew_a \times skew_b), (skew_a + skew_b), |skew_a - skew_b|, \left|\frac{skew_a}{n_a} - \frac{skew_b}{n_b}\right|, (\frac{skew_a}{n_a^2} + \frac{skew_b}{n_b^2}), (\frac{skew_a}{n_a} + \frac{skew_b}{n_b})$.

We generated artificial datasets for a and b with various distributions (standard normal, normal with $\sigma = 20$, uniform, chi-squared, and exponential) and different sample sizes $n \in \{10, 50, 100, 200, 500, 1000\}$. The goal was to obtain a varied panel of pairs (distribution for a, n_a) $\times$ (distribution for b, n_b): distributions with high asymmetry and completely symmetric ones, equal or unequal sample sizes, with large and small size differences. Each sample of the same pair was generated with the same expectation μ, to be under the hypothesis $H_0 : \mu_a = \mu_b (= \mu)$ of Welch's test. If the test rejects H_0, we know it makes a type I error. For each pair considered, we draw 10,000 replicas of different sample pairs, perform Welch's test at the nominal threshold $\alpha = 5\%$, and calculate the rejection rate among the replicas, giving us an estimator of the type I error rate for this given combination. The number of replicas was chosen to be large enough to ensure that the rejection rate is a good estimator of the type I error rate [12, 13]. We also considered several expectations $\mu \in \{2, 5, 10, 100\}$ in case the parameter also impacts the type I error rate.

4.2. Decision model

We want a model that uses only one statistic to classify pairs, to have a quick and simple decision rule: a decision threshold on the statistic in question. We trained two models (decision tree, logistic regression) to classify pairs whose type I error is out of bounds (1) and those for which it is controlled (0) based on each of the candidate statistics. The training sample transmitted to our models was balanced by undersampling, to avoid favoring the majority class.

For each model type, we selected the most interesting statistic (the one creating the greatest heterogeneity for the decision tree, and the one selected by backward selection for logistic regression). We took the highest probability threshold for logistic regression that ensures 0% false negatives on the training sample. In our case, false negatives (not detecting a pair where type I error is out of bounds) are considered more serious errors than false positives (thinking that a pair with controlled type I error is out of bounds) because in the first case, we present a false test result to the user, while in the second case, we postpone the decision to more tests but do not give false conclusions. We compared the two models based on the F_2-score on the rest of the data (excluding the training sample) which provides a more severe estimation of recall, i.e. it maximizes the portion of out-of-bounds pairs detected, which avoids false conclusions.

Both models agree on the following: the best statistics is $\left|\frac{skew_a}{n_a} - \frac{skew_b}{n_b}\right|$ and a threshold of 0.049 achieves the best F_2-score of 0.99 (with no false negatives) to separate valid/invalid pairs on the training sample. Below this threshold, pairs are valid for using Welch's test. On the test sample, the F_2-score is 0.73 (with a recall of 1). We therefore chose to use this statistics and threshold for deciding which test to run.

5. Validating candidate comparisons

The validation of a candidate comparison insight is described in Algorithm 1. Recall from Section 3 that the algorithm validates the candidate insight $a < b$ on a sample of queries Q. Given the sparsity of the data, the sample S may return no data for a or b. If so the candidate is not considered, and the next candidate comparison is checked (line 4). The candidate comes with a p-values indicating if the difference $a < b$ is significant in S. If it is not, the next candidate comparison is checked (line 4). If the candidate is considered and significant, the queries to check violations over the sample Q are generated (line 5), and ran over the materialized cuboids (lines 8). Again, the query result may return no data for a or b. In that case, the cuboid is not counted in the denominator of the prediction (lines 10). If the prediction is such that it is over the threshold (line 15) or such that this threshold can not be reached given the remaining cuboids to check (line 18), then no more validation queries are run and another candidate is checked. Otherwise the algorithm runs until no more queries are left.

Phrased in SQL, the queries generated to check violations of $a < b$ are of the following form:

```
WITH
Q1 AS (SELECT G
      FROM (SELECT G,A FROM view_G
           WHERE A in (a,b) group-by G,A )
      group-by G HAVING count(*) >1),
Q2 AS (SELECT G,A, agg(M) rank () over
        (partition by G ORDER BY agg(M) desc)
      FROM view_G
      WHERE A in (a,b)
```

Algorithm 1 Candidate comparison validation

```
Require: a sample of queries $Q$, two user thresholds $\tau$ and
    $\rho$, a candidate comparison insight $a < b$
Ensure: Valid if $a < b$ is valid, Invalid if it is not, Inconclu-
    sive if $a < b$ is not considered or not significant
 1: $nbQueries = |Q|$
 2: $nbOk = 0$
 3: $nbTest = 0$
 4: if $a < b$ is considered and significant then
 5:     $V$ = generate validation queries over $Q$
 6:     for $q \in V$ do
 7:         $nbTest = nbTest + 1$
 8:         $r$ = evaluate $q$
 9:         if $a$ and $b$ are not in $r$ then
10:             $nbQueries = nbQueries - 1$
11:         end if
12:         if ratio violations in $r < \tau$ then
13:             $nbOk = nbOk + 1$
14:         end if
15:         if $nbOk/nbQueries > \rho$ then
16:             return Valid
17:         end if
18:         if $\frac{nbOk+|Q|-nbTest}{nbQueries} < \rho$ then
19:             return Invalid
20:         end if
21:     end for
22:     if $nbOk/nbQueries > \rho$ then
23:         return Valid
24:     else
25:         return Invalid
26:     end if
27: end if
28: return Inconclusive
```

```
      group-by G,A )
SELECT G,string_agg(A::text,',')
FROM (Q2)
WHERE (G) IN (Q1)
group-by G;
```

where view_G is the materialized view over which the query is evaluated, Q2 is used to order a and b in the result of the comparison query over view_G and Q1 removes group-by instances where a or b do not appear.

Note that Algorithm 1 generates one query per pair (a, b) per element of the sample Q, allowing to leverage existing indexes over attribute A. Another strategy consists of leaving the DBMS to actually compute the ratio of violations of all pairs in a cuboid, by generating one query per element of the sample Q. Phrased in SQL, these queries are of the following form:

```
WITH
Q1 AS (SELECT G,c1.A A1,c2.A A2,
              sign(c1.M-c2.M) sign
      FROM
        (SELECT G,A, agg(M) M
        FROM view_G group-by G,A) c1,
        (SELECT G,A, agg(M) M
        FROM view_G group-by G,A) c2
      WHERE c1.A<c2.A and c1.G=c2.G
),
Q2 AS (SELECT A1, A2, count(*) cnt,
      count(*) filter (WHERE sign=1) pos,
      count(*) filter (WHERE sign=-1) neg
      FROM (Q1)
      group-by A1,A2
)
SELECT A1, A2, (pos-neg)/cnt score
FROM (Q2);
```

We postulate that such queries may not be able to exploit any index and that the self-join between aggregates over the materialized view is likely to be very costly. Therefore, only small datasets may benefit from this strategy.

6. Experiments

This section reports the tests we did to assess the effectiveness and efficiency of our approach.

6.1. Settings

We developed a prototype in Python 3.10 over the Postgres 16 RDBMS[1]. We use the SAMPLE method of Postgres to sample the initial dataset, with the 'SYSTEM_ROWS' parameter[2]. As Postgres does not support query rewriting using materialized view, we implemented a simple rewriting method based only on the list of attributes in the group-by set of the query. Tests are run on a Macbook Pro with Apple M2 Pro chip and 32GB of RAM. Postgres was run with the default configuration, i.e., with 128MB of shared memory buffers and 4MB of working memory.

Dataset	#tuples	Disk sp. (MB)	#cat. attr.	#cub.	Density
F100K	110,358	10	6	32	$4.4.10^{-10}$
F600K	633,938	129	7	64	$5.7.10^{-12}$
SSB	6,001,171	1406	5	16	$2.7.10^{-8}$
Health	12,694,445	2309	6	32	$1.3.10^{-2}$

Table 3
Dataset statistics

Datasets used are in Table 3. F100K and F600K are excerpts from the flight delays dataset[3]. SSB is the artificial dataset of the SSB benchmark [14]. Health is the Rate table from the health insurance dataset[4].

We ran two sets of tests: the first tests compare the result of our approach to ground truth on the small F100K dataset, for the airline attribute. The second set of tests on all datasets reports the time to compute 90 pairs for one attribute: the airline attribute having 14 values (91 pairs in total) for F100k and F600K, the statecode attribute for Health and the lo_suppkey attribute for SSB.

6.2. Effectiveness

Since our approach uses sampling, we ran effectiveness tests to compare its outcome (the comparison insights) to the ground truth, i.e., when comparisons are computed on the non-sampled relation R and all its cuboids. Unless otherwise stated, we arbitrarily fixed $\tau = 0.4$ for the ratio of violations in a cuboid and $\rho = 0.4$ for the ratio of cuboids showing less than 0.4 violations. The percentage of the lattice materialized is fixed to 40%. Note that the materialized cuboids are randomly chosen. Since random sampling is

[1]https://github.com/patrickmarcel/rankInsightGeneration
[2]https://www.postgresql.org/docs/current/tsm-system-rows.html
[3]https://www.kaggle.com/datasets/usdot/flight-delays
[4]https://www.kaggle.com/datasets/hhs/health-insurance-marketplace

used, we run each effectiveness tests 5 times and the results are averaged on the 5 runs.

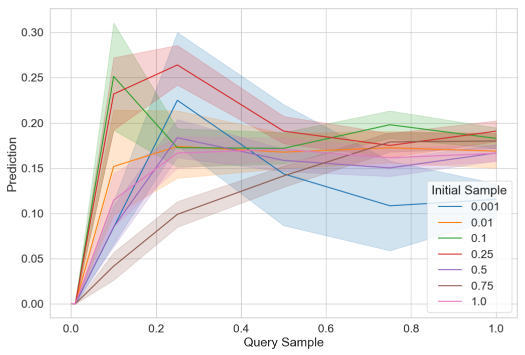

Figure 2: Estimation of the ratio of cuboids over threshold $\tau = 0.4$

We start by showing the average prediction, i.e., the estimation of the ratio of cuboids over the user threshold $\tau = 0.4$ in Figure 2. Both the size of S, the sample of the dataset (each curve), and the size of Q, the sample of the set of aggregate queries over the materialized views (x axis), vary, by changing the sampling ratio in $\{0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1\}$. It can be seen that, except for the very small size of S (0.1% of R), as the query sample grows, all sizes of S converge toward the exact ratio of cuboids.

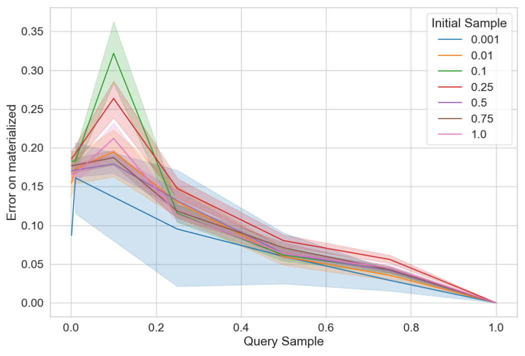

Figure 3: Prediction error over Q compared to CQ

Figure 3 shows the average error (prediction over the sample Q compared to the prediction over all CQ) made by estimating the ratio of cuboids over the user threshold τ, varying the size of the sample of the dataset (each curve) and the size of Q the sample of the set of aggregate queries over the materialized views (x axis) varying the sizes as in the previous test. It can be seen that, except for a very small size, the size of S has little influence on the error, compared to the size of the query sample. This result is good since it shows that the error quickly decreases as the query sample size grows, with an average error below 0.1 with a sample size of 40% and above, even when the size of S is 1% of R. If an error of 0.15 is considered acceptable, then a query sample size of 25% can be used even with 0.1% of the initial dataset.

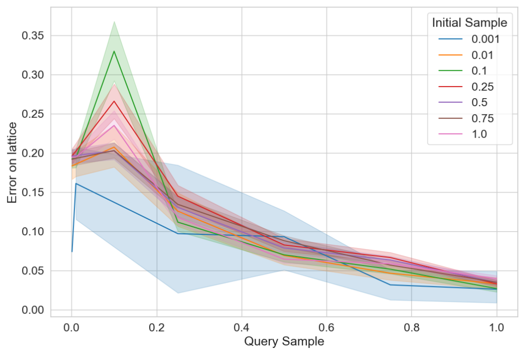

Figure 4: Prediction error over Q compared to L

More interestingly, Figure 4 also shows the average error, but this time as the prediction over the sample Q compared to the prediction over all the lattice L. It can be seen that the error made on L is very close to the error made on CQ, meaning that our approach is a good predictor of the actual number of cuboids of R featuring the comparison insight.

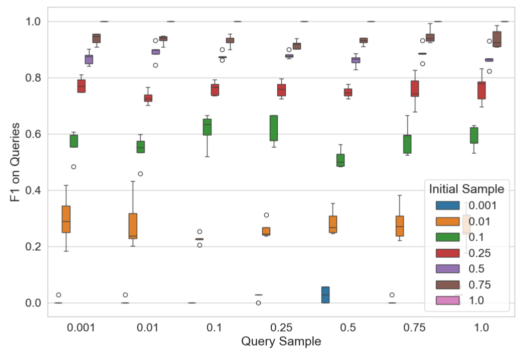

Figure 5: Significant comparisons found on S and Q compared to L (F1 score)

Figure 5 reports the F-measure achieved by our approach compared to the ground truth, i.e., all significant comparisons found (irrespective of the prediction score) by our approach against all the significant comparisons found on L. It can be seen that a sample of R of only 25% is sufficient to achieve a F-measure above 75.

Figure 6 details this result by measuring the Recall @10, i.e., how many of the comparisons having the top-10 prediction scores are found by our approach compared to that found in L. It can be seen that users can expect to find between 30 and 40% of the best comparisons among the first 10 retrieved by our approach, with a sample of half of R and sampling 50% of the queries over the materialized cuboids.

Our last effectiveness test investigates the sensitivity to user parameters τ and ρ. Based on the previous tests, we fixed the initial sample size at 25% of R and the size of query sample at 40% of aggregate queries, varying τ and ρ. Figure 7 shows that ρ does not impact the error, while τ slightly impacts it, since the more violations are tolerated,

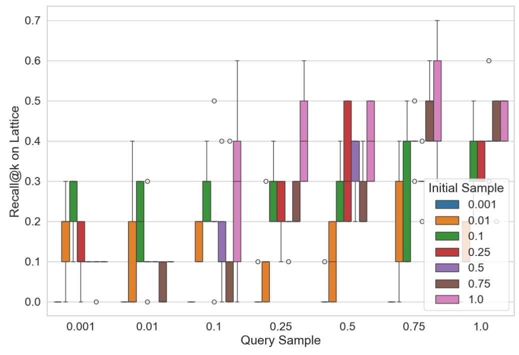

Figure 6: Significant comparisons found on S and Q compared to L (Recall @10)

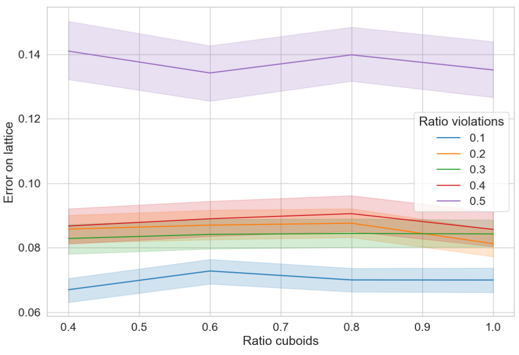

Figure 7: Prediction error over Q compared to L, varying τ and ρ

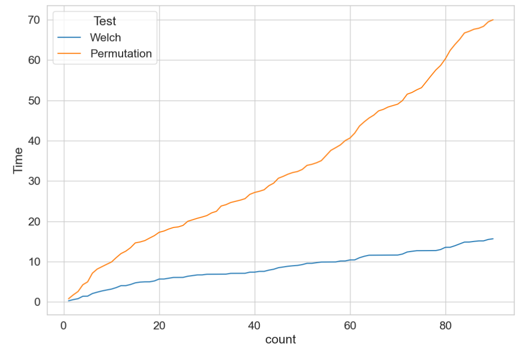

Figure 8: Time (s) for all Welch against all permutations (F100K)

the more impact the sampling of R will have. Other tests, not presented here for lack of space, showed that F-measure and Recall@ 10 are not impacted by these parameters.

6.3. Efficiency

We first check the gain of using the parametric Welch's tests instead of non parametric permutation tests. Figure 8 reports the difference in time (in seconds) when using only Welch's (respectively only permutation) tests on the 90 comparisons (x axis) on F100K, showing that the parametric test is faster by almost an order of magnitude. Figure 9 shows the number of Welch's tests by sample size for the 90 comparisons on F100K. We observe that, expectedly, the larger the sample size, the closer the skewness and size of samples for a and b are, and therefore the more Welch's tests are used. We can conclude that choosing a sample size for S favoring the use of the parametric test allows to save time. Interestingly, for F100K, a sample of only 10% allows to use Welch's tests around 90% of the time.

We next measure the time (in seconds, averaged on 5 runs) it takes to check 90 comparisons (all pairs of the airline attributes in F100K or F600K, 90 pairs of SSB's lo_suppkey, 90 pairs of Health's statecode), using Algorithm 1 in different settings: without index ("False"), with a mono-attribute hash index on the airline (resp., lo_suppkey) attribute ("True"), with a multi-attribute index on all attributes ("mc" for multi-column), with a clustered index on the airline (resp., lo_suppkey) attribute ("cl") and with a multi-attribute index on all attributes when the view is clustered on the airline (resp., lo_suppkey) attribute ("mc-cl"). Note that for SSB and Health, only multi-column indexes were tested. In each case, 40% of the lattice is materialized.

The results are reported in Figures 10, 11 and 12. The time taken is linear in the number of comparisons, and increases with the dataset size and the number of cuboids. This is

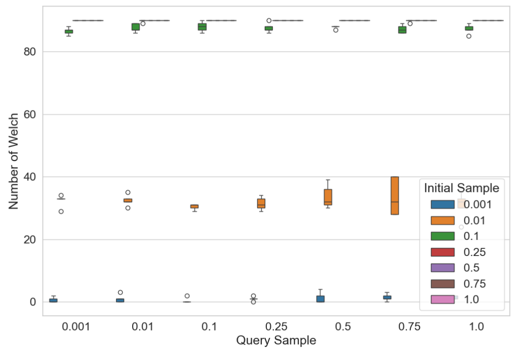

Figure 9: Number of Welch's tests (F100K)

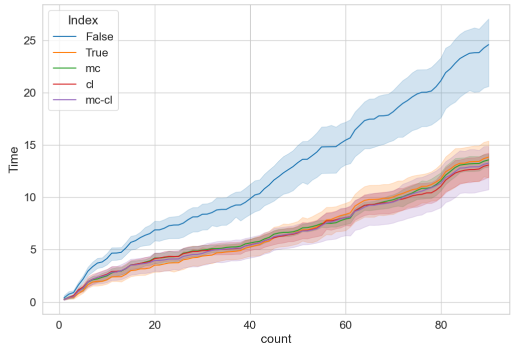

Figure 10: Time (s) by number of comparisons (F100K)

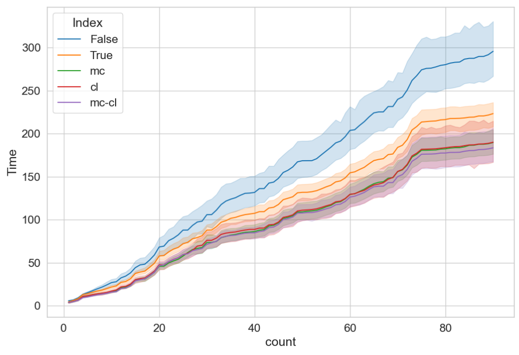

Figure 11: Time (s) by number of comparisons (F600K)

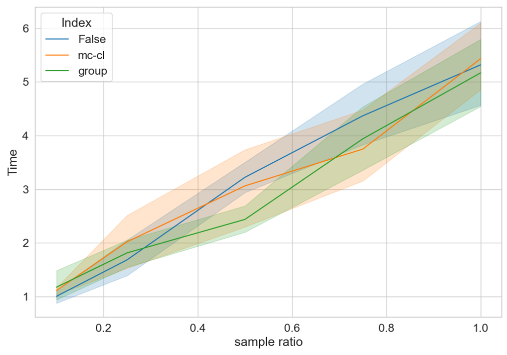

Figure 13: Time (s) by size of Q (F100K)

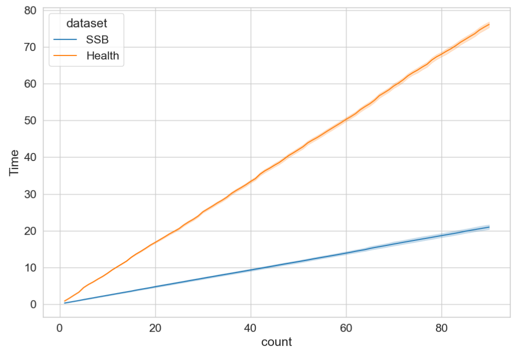

Figure 12: Time (s) by number of comparisons (SSB, Health)

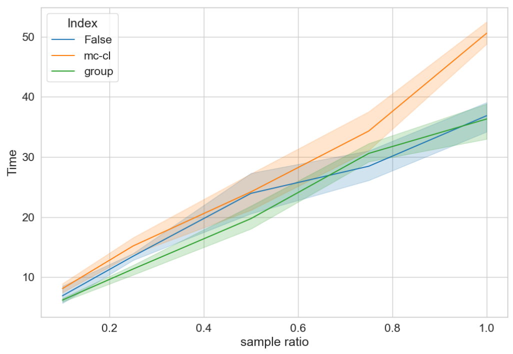

Figure 14: Time (s) by size of Q (F600K)

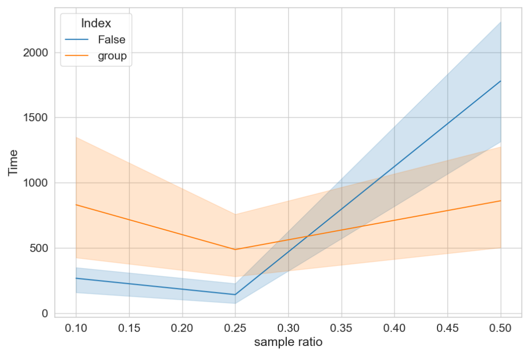

Figure 15: Time (s) by size of Q (SSB)

because sampling R and hypothesis generation are very fast, the cost comes from the evaluation of validation queries and the size of cuboids over which they are evaluated. It takes less than 15 seconds to compute 90 comparison insights over F100K (the smallest dataset), around 20 seconds over SSB (much larger but with only 16 cuboids in its lattice), around 75 seconds on Health (the largest, densest dataset, with number of cuboids between SSB and F600K) and around 3 minutes for F600K (smaller than SSB and Health but with 4 times more cuboids). Expectedly, it can be seen that indexes are useful, with a slight advantage for clustered indexes.

Our last tests aim to assess the alternative strategy that sends one query per element of Q. We measure the time (in seconds) it takes to check all 90 pairs, changing the size of Q. We tested different index configurations: without index ("False"), with a multi-attribute index on all attributes when the view is clustered on the airline (resp., lo_suppkey) attribute ("mc-cl") and with a multi-attribute index on all attributes but the airline (resp., lo_suppkey) attribute ("group"). Results are reported in Figures 13, 14 and 15. As expected, this strategy is beneficial for smaller datasets, with a substantial speedup for both F100K and F600K. For instance, with a sample size of 50%, the time to compute 90 comparisons is less than 4 seconds on F100K while it was more than 10 seconds with a sample size of 40% using the first strategy (Algorithm 1). Indexes and clustering are not useful. On F100K, we also tested different values of parameters τ and ρ, with a positive impact for ρ (since the more cuboids we want, the faster it is to prune cuboids where τ is not satisfied) and no impact for τ, on the computation time. Also as expected, this strategy is very bad for SSB (Figure 15) since the size of its cuboids prevents an efficient self-join.

6.4. Lessons learned

The tests run allow to draw a few suggestions for setting up the parameters of the approach, to achieve good effectiveness and efficiency. Efficiency-wise, it is suggested to opt for Algorithm 1 when the dataset is small, and opt for the second strategy for larger datasets (over 1 GB on a laptop). As to effectiveness, if the goal is to predict the number of cuboids with few violations, $S = 10\%$ of R, $\tau < 20\%$ and $Q = 40\%$ of all aggregate queries should enable to obtain good predictions. If the goal is to check the presence of a given comparison insight, S should be above 25% of R.

7. Related work

Exploratory Data Analysis, insights and Automatic generation of data explorations Exploratory Data Analysis (EDA), the notoriously tedious task of interactively analyzing datasets to gain insights, has attracted a lot of attention both recently [15, 16] and since the early ages of discovery-driven exploration of multidimensional data [17]. Sunita Sarawagi's pioneering work [18, 19, 20] proposed techniques for interactively browsing interesting cells in a data cube. Our approach can actually be seen as complementary to Sunita Sarawagi's DIFF [21], aiming at explaining differences in a multidimensional dataset. In the case of DIFF, the user is supposed to point the comparison to be explained while our approach suggests such comparisons.

EDA has developed beyond analyzing multidimensional datasets with classical OLAP operations like drill-down. EDA operations include data retrieval, data representation, and data mining tasks [22].

One key aspect of supporting EDA is quantifying the importance of insights [2, 6]. It is commonly admitted that interestingness in EDA is manifold [23, 24]. For instance, statistical significance and coverage (importance of the data scope against the entire dataset) are used in [25] and [2].

Approaches for automatically generating EDA sessions can be divided into two categories: *generate and select*, or *guided EDA*. Generate and select methods (see e.g., [25, 2]) generate many, if not all, possible insights and then select the best ones. Guided EDA methods (see e.g., [3, 26]) generate the session as the algorithm explores the search space, mimicking a human analyst.

Comparisons Several studies highlighted the importance of comparisons when analyzing data. Blount et al. [7] examined 67 stories produced using EDA, including award-winning data stories, from both professional journalists and data science-aware students, and found comparisons to be the most popular pattern among novices and professionals alike. Zgraggen et al. [6] define comparison insights as observations, hypotheses, and generalizations directly extracted from data that do not require prior knowledge or domain expertise. The authors designed an experiment where participants explored a synthetic dataset and instructed them to report their reliable insights. 60% of user-reported insights were spurious, which underlines the need for systems to be able to automatically characterize comparison insights.

Francia et al. [27] and, independently, Siddiqui et al. [8] respectively defined the Assess and Compare operators to give a clear semantics and logical foundations of comparisons (and labeling the result of the comparison in [27]) of two series of data.

Since comparisons happen frequently in practice, with a high risk of comparison-based insights being spurious, there is a need to automate the production of non-spurious comparison insights. In that sense our present work can be seen as a follow-up to [4], where we develop a robust input space sampling method (Section 4) and consider output space sampling to validate candidate insights (Section 5).

Note that none of the works mentioned above, in particular [25, 2, 7, 6, 8], can be used as baselines, since they do not propose a method to automatically extract comparison insights like the one proposed here.

Sampling, approximate query answering Approximate query answering techniques [28, 29, 30] are aimed at answering typical aggregation queries much faster than the exact algorithms implemented by DBMS and do so with a bounded error. AQP methods may also rely on common probabilistic bounds such as Hoeffding and Chebyshev [28] to qualify this error. To answer queries faster while remaining accurate AQP methods need to build samples that are accurate stand-in for the complete database. This is especially necessary for group-by query where uniform samples might entirely miss small groups [30]. To avoid this issue AQP method have been designed with many specific biased sampling techniques. One such method is Congressional Sampling [28] that simply ensures small groups are represented by at least n element in the final sample. BlinkDB's [29] set of multi-dimensional, multi-resolution samples are used with a selection strategy which determines automatically the size of the sample based on accuracy or response time constraints. However, AQP advanced sampling strategies while useful for their specific use case of providing accurate results over many different queries, may be an unnecessary cost for our approach. Indeed if samples are too small the statistical testing will simply fail to reject H0 which would already be more likely for smaller groups even in the complete database.

8. Conclusion

We propose an approach to find comparison insights in a multidimensional dataset, by sampling both the dataset and the set of aggregate queries that can be computed over it. Our approach makes use of both input space sampling, since we sample the initial dataset for finding candidate comparison insights, and output space sampling, as we also sample the set of aggregate queries to validate the candidates.

While our results are promising, the approach still needs to be improved both in terms of effectiveness and efficiency. We outline two promising directions. One could use frequent pattern mining to extract frequent comparisons. However, this would require to first compute the datacube of the dataset and then run a frequent pattern algorithm. Sampling patterns could be used [31], and it is part of our future work to compare our approach with frequent pattern sampling approaches. Our future work will also investigate the use of BlinkDB [29], that aims at answering SQL-based aggregation queries over stored data, by building and maintaining a set of multi-dimensional, multi-resolution samples from original data, over time. Its dynamic sample selection strategy determines automatically the size of the sample based on accuracy or response time constraints.

On the longer term, we plan to adapt the approach to other types of insights and other data models.

References

[1] T. D. Bie, L. D. Raedt, et al., Automating data science, Commun. ACM 65 (2022) 76–87.

[2] B. Tang, S. Han, M. L. Yiu, R. Ding, D. Zhang, Extracting top-k insights from multi-dimensional data, in: Proceedings of SIGMOD, Chicago, IL, USA, 2017, pp. 1509–1524.

[3] O. B. El, T. Milo, A. Somech, Automatically generating data exploration sessions using deep reinforcement learning, in: Proceedings of SIGMOD, Portland, OR, USA, 2020, pp. 1527–1537.

[4] A. Chanson, N. Labroche, P. Marcel, S. Rizzi, V. T'kindt, Automatic generation of comparison notebooks for interactive data exploration, in: J. Stoyanovich, J. Teubner, P. Guagliardo, M. Nikolic, A. Pieris, J. Mühlig, F. Özcan, S. Schelter, H. V. Jagadish, M. Zhang (Eds.), Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022, OpenProceedings.org, 2022, pp. 2:274–2:284. URL: https://doi.org/10.48786/edbt.2022.15. doi:10.48786/EDBT.2022.15.

[5] B. Youngmann, S. Amer-Yahia, et al., Guided exploration of data summaries, Proc. VLDB Endow. 15 (2022) 1798–1807.

[6] E. Zgraggen, Z. Zhao, R. C. Zeleznik, T. Kraska, Investigating the effect of the multiple comparisons problem in visual analysis, in: R. L. Mandryk, M. Hancock, M. Perry, A. L. Cox (Eds.), Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018, ACM, 2018, p. 479. URL: https://doi.org/10.1145/3173574.3174053. doi:10.1145/3173574.3174053.

[7] T. Blount, L. Koesten, Y. Zhao, E. Simperl, Understanding the use of narrative patterns by novice data storytellers, in: Proceedings of CHIRA, Budapest, Hungary, 2020, pp. 128–138.

[8] T. Siddiqui, S. Chaudhuri, V. R. Narasayya, COMPARE: accelerating groupwise comparison in relational databases for data analytics, Proceedings of VLDB Endow. 14 (2021) 2419–2431. URL: http://www.vldb.org/pvldb/vol14/p2419-siddiqui.pdf.

[9] A. Chanson, N. Labroche, P. Marcel, V. T'Kindt, Comparison queries generation using mathematical programming for exploratory data analysis, IEEE Transactions on Knowledge and Data Engineering (2024).

[10] J. Gray, A. Bosworth, A. Layman, H. Pirahesh, Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total, in: S. Y. W. Su (Ed.), Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana, USA, IEEE Computer Society, 1996, pp. 152–159. URL: https://doi.org/10.1109/ICDE.1996.492099. doi:10.1109/ICDE.1996.492099.

[11] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate - a practical and powerful approach to multiple testing, J. Royal Statist. Soc., Series B 57 (1995) 289 – 300. doi:10.2307/2346101.

[12] J. Algina, T. Oshima, W.-Y. Lin, Type i error rates for welch's test and james's second-order test under nonnormality and inequality of variance when there are two groups, Journal of Educational Statistics 19 (1994) 275–291.

[13] A. J. Bishara, J. B. Hittner, Testing the significance of a correlation with nonnormal data: comparison of pearson, spearman, transformation, and resampling approaches., Psychological methods 17 (2012) 399.

[14] P. E. O'Neil, E. J. O'Neil, X. Chen, S. Revilak, The star schema benchmark and augmented fact table indexing, in: R. O. Nambiar, M. Poess (Eds.), Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers, volume 5895 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 237–252. URL: https://doi.org/10.1007/978-3-642-10424-4_17. doi:10.1007/978-3-642-10424-4_17.

[15] S. Idreos, O. Papaemmanouil, S. Chaudhuri, Overview of data exploration techniques, in: SIGMOD, 2015, pp. 277–281. URL: https://doi.org/10.1145/2723372.2731084. doi:10.1145/2723372.2731084.

[16] T. Milo, A. Somech, Automating exploratory data analysis via machine learning: An overview, in: SIGMOD, 2020, p. 2617–2622. URL: https://doi.org/10.1145/3318464.3383126. doi:10.1145/3318464.3383126.

[17] S. Sarawagi, R. Agrawal, N. Megiddo, Discovery-driven exploration of OLAP data cubes, in: EDBT, 1998, pp. 168–182.

[18] S. Sarawagi, Explaining differences in multidimensional aggregates, in: VLDB, 1999, pp. 42–53.

[19] S. Sarawagi, User-adaptive exploration of multidimensional data, in: VLDB, 2000, pp. 307–316.

[20] G. Sathe, S. Sarawagi, Intelligent rollups in multidimensional OLAP data, in: VLDB, 2001, pp. 531–540.

[21] S. Sarawagi, idiff: Informative summarization of differences in multidimensional aggregates, Data Min. Knowl. Discov. 5 (2001) 255–276. URL: https://doi.org/10.1023/A:1011494927464. doi:10.1023/A:1011494927464.

[22] T. Milo, A. Somech, Next-step suggestions for modern interactive data analysis platforms, in: SIGKDD, ACM, 2018, pp. 576–585.

[23] L. Geng, H. J. Hamilton, Interestingness measures for data mining: A survey, ACM Comput. Surv. 38 (2006) 9.

[24] P. Marcel, V. Peralta, P. Vassiliadis, A framework for learning cell interestingness from cube explorations, in: ADBIS, 2019, pp. 425–440.

[25] R. Ding, S. Han, Y. Xu, H. Zhang, D. Zhang, QuickInsights: Quick and automatic discovery of insights from multi-dimensional data, in: Proceedings of SIGMOD, Amsterdam, The Netherlands, 2019, pp. 317–332. URL: https://doi.org/10.1145/3299869.3314037. doi:10.1145/3299869.3314037.

[26] A. Personnaz, B. Youngmann, S. Amer-Yahia, Eda4sum: Guided exploration of data summaries, Proc. VLDB Endow. 15 (2022) 3590–3593. URL: https://doi.org/10.14778/3554821.3554851. doi:10.14778/3554821.3554851.

[27] M. Francia, M. Golfarelli, P. Marcel, S. Rizzi, P. Vassiliadis, Assess queries for interactive analysis of data cubes, in: EDBT, 2021, pp. 121–132.

[28] S. Acharya, P. B. Gibbons, V. Poosala, S. Ramaswamy, The aqua approximate query answering system, in: A. Delis, C. Faloutsos, S. Ghandeharizadeh (Eds.), SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA, ACM Press, 1999, pp. 574–576.

[29] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, I. Stoica, Blinkdb: queries with bounded er-

rors and bounded response times on very large data, in: Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013, 2013, pp. 29–42. URL: https://doi.org/10.1145/2465351.2465355. doi:10.1145/2465351.2465355.

[30] A. Galakatos, A. Crotty, E. Zgraggen, C. Binnig, T. Kraska, Revisiting reuse for approximate query processing, Proc. VLDB Endow. 10 (2017) 1142–1153. URL: http://www.vldb.org/pvldb/vol10/p1142-galakatos.pdf. doi:10.14778/3115404.3115418.

[31] A. Giacometti, A. Soulet, Anytime algorithm for frequent pattern outlier detection, Int. J. Data Sci. Anal. 2 (2016) 119–130. URL: https://doi.org/10.1007/s41060-016-0019-9. doi:10.1007/S41060-016-0019-9.