DPXPlain: Privately Explaining Aggregate Query Answers

Yuchao Tao  
Yuchao Tao  
yctao@cs.duke.edu  
Duke University

Amir Gilad  
Amir Gilad  
agilad@cs.duke.edu  
Duke University

Ashwin Machanavajjhala  
Ashwin Machanavajjhala  
ashwin@cs.duke.edu  
Duke University

Sudeepa Roy  
Sudeepa Roy  
sudeepa@cs.duke.edu  
Duke University

ABSTRACT
Differential privacy (DP) is the state-of-the-art and rigorous notion of privacy for answering aggregate database queries while preserving the privacy of sensitive information in the data. In today’s era of data analysis, however, it poses new challenges for users to understand the trends and anomalies observed in the query results: Is the unexpected answer due to the data itself, or is it due to the extra noise that must be added to preserve DP? In the second case, even the observation made by the users on query results may be wrong. In the first case, can we still mine interesting explanations from the sensitive data while protecting its privacy? To address these challenges, we present a three-phase framework DPXPLAIN, which is the first system to the best of our knowledge for explaining group-by aggregate query answers with DP. In its three phases, DPXPLAIN (a) answers a group-by aggregate query with DP, (b) allows users to compare aggregate values of two groups and with high probability assesses whether this comparison holds or is flipped by the DP noise, and (c) eventually provides an explanation table containing the approximately ‘top-k’ explanation predicates along with their relative influences and ranks in the form of confidence intervals, while guaranteeing DP in all steps. We perform an extensive experimental analysis of DPXPLAIN with multiple use-cases on real and synthetic data showing that DPXPLAIN efficiently provides insightful explanations with good accuracy and utility.

1 INTRODUCTION

Differential privacy (DP) [14, 40–42] is the gold standard for protecting privacy in query processing and is critically important for sensitive data analysis. It has been widely adopted by organizations like the U.S. Census Bureau [3, 38, 58, 84] and companies like Google [44, 97], Microsoft [29], and Apple [89]. The core idea behind DP is that a query answer on the original database cannot be distinguished from the same query answer on a slightly different database. This is usually achieved by adding random noise to the query answer to create a small distortion in the answer. Recent works have made significant advances in the usability of DP, allowing for complex query support [32, 56, 59, 69, 90, 97], and employing DP in different settings [32, 45, 48, 77, 90, 99]. These works assist in bridging the gaps between the functionality of non-DP databases and databases that employ DP.

Automatically generating meaningful explanations for query answers in response to questions asked by users is an important step in data analysis that can significantly reduce human efforts and assist users. Explanations help users validate query results, understand trends and anomalies, and make decisions about next steps regarding data processing and analysis, thereby facilitating data-driven decision making. Several approaches for explaining aggregate and non-aggregate query answers have been proposed in database research, including intervention [81, 82, 98], Shapley values [67], counterbalance [75], (augmented) provenance [5, 65], responsibility [73, 74], and entropy [43] (discussed in Section 6).

One major gap that remains wide open is to provide explanations for analyzing query answers from sensitive data under DP. Several new challenges arise from this need. First, in DP, the (aggregate) query answers shown to users are distorted due to the noise that must be added for preserving privacy, so the explanations need to separate the contributions of the noise from the data. Second, even after removing the effect of noise, new techniques have to be developed to provide explanations based on the sensitive data and measure their effects. For instance, standard explanation methods in non-DP settings are typically deterministic, while it is known that DP methods must be randomized. Therefore, no deterministic explanations can be provided, and even no deterministic scores or ranks of explanations can be displayed in response to user questions if we want to guarantee DP in the explanation system. Third, the system needs to ensure that the returned explanations, scores, and ranks still have high accuracy while being private.

In this paper, we propose DPXPLAIN, a novel three-phase framework that generates explanations under DP for aggregate queries based on the notion of intervention [82, 98]. DPXPLAIN surmounts the aforementioned challenges and is the first system combining DP and explanations to the best of our knowledge. We illustrate DPXPLAIN through an example.

Example 1.1. Consider the Adult (a subset of Census) dataset [35] with 48,842 tuples. We consider the following attributes: age, workclass, education, marital-status, occupation, relationship, race, sex, native-country,
Figure 1: Database instance and the three phases of the DPXPlain framework.

And high-income, where high-income is a binary attribute indicating whether the income of a person is above 50K or not; some relevant columns are illustrated in Figure 1a.

In the first phase (Phase-1) of DPXPlain, the user submits a query and gets the results as shown in Figure 1b. This query is asking the fraction of people with high income in each marital-status group. As Figure 1b shows, the framework returns the answer with two columns: group and Priv-answer. Here group corresponds to the group-by attribute marital-status. However, since the data is private, instead of seeing the actual aggregate values avg-high-income, the user sees a perturbed answer Priv-answer for each group as output by some differentially private mechanism with a given privacy budget (here computed by the Gaussian mechanism with privacy budget \( \rho = 0.1 \) [14]).

The third column True-answer shown in grey (hidden for users) in Figure 1b shows the true aggregated output for each group.

In the second phase (Phase-2) of DPXPlain, the user selects two groups to compare their aggregate values and asks for explanations. However, unlike standard explanation frameworks [43, 65, 75, 82, 98] where the answers to a query are correct and hence the question asked by the user is also correct, in the DP setting, the answers that the user sees are perturbed. Therefore, the user question and the direction of comparison may not be valid. Hence our system first tests the validity of the question. If the question is valid, our system provides a data-dependent explanation of the user question. We explain this below with the running example.

First, consider the question in Figure 2 comparing the last two groups in Figure 1b (spouse in armed forces vs. civil-ian). In this example, even though the noisy avg-high-income for "Married-AF-spouse" is larger than the noisy value for "Married-civ-spouse", this might not be true in the real data (as is the case in the True-answer column). Hence, our system tests whether the user question could potentially be explained just using the noise introduced by DP rather than from the data itself. To do this, our system tests the validity of the user question by computing a confidence interval around the difference between these two outputs. In this case, the confidence interval is (−0.259, 0.460). Since it includes 0 and negative values, we cannot conclude with high probability that "Married-AF-support" > "Married-civ-spouse" is true in the original data. Since the validity of the user question is uncertain, we know that any further explanation might not be meaningful and the user may choose to stop here. In other words, the explanation for the comparison in the user question is primarily attributed to the added noise by the DP mechanism. If the user chooses to proceed to the next phase for further explanations from the data, they might not be meaningful.

Now consider the comparison between two other groups "Never-married" and "Married-civ-spouse", in Figure 1c. In this case, the confidence interval around the difference does not include 0 and is tight around a positive number of 0.4, which indicates that the user question is correct with high probability. Notice that it is still possible for a valid question to have a confidence interval that includes zero given sufficiently large noise. Since the question is valid, the user may continue to the next phase.

In the third phase (Phase-3) of DPXPlain, for the questions that are likely to be valid, DPXPlain can provide a further detailed data-dependent explanation for the question. To achieve this again with DP, our framework reports an "Explanation Table" to the user.
user as Figure 1d shows, which includes the top-5 explanation predicates. The explanation predicates explain the user question using the notion of intervention as done in previous work [82, 98] for explaining aggregate queries in the non-DP setting. Intuitively, if we intervene in the database by (hypothetically) removing tuples that satisfy the predicate, and re-evaluate the query, then the difference in the aggregate values of the two groups mentioned in the question will reduce. In the simplest form, explanation predicates are singleton predicates of the form “attribute = <value>”, while in general, our framework supports more complex predicates involving conjunction, disjunction, and comparison (>, ≥ etc.). In Figure 1d, the top-5 simple explanation predicates, as computed by DPXPlain, are shown out of 103 singleton predicates, according to their influences on the question but perturbed by noises to satisfy DP. The amount of noise is proportional to the sensitivity of the influence function, the maximum possible change of the influence of any explanation predicate when adding or removing a single tuple from the database. Once the top-5 predicates are selected, the explanation table also shows both their relative influence (intuitively, how much they affect the difference of the group aggregates in the question) and their ranks (that might be far away from the true top-5) in the form of confidence interval under DP.

From this table, occupation = “Exec-managerial” is returned as the top explanation predicate, indicating that the people with this job contribute more to the average high income of the married group compared to the never-married group. In other words, managers tend to earn more if they are married than those who are single, which probably can be attributed to the intuition that married people might be older and have more seniority, which is consistent with the third explanation age = ”(40, 50]” in Figure 1d as well. Although these explanations are chosen at random, we observe that the first three explanations are almost constantly included. This is consistent with the narrow confidence interval of rank for the first three explanation predicates, which are all around [1, 8]. Looking at the confidence intervals of the relative influence and ranks in the explanation table, the user also knows that the first three explanations are likely to have some effect on the difference between the married and unmarried groups.

However, for the last two explanations, the confidence intervals of influences are closer to 0 and the confidence intervals of ranks are wider, especially for the fifth one which includes negative influences in the interval and has a wide range of possible ranks (96 out of 103 simple explanation predicates in total).

Our contributions.

- We design a low sensitivity influence function inspired by previous work on non-private explanations [98], which is the key to the accurate selection of the top-k explanation predicates.
- We design an algorithm that uses a noisy binary search technique to find the confidence intervals of the explanation ranks, which overcomes the high sensitivity challenge of the rank function.
- We have implemented a prototype of DPXPlain [2] to evaluate our approach. We include two case studies on a real and a synthetic dataset showing the entire process and the obtained explanations. We have further performed a comprehensive accuracy and performance evaluation, showing that DPXPlain correctly indicates the validity of the question with 100% accuracy for 8 out of 10 questions, selects at least 80% of the true top-5 explanation predicates correctly for 8 out of 10 questions, and generates descriptions about their influences and ranks with high accuracy.

2 PRELIMINARIES

We now give the necessary background for our model. The DPXPlain framework supports single-block SELECT - FROM - WHERE - GROUP BY queries with aggregates (Figure 3) on single tables. Hence the database schema \( \mathcal{A} = (A_1, \ldots, A_N) \) is a vector of attributes of a single relational table. Each attribute \( A_i \) is associated with a domain \( \text{dom}(A_i) \), which can be continuous or categorical. A database (instance) \( D \) over a schema \( \mathcal{A} \) is a bag of tuples (duplicate tuples are allowed) \( t_i = (a_1, \ldots, a_m) \), where \( a_i \in \text{dom}(A_i) \) for all \( i \). The domain of a tuple is denoted as \( \text{dom}(A_i) = \text{dom}(A_j) \times \cdots \times \text{dom}(A_m) \). We denote \( A_i^{\text{max}} = \max\{|a| \mid a \in \text{dom}(A_i)\} \) as the maximum absolute value of \( A_i \). The value of the attribute \( A_i \) of tuple \( t \) is denoted by \( t.A_i \).

Figure 3: Group-by query with aggregates supported by DPXPlain. The true results are denoted by \( (a_i, a_t) \) and the noisy results released by a DP mechanism are denoted by \( (a_t, \delta_t) \) where \( a_t \) is the value of \( A_{gb} \) and \( \delta_t \) are aggregate values. We consider group-by aggregate queries \( q \) of the form shown in Figure 3. Here \( A_{gb} \) is the group-by attribute and \( A_{agg} \) is the aggregate attribute, \( \phi \) is a predicate without subqueries, and \( \text{agg} \in \{\text{COUNT}, \text{SUM}, \text{AVG}\} \) is the aggregate function. When query \( q \) is evaluated on database \( D \), its result is a set of tuples \( (a_i, a_t) \), where \( a_i \in \text{dom}(A_{gb}) \) and \( a_t = \text{agg}(t.A_{agg} \mid t \in D, \phi(t) = \text{true}, t.A_{gb} = a_t) \). For brevity, we will use \( \phi'(D) \) to denote \( \{t \mid \phi'(t) = \text{true}\} \) for any predicate \( \phi' \), and \( \text{agg}(A_{agg}, D') \), or simply \( \text{agg}(D') \) when it is clear from context, to denote \( \text{agg}(t.A_{agg} \mid t \in D') \) for any \( D' \subseteq D \). Hence, \( a_t \) is calculated from \( A_{agg} \) and \( A_{gb} \) for any \( D' \subseteq D \). The example schema is \( \mathcal{A} = \{\text{marital-status, occupation, age, relationship, race, workclass, sex, native-country, education, high-income}\} \). All the attributes are categorical attributes and the domain of

\[ q = \text{SELECT} \ A_{gb}, \ \text{agg}(A_{agg}) \ \text{FROM} \ D \ \text{WHERE} \ \phi \ \text{GROUP BY} \ A_{gb}; \]
we use Gaussian mechanism to release where \( D \) we use Gaussian noise to perturb query answers and derive confidentiality. DP allows users to query the database without compromising the privacy by guaranteeing that the query result will not differ too much (defined in the sequel) even if it is evaluated on any two different but neighboring databases defined below.

**Definition 2.2 (Neighboring Database).** Two databases \( D \) and \( D' \) are neighboring (denoted by \( D \approx D' \)) if \( D' \) can be transformed from \( D \) by adding or removing a tuple in \( D \).

In this paper, we consider a relaxation of DP called \( \rho \)-zero-concentrated differential privacy (zCDP) [14, 42] for several reasons, and refer to it simply as DP if not otherwise stated. First, we use Gaussian noise to perturb query answers and derive confidence intervals, which does not satisfy pure \( \varepsilon \)-DP [41] but satisfies approximate \((\varepsilon, \delta)\)-DP [41] and \( \rho \)-zCDP. Second, \( \rho \)-zCDP only has one parameter \( \rho \), compared to \((\varepsilon, \delta)\)-DP which has two parameters, so it is easier to understand and control. Third, \( \rho \)-zCDP allows for tighter analyses for tracking the privacy budget (controlled by \( \rho \)) over multiple private releases, which is the case for this framework. A lower \( \rho \) value implies a lower privacy loss.

**Definition 2.3 (Zero-Concentrated Differential Privacy (zCDP) [14]).** A mechanism \( M \) is said to be \( \rho \)-zero-concentrated differential private, or \( \rho \)-zCDP for short, if for any neighboring datasets \( D \) and \( D' \) and all \( \alpha \in (1, \infty) \) it holds that

\[
D_\alpha(D) \leq \rho \alpha
\]

where \( D_\alpha(D) \) denotes the R\( \text{\c{e}nyi} \) divergence of the distribution \( D \) from the distribution \( D' \) at order \( \alpha \) [76].

A popular approach for providing \( z \)-CDP to a query result is to add Gaussian noise to the result before releasing it to a user. This approach is called Gaussian mechanism [14, 41].

**Definition 2.4 (Gaussian Mechanism).** Given a query \( q \) and a noise scale \( \sigma \), Gaussian mechanism \( M_C \) is given as:

\[
M_C(D, q, \sigma) = q(D) + N(0, \sigma^2)
\]

where \( N(0, \sigma^2) \) is a random variable from a normal distribution with mean zero and variance \( \sigma^2 \).

**Example 2.5.** Suppose there is a database \( D \) with 100 tuples. Consider a query \( q = \text{"SELECT COUNT(*) FROM } D \text{"}, which counts the total number of tuples in a database \( D \). Here \( q(D) = 100 \). Now we use Gaussian mechanism to release \( q(D) \), which is to randomly sample a noise \( z \) from distribution \( N(0, \sigma^2) \). Here we assume \( \sigma = 1 \).

Finally, we got a noisy result \( \hat{q}(D) = 102.32 \), which we may round to an integer in postprocessing without sacrificing the privacy guarantee (Proposition 2.9 below).

The privacy guarantee from the Gaussian mechanism depends on both the noise scale it uses and the sensitivity of the query. Query sensitivity reflects how sensitive the query is to the change of the input. More noise is needed for a more sensitive query to achieve the same level of privacy protection.

**Definition 2.6 (Sensitivity).** Given a scalar query \( q \) that outputs a single number, its sensitivity is defined as:

\[
\Delta_q = \sup_{D \approx D'} |q(D) - q(D')|
\]

**Example 2.7.** Continuing Example 2.5, since the query \( q \) returns the database size, for any two neighboring databases, their sizes always differ by 1, so the sensitivity of \( q \) is 1.

**Theorem 2.8 (Gaussian Mechanism [14]).** Given a query \( q \) with sensitivity \( \Delta_q \) and a noise scale \( \sigma \), its Gaussian mechanism \( M_C \) satisfies \((\Delta_q^2/2\sigma^2)\)-zCDP. Equivalently, given a privacy budget \( \rho \), choosing \( \sigma = \Delta_q/\sqrt{2\rho} \) in Gaussian mechanism satisfies \( \rho \)-zCDP.

**Composition Rules.** In our analysis, we will use the following standard composition rules and other known results from the literature of DP [72] (in particular, zCDP [14]) frequently:

- **Parallel composition:** if mechanisms take disjoint data as input, the total privacy loss is the maximum privacy loss from each.
- **Sequential composition:** if mechanisms take overlapping data as input, the total privacy loss is the sum of each privacy loss.
- **Post-processing:** if we run a mechanism and post-process the result without accessing the data, the total privacy loss is only the privacy loss from the mechanism.

**Private Query Answering.** Recall that we have group-by aggregation query of the form \( q = \text{SELECT } a_{gb} \) agg \( (a_{agg}) \) FROM \( D \) WHERE \( \phi \) GROUP BY \( a_{gb} \), and it returns a list of tuples \( (a_1, a_1) \) where \( a_1 \in \text{dom}(a_{gb}) \) and \( a_1 \) is the corresponding aggregate value. Since no single tuple can exist in more than one group, adding or removing a single tuple can at most change the result of a single group. As mentioned earlier, Phase-1 returns noisy aggregate values \( \hat{a}_1 \) for each \( a_1 \) instead of \( a_1 \). The following holds:

**Observation 2.1.** According to the parallel composition rule (Proposition 2.9), if for each \( a_1 \), its (noisy) aggregate value \( \hat{a}_1 \) is released under \( \rho_q \)-zCDP, the entire release of results including all groups \( (a_1, \hat{a}_1 : a_1 \in \text{dom}(a_{gb})) \) satisfies \( \rho_q \)-zCDP.

For a COUNT or SUM query, we use the Gaussian mechanism for each group \( a_1 \): \( \hat{a}_1 = a_1 + N(0, \sigma^2) \), where the noise scale \( \sigma = \Delta_q/\sqrt{2\rho_q} \) to satisfy \( \rho_q \)-zCDP by Theorem 2.8. The sensitivity term \( \Delta_q = 1 \) for COUNT and \( \Delta_{max} \) for SUM, the maximum absolute value of the aggregation attribute in its domain. For an AVG query, since \( AVG = SUM/COUNT \), we decompose it into a SUM and a COUNT query, privately answer each of them by half of the privacy budget \( \rho_q/2 \) to get \( \hat{a}_1^C \) and \( \hat{a}_1^O \) for each group \( a_1 \), and release \( \hat{a}_1 = \hat{a}_1^C + \hat{a}_1^O \) as a post-processing step. The noisy query answers of the group-by query with AVG satisfy \( \rho_q \)-zCDP by the sequential composition rule (Proposition 2.9).
Confidence Level and Interval. Confidence intervals are commonly used to determine the error margin in uncertain computations and are used in various fields including machine learning [55] and DP [46]. In our context, we use confidence intervals to measure the uncertainty in the user question and our explanations.

**Definition 2.10 (Confidence Level and Interval [96]).** Given a confidence level $\gamma$ and an unknown but fixed parameter $\theta$, a random interval $I = (I^L, I^U)$ is said to be its confidence interval, or CI, with confidence level $\gamma$ if the following holds:

$$Pr[I^L < \theta < I^U] \geq \gamma$$

**Example 2.11.** Let $\theta = 0$. Suppose with probability 50% we have $I^L = -1$ and $I^U = 1$, and with another probability 50% we have $I^L = 1$ and $I^U = 2$. Therefore, $Pr[I^L \leq \theta \leq I^U] = 50\%$, and we can conclude that the random interval $I = (I^L, I^U)$ is a 50\% confidence interval for $\theta$.

3 PRIVATE EXPLANATIONS IN DPXPLAIN

In this section, we provide the model for private explanations of query results at the center of DPXPLAIN.

**User Question and Standard Explanation Framework.** In Phase-2 of DPXPLAIN, given the noisy results of a group-by-aggregation query from Phase-1, users can ask questions comparing the aggregate values of two groups:

**Definition 3.1 (User Question).** Given a database $D$, a group-by-aggregation query $q$ as shown in Figure 3, a DP mechanism $M$, and two noisy answer tuples $(a_i, \hat{o}_i), (a_j, \hat{o}_j) \in M(D,q)$ where $\hat{o}_i > \hat{o}_j$, a user question has the form "why is the (noisy) aggregate value $\hat{o}_i$ of group $a_i$ larger than the aggregate value $\hat{o}_j$ of group $a_j$?", which is denoted by "why ("Married-civ-spouse", 'Never-married', >)?."

**Example 3.2.** The question from Figure 1c is denoted as "why ("Married-civ-spouse", 'Never-married', >)?".

To explain a user question, several previous approaches return top-k predicates that have the highest influences over the group difference in the question [43, 65, 82, 98]. We follow this paradigm and define explanation predicates.

**Definition 3.3 (Explanation Predicate).** Given a database $D$ with a set of attributes $A$, a group-by-aggregation query $q$ (Figure 3) with group-by attribute $A_{gb}$ and aggregate attribute $A_{agg}$ and a predicate size $l$, an explanation predicate $p$ is a Boolean expression of the form $p = \varphi_1 \wedge \cdots \wedge \varphi_l$, where each $\varphi_i$ has the form $A_l = a_l$ such that $A_l \in A \setminus \{A_{gb}, A_{agg}\}$ is an attribute, and $a_l \in \text{dom}(A_l)$ is its value.

We assume $\text{dom}(A_l)$ is discrete, finite, and data-independent. We focus here on the conjunction of equality predicates. However, our framework can also handle predicates that contain disjunctions and inequalities of the form $A_l \circ a_l$ where $\circ \in \{>, \leq, \geq\}$ when the constant $a_l$ is from a finite and data-independent set.

**New challenges for explanations with DP.** Unlike standard explanation framework on aggregate queries [65, 82, 98], the existing frameworks are not sufficient to support DP and need to be adapted: (i) the question itself might not be valid due to the noise injected into the queries, (ii) the selection of top-k explanation predicates needs to satisfy DP, which further requires the influence function to have low sensitivity so that the selection is less perturbed, and (iii) since the selected explanation predicates are not guaranteed to be the true top-k, it is also necessary to output extra descriptions under DP for each selected explanation predicate about their actual influences and ranks. We detail the adjustments as follows.

**Question Validation with DP (Phase-2).** While the user is asking "why is $\hat{o}_i > \hat{o}_j$?", in reality, it may be the case that the true results satisfy $o_i \leq o_j$, i.e., they have opposite relationship than the one observed by the user. This indicates that $\hat{o}_i > \hat{o}_j$ is the result of the noise being added to the results. In this scenario, one option to explain the user’s observation of $\hat{o}_i > \hat{o}_j$ will be releasing the true values (equivalently, the added exact noise values), which will violate DP. Instead, to provide an explanation in such scenarios, we generate a confidence interval for the difference of two (hidden) aggregate values $o_i - o_j$, which can include negative values (discussed in detail in Section 4.1). This leads to the first problem we need to solve in the DPXPLAIN framework:

**Problem 1 (Private Confidence Interval of Question).** Given a dataset $D$, a query $q$, a DP mechanism $M$, a privacy budget $\rho$, a confidence level $\gamma$, and a user question $(\alpha, \alpha, >)$ on the noisy query answers output by $M$ satisfying $\rho_2$-zCDP, find a confidence interval (see Definition 2.10) for the user question $I_{uq} = (I_{uq}^L, I_{uq}^U)$ for $o_i - o_j$ at confidence level $\gamma$ without extra privacy cost.

In Phase-2, the framework returns a confidence interval of $o_i - o_j$ to the user. If it includes zero or negative numbers, it is possible that $o_i \leq o_j$, and the user’s observation of $\hat{o}_i > \hat{o}_j$ is the result of the noise added by the DP mechanism. In such cases, the user may stop at Phase-2. If the user is satisfied with the confidence interval for the validity of the question, she can proceed to Phase-3.

**Influence Function (Phase-3).** When considering DP, the order of the explanation predicates is perturbed by the noise we add to the influences according to the sensitivity of the influence function (discussed in detail in Section 4.3.1). To provide useful explanations, this sensitivity needs to be low, which means the influence does not change too much by adding or removing a tuple from the database. For example, a counting query that outputs the database size $n$ has sensitivity 1, since its result can only change by 1 for any neighboring databases. Following this concept, we propose the second and a core problem for the DPXPLAIN framework, which is also critical to the subsequent problems defined below.

**Problem 2 (Influence Function with Low Sensitivity).** Find an influence function $\text{Inf} : \mathcal{P} \rightarrow \mathbb{R}$ that maps an explanation predicate to a real number and has low sensitivity.

**Private Top-k Explanations (Phase-3).** In DPXPLAIN, to satisfy DP, in Phase-3 we output the top-k explanation predicates ordered by the noisy influences, and release the influences and ranks of these predicates in the form of confidence intervals to describe the uncertainty. To achieve this goal, we tackle the following three sub-problems.

**Problem 3 (Private Top-k Explanation Predicates).** Given a set of explanation predicates $\mathcal{P}$, an integer $k$, and a privacy parameter $P_{\text{top-k}}$, find the top-k highest influencing predicates $p_1, p_2, \ldots, p_k$ from $\mathcal{P}$ while satisfying $P_{\text{top-k}}$-zCDP.
Problem 4 (Private Confidence Interval of Influence). Given a confidence level $\gamma$, $k$ explanation predicates $p_1, p_2, \ldots, p_k$, and a privacy parameter $\rho_{\text{InfOut}}$, find a confidence interval $I_{\text{infOut}} = (I_{\text{infOut}}^L, I_{\text{infOut}}^U)$ for influence $I_{\text{InfOut}}(p_u)$ at confidence level $\gamma$ for each $u \in \{1, \ldots, k\}$ satisfying $\rho_{\text{InfOut}}$-zCDP (overall privacy budget).

Problem 5 (Private Confidence Interval of Rank). Given a confidence level $\gamma$, $k$ explanation predicates $p_1, p_2, \ldots, p_k$, and a privacy parameter $\rho_{\text{Rank}}$, find a confidence interval $I_{\text{Rank}} = (I_{\text{Rank}}^L, I_{\text{Rank}}^U)$ for rank of $p_u$ at confidence level $\gamma$ for each $u \in \{1, \ldots, k\}$ satisfying $\rho_{\text{Rank}}$-zCDP (overall privacy budget).

4 COMPUTING EXPLANATIONS UNDER DP

Next we provide solutions to problems 1, 2, 3, 4, and 5 in Sections 4.1, 4.2, 4.3.1, 4.3.2, and 4.3.3 respectively, and analyze their properties. We summarize the entire DPXPLAIN framework in Section 4.4.

4.1 Confidence Interval for a User Question

For Problem 1, the goal is to find a confidence interval of $o_j - o_j$ for the user question at the confidence level $\gamma$ without extra privacy cost in Phase-2. We divide the solution into two cases. (1) When the aggregation is COUNT or SUM, the noisy difference $\hat{o}_i - \hat{o}_j$ follows Gaussian distribution, which leads to a natural confidence interval. (2) When the aggregation is AVG, the noisy difference does not follow Gaussian distribution, but we show that the confidence interval in this case can be derived through multiple partial confidence intervals. The solutions below only take the noisy query result as input, which does not incur extra privacy loss according to the post-processing property of DP (Proposition 2.9). The pseudo codes can be found in the full version [2].

Confidence interval for COUNT and SUM. For a COUNT or SUM query, recall from Section 2 that $\hat{o}_i$ and $\hat{o}_j$ are produced by adding Gaussian noises to $o_i$ and $o_j$ with some noise scale $\sigma$. Therefore, the difference between $\hat{o}_i$ and $\hat{o}_j$ also follows Gaussian distribution with mean $o_i - o_j$ and scale $\sqrt{2}\sigma$ (since the variance is $2\sigma^2$). Following the standard properties of Gaussian distribution, the interval with center $c$ as $\hat{o}_i - \hat{o}_j$ and margin $m$ as $\sqrt{2}\sigma\text{erf}^{-1}(\gamma)$, or $(c-m, c+m)$, is a $\gamma$ level confidence interval of $o_i - o_j$ [96].

Confidence interval for AVG. For an AVG query, even the single noisy answer $\hat{o}_i$ does not follow Gaussian distribution, because it is a division between two Gaussian variables as described in Section 2: $\hat{o}_i = \hat{o}_i^T / \hat{o}_i^C$. However, we can still infer a range for $\hat{o}_i$ based on the confidence intervals of $\hat{o}_i^T$ and $\hat{o}_i^C$. More specifically, we first derive partial confidence intervals for $\hat{o}_i^T$ and $\hat{o}_i^C$ as discussed above, denoted by $I_i^T$ and $I_i^C$, individually at some confidence level $\beta$. Let $I_i = I_i^T / I_i^C := \{x/y \mid x \in I_i^T, y \in I_i^C\}$ to be the set that includes all possible divisions between any numbers from $I_i^T$ and $I_i^C$. If $I_i^C$ contains zero, we return a trivial confidence interval $(\infty, \infty)$ that is always valid. Otherwise, $I_i^T$ is a $2\beta - 1$ level confidence interval for the division, as stated in the following proposition.

\[ \text{erf}^{-1} \text{ is the inverse function of the error function } \text{erf} \]  
\[ \text{erf} \int_{-x}^{x} e^{-t^2} dt. \]

*In the algorithm, we only need the maximum and the minimum of the set to construct the interval, which can be solved by a numerical optimizer.*

Lemma 4.1. Given $I_i^T$ and $I_i^C$ as two beta level confidence intervals of $\hat{o}_i^T$ and $\hat{o}_i^C$ separately, the derived interval $I_i = \{x/y | x \in I_i^T, y \in I_i^C\}$ is a $2\beta - 1$ level confidence interval of $\hat{o}_i^T / \hat{o}_i^C$.

Proof. The following holds:

\[ \text{Pr}[\hat{o}_i^T / \hat{o}_i^C \in I_i] \geq \text{Pr}[\hat{o}_i^T \in I_i^T \land \hat{o}_i^C \in I_i^C] \approx 1 - (\text{Pr}[\hat{o}_i^T \notin I_i^T] + \text{Pr}[\hat{o}_i^C \notin I_i^C]) \geq 1 - (1 - (1 - \beta) + (1 - \beta)) = 2\beta - 1. \]

The first inequality above is due to fact that the second event is sufficient for the first event: if two numbers are from $I_i^T$ and $I_i^C$, their division belongs to the set $I_i$ by definition. The next inequality holds by applying the union bound. The third inequality is by definition.

Furthermore, the difference $\hat{o}_i - \hat{o}_j$ is a subtraction between two ratios of two Gaussian variables, which can be expressed as an arithmetic combination of multiple Gaussian variables: $\hat{o}_i - \hat{o}_j = X_i / Y_i - X_j / Y_j$, where $X_i = N(\hat{o}_i^T, \sigma_i^2)$ and $Y_i = N(\hat{o}_i^C, \sigma_i^2)$ for $t \in \{i,j\}$. Similar to Lemma 4.1, we can derive the confidence interval for $\hat{o}_i - \hat{o}_j$ based on 4 partial confidence intervals of $\hat{o}_i^T$, $\hat{o}_i^C$, $\hat{o}_j^T$, and $\hat{o}_j^C$ instead of 2. The confidence level we set for each partial confidence interval is $\beta = 1 - (1 - \gamma)/4$ by applying union bound on the failure probability $1 - \gamma$ that one of the four variables is outside its interval. After we have 4 partial confidence intervals $I_i^T$, $I_i^C$, $I_j^T$, and $I_j^C$ for $\hat{o}_i^T$, $\hat{o}_i^C$, $\hat{o}_j^T$, and $\hat{o}_j^C$ separately, similar to Lemma 4.1, we combine them together as $I_i = I_i^T / I_i^C - I_j^T / I_j^C$ and derive the confidence interval for $o_i - o_j$ as $(\inf I_i, \sup I_i)$, which is guaranteed to be at confidence level $\gamma$. If $j$ is included in either $I_i^T$ or $I_i^C$, we set the confidence interval to be $(\infty, \infty)$ instead. Although there is no theoretical guarantee of the interval width, from two case studies in Section 5.2, we demonstrate narrow confidence intervals of AVG queries in practice, and observe no extreme case $(\infty, \infty)$ in the experiments.

4.2 Influence Function with Low Sensitivity

For Problem 2, the goal is to design an influence function that has low sensitivity. Inspired by PrivBayes [100], we start by adapting a known influence function to our framework.

Our influence function of an explanation predicate with respect to a comparison user question is inspired by the Scorpion framework [98], where the user questions seek explanations for outliers in the results of a group-by aggregate query. Scorpion identifies predicators on the input that cause the outliers to disappear from the output. Given the group-by aggregation query shown in Figure 3 and a group $a_t \in \text{dom}(A_{gh})$, recall from Section 2 that the true aggregate value for $a_t$ is $a_t = \text{agg}(A_{agg}, g_t(D))$, where $g_t = \phi(A_{gh} = a_t)$, i.e., $g_t(D)$ denotes the set of tuples that contribute to the group $a_t$. Scorpion measures the influence of an explanation predicate $p$ to some group $a_t$ as the ratio between the change of output aggregate value and the change of group size:

\[ \frac{\text{agg}(g_t(D)) - \text{agg}(g_t(-p(D)))}{|g_t(p(D))|} \]

Here $-p(D)$ denotes $D - p(D)$, i.e., the set of tuples in $D$ that do not satisfy the predicate $p$. To adapt this influence function to DPXPLAIN, we make the following two changes.

- First, it should measure the influence w.r.t. the comparison from the user question $(a_t, a_{t'})$ instead of a single group. A natural extension is to change the target aggregate on $g_t$ in the numerator

\[ \text{in} \text{to the difference between the aggregate values of two groups} \]
Before and after applying the explanation predicate \( p \), and change the denominator as the maximum change in \( g_i \) or \( g_j \) when \( p \) is applied, which gives the following influence function:

\[
\frac{\text{agg}(g_i(D)) - \text{agg}(g_j(D)) - (\text{agg}(g_i(\neg p(D))) - \text{agg}(g_j(\neg p(D))))}{\max(|g_i(p(D))|, |g_j(p(D))|)}
\]

(2)

- Second and more importantly, in DPXPlain, we need to preserve DP when we use influence function to sort and rank multiple explanation predicates, or to release the influence and rank of an explanation predicate. Therefore, we need to account for the sensitivity of the influence function, which is determined by the worst-case change of influence when a tuple is added or removed from the database. If the predicate only selects a small number of tuples, the denominator in (2) is small and thus changing the denominator in (2) by one (when a tuple is added or removed) can result in a big change in the influence as illustrated in the following example, making (2) unsuitable for DPXPlain.

**Example 4.2 (The Issue of the Influence Sensitivity).** Suppose there are two groups \( a_i \) and \( a_j \) in \( D \) with 1000 tuples in each, and the aggregate function \( \text{agg} = \text{SUM} \) on attribute \( A_{\text{agg}} \) with domain \([0, 100] \), and the explanation predicate \( p \) matches only 1 tuple from the group \( a_i \) with \( A_{\text{agg}} = 100 \) and no tuple from \( a_j \). Suppose \( \text{agg}(g_i(D)) = 20,000 \), \( \text{agg}(g_j(D)) = 10,000 \), then \( \text{agg}(g_i(\neg p(D))) = 19,900 \) and \( \text{agg}(g_j(\neg p(D))) = 10,000 \). Therefore, from Equation (2), the influence of \( p \) is \(((20,000 - 10,000) - (19,900 - 10,000))/\max(1, 0) = 100 \) on the original database \( D \). However, suppose a new tuple that satisfies \( p \) and belongs to group \( a_i \) is added with \( A_{\text{agg}} = 2 \). Now the influence in Equation (2) becomes \(((20,002 - 10,000) - (19,900 - 10,000))/\max(2, 0) = 102/2 = 51 \). Note that while we added a tuple contributing only 2 to the sum, it led to a change of 100-51 = 49 to the influence function because of the small denominator.

Therefore, we propose a new influence function that is inspired by Equation (2) but has lower sensitivity. Note that the denominator in Scorpion’s influence function in Equation (2) acts as a normalizing factor, whose purpose is to penalize the explanation predicate that selects too many tuples, e.g., to prohibit the removal of the entire database by a dummy predicate. To have a similar normalizing factor with low sensitivity, we multiply the numerator in Equation (2) by \( \min(|g_i(\neg p(D))|, |g_j(\neg p(D))|)/\max(|g_i(D)|, |g_j(D)|+1) \). From this new normalizing factor, the numerator captures the minimum of the number of tuples that are not removed from each group, and the denominator keeps the normalizing factor in the interval \([0, 1] \) and does not change for different explanation predicates. Similar to Scorpion, if \( p(D) \) constitutes a large fraction of \( D \) (e.g., if \( p(D) = D \)), then the normalizing factor is small, reducing the value of the influence. Also note that, unlike standard SQL query answering where only non-empty groups are shown in the results, in DP, all groups from the actual domain have to be considered, hence unlike Equation (1), \( g_i(D), g_j(D) \) could be zero, hence 1 is added in the denominator to avoid division by zero. When \( \text{agg} = \text{AVG} \), we remove the constant denominator to boost the signal of the influence and keep the sensitivity low, which will be discussed in the sensitivity analysis after Proposition 4.4 and in Example 4.5.

**Definition 4.3 (Influence of Explanation Predicates).** Given a database \( D \), a query \( q \) as shown in Figure 3, and a user question \((a_i, a_j, >)\), the influence of an explanation predicate \( p \) is defined as

\[
\text{Inf}(p; (a_i, a_j, >), D), \text{ or simply } \text{Inf}(p) \text{ when clear from context:}
\]

\[
\text{Inf}(p) = \left(\text{agg}(g_i(D)) - \text{agg}(g_j(D)) - (\text{agg}(g_i(\neg p(D))) - \text{agg}(g_j(\neg p(D))))\right) \times \frac{\min(|g_i(\neg p(D))|, |g_j(\neg p(D))|)}{\max(|g_i(D)|, |g_j(D)| + 1)}
\]

for \( \text{agg} \in \{\text{COUNT, SUM}\} \)

(3)

The next proposition summarizes the sensitivity of eq. (3).

**Proposition 4.4. [Influence Function Sensitivity]** Given an explanation predicate \( p \) and a user question with respect to a group-by query with aggregation \( \text{agg} \), the following holds:

1. If \( \text{agg} = \text{COUNT} \), the sensitivity of \( \text{Inf}(p) \) is 4.
2. If \( \text{agg} = \text{SUM} \), the sensitivity of \( \text{Inf}(p) \) is 4 \( A_{\text{agg}}^{\text{max}} \).
3. If \( \text{agg} = \text{AVG} \), the sensitivity of \( \text{Inf}(p) \) is 16 \( A_{\text{agg}}^{\text{max}} \).

We give an intuitive proof as follows, where the formal proofs are deferred to the full version [2] due to space restrictions. When \( \text{agg} = \text{COUNT} \), we combine two group differences \( \text{agg}(g_i(D)) - \text{agg}(g_j(D)) - (\text{agg}(g_i(\neg p(D))) - \text{agg}(g_j(\neg p(D)))) \) into a single group difference as \( \text{agg}(g_i(p(D)) - \text{agg}(g_j(p(D))) \), which is considered as a subtraction between two counting queries. We prove that the sensitivity of a counting query after a multiplication with the normalizing factor will multiply its original sensitivity by 2. Since we have two counting queries, the final sensitivity is 4. When \( \text{agg} = \text{SUM} \), the proof is similar except we need to multiply the final sensitivity by \( A_{\text{agg}}^{\text{max}} \), the maximum absolute domain value of \( A_{\text{agg}} \). For \( \text{AVG} \), we view it as a summation of 4 AVG queries that times with \( \min(|g_i(\neg p(D))|, |g_j(\neg p(D))|) \). Intuitively, we change \( \text{AVG} \) to \( \text{SUM} \) and bound the sensitivity. This sensitivity now becomes relatively small since we have amplified the influence.

Intuitively, the sensitivity of \( \text{Inf}(p) \) is low compared to its value. When \( \text{agg} = \text{COUNT} \), \( \text{Inf}(p) \) is \( O(n) \) and \( \Delta_{\text{inf}} \) is \( O(1) \), where \( n \) is the size of database. When \( \text{agg} \in \{\text{SUM, AVG}\} \), \( \text{Inf}(p) \) is \( O(n A_{\text{agg}}^{\text{max}}) \) and \( \Delta_{\text{inf}} \) is \( O(A_{\text{agg}}^{\text{max}}) \). Therefore, the sensitivity of influence \( \Delta_{\text{inf}} \) is low compared to the influence itself. However, as the example below shows, if we define the influence function for \( \text{AVG} \) the same way as \( \text{COUNT} \) or \( \text{SUM} \), both \( \text{Inf}(p) \) and \( \Delta_{\text{inf}} \) will become \( O(A_{\text{agg}}^{\text{max}}) \), which makes the sensitivity (relatively) large.

**Example 4.5 (The Issue with AVG Influence).** Consider an AVG group-by query where the domain of the aggregate attribute is \([0, 100] \), and an explanation predicate \( p \) such that for group \( a_i \) we have 2 tuples with \( \text{AVG}(g_i(D)) = 100/2 = 50 \), \( \text{AVG}(g_i(\neg p(D))) = 0/1 = 0 \), and for group \( a_j \) we have two tuples with \( \text{AVG}(g_j(D)) = 100/2 = 50 \) and \( \text{AVG}(g_j(\neg p(D))) = 100/2 = 50 \). Suppose we define the influence function for \( \text{AVG} \) the same way as \( \text{COUNT} \) or \( \text{SUM} \), both \( \text{Inf}(p) \) and \( \Delta_{\text{inf}} \) will become \( O(A_{\text{agg}}^{\text{max}}) \). Note that a single removal of a tuple completely changes the influence to 0, and this change is equal to the influence itself, which is relatively large and therefore not a good choice for \( \text{AVG} \).

Note that the user question “why \((a_i, a_j, >)\)” is asked based on the noisy results \( \hat{a}_i > \hat{a}_j \), while the influence function uses the true results, i.e., even if \( \hat{a}_i \leq \hat{a}_j \), we still consider \( \text{agg}(g_i(D)) - \text{agg}(g_j(D)) \) in \( \text{Inf}(p) \). Hence \( \text{Inf}(p) \) can be positive or negative and
removing tuples satisfying $p$ can make the gap smaller or larger. In the full version [2], we show that $\text{Inf}(p)$ is not monotone with $p$-s.

4.3 Private Top-$k$ Explanations

In this section, we discuss the computation of the top-$k$ explanation predicates and the confidence intervals of influences and ranks.

4.3.1 Problem 3: Private Top-$k$ Explanation Predicates. The goal is to find with DP the top-$k$ explanation predicates from a set of explanation predicates $P$ in terms of their (true) influences $\text{Inf}(p)$, which is the first step in Phase-3 of DPxPlain (Figure 1). Note that simply choosing the true top-$k$ explanation predicates in terms of their $\text{Inf}(p)$ is not differentially private.

In DPxPlain, we adopt the One-shot Top-$k$ mechanism [36, 37] to privately select the top-$k$. It works as follows. For each explanation predicate $p \in P$, it adds a Gumbel noise $^{10}$ to its influence with scale $\sigma = 2\Delta_{\text{Inf}} \sqrt{k/(8\text{TOP}_k)}$, where $\Delta_{\text{Inf}}$ is the sensitivity of the influence function (discussed in Proposition 4.4), reorders all the explanation predicates in descending order by their noisy influences, and outputs the first $k$ explanation predicates. It satisfies $\text{TOP}_k$-zCDP [16, 31, 36, 79], since it is equivalent to iteratively applying $k$ exponential mechanisms [41], where each satisfies $e^{2}\text{TOP}_k$-zCDP [16, 31, 36, 79] and $c = \sqrt{8\text{TOP}_k/k}$ [36, 37]. Therefore, in total it satisfies $(ke^{2}/8)$-zCDP by the sequential composition property (Proposition 2.9) which is also $\text{TOP}_k$-zCDP. The returned list of top-$k$ predicates is close to that of the true top-$k$ in terms of their influences; the proof is based on the utility proposition of the exponential mechanism in Theorem 3.11 of [41]. Since this algorithm iterates over each explanation predicate, the time complexity is proportional to the size of the explanation predicate set $P$. By Definition 3.3, this number is $O((m/2)^{N^{l}})$, where $N$ is the maximum domain size of an attribute, $l$ is the number of conjuncts in the explanation predicate, and $m$ is the number of attributes. In our experiments (Section 5), we fix $l = 1$ and use all the singleton predicates as the set $P$, so its size is linear in the number of attributes. We summarize the properties of this approach in the following proposition and defer the pseudo codes and proofs to [2].

Proposition 4.6. Given an influence function $\text{Inf}$ with sensitivity $\Delta_{\text{Inf}}$, a set of explanation predicates $P$, a privacy parameter $\text{TOP}_k$ and a size parameter $k$, the following holds:

1. One-shot Top-$k$ mechanism finds $k$ explanation predicates while satisfying $\text{TOP}_k$-zCDP.
2. Denote by OPT$^{(i)}$ the $i$-th highest (true) influence, and by $M^{(i)}$ the $i$-th explanation predicate selected by the One-shot Top-$k$ mechanism. For $\forall i$ and $\forall i \in \{1, 2, \ldots, k\}$, we have

$$
\Pr[\text{Inf}(M^{(i)}) \leq \text{OPT}^{(i)} - \frac{2\Delta_{\text{Inf}}}{\sqrt{8\text{TOP}_k/k}} (\ln(|P|) + 1)] \leq e^{-t} \quad (4)
$$

Example 4.7. Reconsider the user question in Figure 1c. For this question, we have in total 103 explanation predicates as the set of explanation predicates. The privacy budget $\text{TOP}_k = 0.05$, the size parameter $k = 5$, and the sensitivity $\Delta_{\text{Inf}} = 16$. For each of the explanation predicate, we add a Gumbel noise with scale $\sigma = 113$ to their influences. For example, for the predicates shown in Figure 1d, their noisy influences are 990, 670, 645, 475, 440, which are the highest 5 among all the noisy influences. The true influences for these five ones are 547, 501, 555, 434, 113. To see how close it is to the true top-5, we compare their true influences with the true highest five influences: 555, 547, 501, 434, 252, which shows the corresponding differences in terms of influence are 8, 46, 54, 0, 134. By Equation (4), the probability that such difference is beyond 864 is at most 5% for each explanation predicate. Finally, we sort explanation predicates by their noisy influences and report the top-$k$. These $k$ predicates will be reordered as discussed in Section 4.4.

4.3.2 Problem 4: Private Confidence Interval of Influence. The goal is to generate a confidence interval of influence $\text{Inf}(p)$ (Definition 4.3) of each explanation predicate $\text{Inf}(p_{1}), \text{Inf}(p_{2}), \ldots, \text{Inf}(p_{k})$ from the selected top-$k$ (Section 4.3.1). For each $\text{Inf}(p_{i})$, we apply the Gaussian mechanism (Theorem 2.8) with privacy budget $\text{Inf}_{li}/k$ to release a noisy influence $\widehat{\text{Inf}}_i$ with noise scale $\sigma = \Delta_{\text{Inf}}/\sqrt{2\text{Inf}_{li}/k}$. The sensitivity term $\Delta_{\text{Inf}}$ is determined by Proposition 4.4. Following the standard properties of Gaussian distribution, for each $\text{Inf}(p_{i})$, we set the confidence interval by a center $c$ as $\widehat{\text{Inf}}_i$ and a margin $m$ as $\sqrt{2\gamma \sigma \text{erf}^{-1}(\gamma)}$, or $(c-m, c+m)$, as a $\gamma$ level confidence interval of $\text{Inf}(p_{i})$ [96]. Together, it satisfies $\text{Inf}_{li}$-zCDP according to the composition property by Proposition 2.9. Pseudo codes can be found in the full version [2].

4.3.3 Problem 5: Private Confidence Interval of Rank. The goal is to find the confidence interval of the rank of each explanation predicate from the selected top-$k$ (Section 4.3.1). We denote rank($p$) as the rank of $p \in P$ by the natural ordering of the predicates imposed by their (true) influences according to the influence function $\text{Inf}$, and denote rank$^{-1}(t)$ (for an integer $1 \leq t \leq |P|$) as the predicate ranked in the $t$-th place according to $\text{Inf}$. One trivial example of a confidence interval of rank is $[1, |P|]$, which has no privacy loss and always includes the true rank.

Unlike the sensitivity of the influence function, the sensitivity of rank($p$) is high, since adding one tuple could possibly change the highest influence to be the lowest and vice versa. Fortunately, we can employ a critical observation about rank and influence.

Proposition 4.8. Given a set of explanation predicates $P$, an influence function $\text{Inf}$ with global sensitivity $\Delta_{\text{Inf}}$, and an integer $1 \leq t \leq |P|$, $\text{Inf}(\text{rank}^{-1}(t))$ has sensitivity $\Delta_{\text{Inf}}$.

The intuition behind this proof (details in [2]) is that, fixing an explanation predicate $p = \text{rank}^{-1}(t)$, for a neighboring database, if its influence is increased, its rank will be moved to the top which pushes down other explanation predicates with lower influences, so the influence at the rank $t$ in the neighboring database is still low. For a target explanation predicate $p$, since both $\text{Inf}(p)$ and $\text{Inf}(\text{rank}^{-1}(t))$ have low sensitivity as $\Delta_{\text{Inf}}$, intuitively we can check whether $t$ is close to the rank of $p$ by checking whether their influences $\text{Inf}(p)$ and $\text{Inf}(\text{rank}^{-1}(t))$ are close by adding a little noise to satisfy DP. Given this observation, we devise a binary-search-based strategy to find the confidence interval of rank.

Noisy binary search mechanism. We decompose the problem into finding two bounds of the confidence interval separately by a subroutine $\text{RANKBOUND}(p, p_{\beta}, p_{\beta}, \text{dir})$ that guarantees that it will find a lower ($\text{dir} = -1$) or upper ($\text{dir} = +1$) bound of rank with

---

10For a Gumbel noise $Z \sim \text{Gumbel} (\sigma)$, its CDF is $\Pr[Z \leq z] = \exp(-\exp(-z/\sigma))$. 
probability β for the explanation predicate p using privacy budget ρ. We divide the privacy budget ρ into two parts by a parameter η ∈ (0, 1) and return (RankBound(ρu, ηp, β, 1), RankBound(ρu, (1− η)p, β, +1)) as the confidence interval of rank for each predicate ρu for u ∈ {1, . . . , k}, where ρ = ρRank/k to divide the total privacy budget equally, and β = (γ + 1)/2 to ensure a confidence of γ.

The subroutine RankBound(p, ρ, β, dir) works as follows. It is a noisy binary search with at most N = ⌊log2(|P|)⌋ loops. We initialize the search pointers tlow = 1 and thigh = |P| as the two ends of possible ranks. Within each loop, we check the difference of influences at t = [(thigh + tlow)/2] by adding a Gaussian noise:

\[ t = \text{Inf}(p) - \text{Inf}(\text{rank}^{-1}(t)) + N(0, \sigma^2) \]  

(5)

The noise scale is set as \( \sigma = (2\Delta_{\text{Inf}})/\sqrt{2(p/N)} \) to satisfy \( ρ/N \)-zCDP. Instead of comparing the noisy difference t with 0 to check whether t is a close bound of rank(p), we compare it with the following slack constant ξ so that w.h.p. t is a true bound of rank(p).

\[ \xi = \sigma \sqrt{2 \ln(N/(1 - β))} \times \text{dir} \]  

(6)

We update the binary search pointers by the comparison as follows: if \( t ≥ \xi \), we set \( t_{\text{high}} = \text{max}(t - 1, 1) \), otherwise \( t_{\text{low}} = \text{min}(t + 1, |P|) \). The binary search stops when \( t_{\text{high}} ≤ t_{\text{low}} \) and returns \( t_{\text{high}} \) as the rank bound. We defer the pseudo codes to [2].

Example 4.9. Figure 4 shows an example of RankBound for finding the upper bound of the confidence interval for rank(p) for some explanation predicate p (with true rank 3 shown in red). The upper part of the figure shows the influences of all the explanation predicates in descending order, and the lower part shows the status of the binary search pointers in each loop. The search contains three loops starting from \( t_{low} = 1 \) and \( t_{high} = 15 \). Within each loop, to illustrate the idea, it is equivalent to adding a Gaussian noise to \( \text{Inf}(\text{rank}^{-1}(t)) \), which is shown as a blue circle, compare it with \( \text{Inf}(p) - \xi \), which is shown as a dashed line, and update the pointers accordingly. For example, in loop 1, the blue circle 1 is in the green region, so the pointer \( t_{high} \) is moved from 15 to 7 (shown in the lower part). Finally, it breaks at \( t_{low} = t_{high} = 5 \).

We now show that noisy binary search mechanism satisfies the privacy requirement, and outputs valid confidence intervals. In Section 5, we show that the interval width is empirically small.

Theorem 4.10. Given a database D, a predicate space P, an influence function \( \text{Inf} \) with sensitivity \( \Delta_{\text{Inf}} \), explanation predicates \( \rho_1, \rho_2, \ldots, \rho_k \), a confidence level γ, and a privacy parameter \( \rho_{\text{Rank}} \), noisy binary search mechanism returns confidence intervals \( I_1, I_2, \ldots, I_k \) such that

1. Noisy binary search mechanism satisfies \( \rho_{\text{Rank}} \)-zCDP.
2. For \( u ∈ [1, k] \), \( I_u \) is a γ level confidence interval of \( \text{rank}(p_u) \).

The proof of item 1 follows from the composition theorem and the property of Gaussian mechanism [14]. The proof of item 2 is based on the property of the random binary search. We defer the formal proofs and a weak utility bound to the full version [2].

4.4 Putting it All Together

We now show how all the steps fit together into DPXPLAIN.
5 EXPERIMENTS

In this section, we evaluate the quality and efficiency of the explanations generated by DPXPlain. To our knowledge, there are no existing benchmarks for explanations for query answers (even without privacy consideration) in the database research literature. We have implemented DPXPlain [1] in Python 3.7.4 using the Pandas [92], NumPy [51], and SciPy [95] libraries. All experiments were run on Intel i7-7700 CPU @ 3.60GHz with 32 GB of RAM.

5.1 Experiment Setup

We first detail the data, queries, questions, and parameters.

Datasets. We consider two datasets in our experiments.

- **IPUMS-CPS (real data):** A dataset of Current Population Survey from the U.S. Census Bureau [47] with 1,146,552 tuples from the year 2011 to 2019. The dataset contains 8 categorical attributes where domain sizes vary from 3 to 36 and one numerical attribute. The attribute `AGE` is discretized as 10 years per range, e.g., `[0, 10)` is considered a single value. To set the domain of numerical attributes, we only include tuples with attribute `INCTOT` (the total income) smaller than 200k as a domain bound.

- **German-Credit (synthetic data):** A corrected collection of credit data [50]. It includes 20 attributes where the domain sizes vary from 2 to 11 and a numerical attribute. Attributes `duration`, `credit_amount`, and `age` are discretized. The domain of attribute `good-credit` is yes or one. We synthesize the dataset to 1 million rows by combining a Bayesian network learner [7] and XGBoost [13] following the strategy of QUAIL [80].

Queries and Questions. The queries and questions used on the experiments are shown in Table 1.

<table>
<thead>
<tr>
<th>Explanation Predicate</th>
<th>Rel Ind L 95%-CI Rank L 95%-CI</th>
<th>Rel Ind U 95%-CI Rank U 95%-CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATE = “Single”</td>
<td>12.18% 12.32% 1 1 12.41% 1 1</td>
<td>6.54% 6.67% 2 2 6.54% 6.67% 2 2</td>
</tr>
<tr>
<td>EDUC = “Master’s degree”</td>
<td>7.16% 7.42% 2 3 7.32% 2 2</td>
<td>6.54% 6.75% 2 2 6.54% 6.75% 2 2</td>
</tr>
<tr>
<td>RACE = “White”</td>
<td>6.11% 6.75% 2 2 6.11% 6.75% 2 2</td>
<td>6.01% 6.41% 2 2 6.01% 6.41% 2 2</td>
</tr>
<tr>
<td>RELATE = “Spouse”</td>
<td>5.70% 6.04% 2 2 5.70% 6.04% 2 2</td>
<td>6.01% 6.41% 2 2 6.01% 6.41% 2 2</td>
</tr>
<tr>
<td>CLASSNBR = “NUI”</td>
<td>3.83% 4.18% 2 2 3.83% 4.18% 2 2</td>
<td>6.42% 6.83% 2 2 6.42% 6.83% 2 2</td>
</tr>
</tbody>
</table>

Figure 5: Phase-3 of DPXPlain for the case IPUMS-CPS.
In Phase-3 of the German-Credit dataset, we find that 8 out of 10 questions (plotted together for clarity) from Table 1 are classified correctly with an accuracy of 100%.

5.3 Accuracy and Performance Analysis

We detail our experimental analysis for the different questions and configurations of DPXPlain. All results are averaged over 10 runs.

Correctness of noise interval. In Phase-2 of DPXPlain, the validity of the question is suggested as follows: if the confidence interval contains non-positive numbers, the question is invalid, otherwise valid. From Figure 7, we find that 8 out of 10 questions (plotted together for clarity) from Table 1 are classified correctly with an accuracy of 100% given a wide range of privacy budget of query $\rho_q$.

Accuracy of top-k explanation predicates. In Phase-3 of DPXPlain, we first select top-k explanation predicates. We measure the accuracy of the selection by Precision@k \[52\], the fraction of the selected top-k explanation predicates that are actually ranked within top-k. Another experiment on the full ranking is included in the full version \[2\]. From Figure 8a, we find that the privacy budget of top-k selection $\rho_{Topk}$ has a positive effect to Precision@k at $k = 5$ for various questions. When $\rho_{Topk} = 1.0$, all the questions except I2 and I5 have Precision@k $\geq 0.8$. The selection accuracy of question I2 and I5 are generally lower because of small group sizes, and, therefore, the influences of explanation predicates are small and the rankings are perturbed by the noise more significantly.

From Figure 8b, we find that the trend of Precision@k by k is different across questions and there is no clear trend that Precision@k increases as k increases. For example, for G3, it first decreases from k=3 to k=5, but increases from k=5 to k=6. When k = 3, most questions have high Precision@k; this is because the highest three influences are much higher than the others, which makes the probability high to include the true top three. With larger k, explanation predicates that have similar scores have an equal probability to be included in top-k and therefore the top-k selected by the algorithm are different from the true top-k selections. The relationship between Precision@k and k depends on the distribution of all the explanation predicate influences.

Figure 6: Phase-3 of DPXPlain for the case German-Credit.

Figure 7: The probability of correctly validating user questions. All questions except I2 and I5 (Figure 7) are at 100%.

Figure 8: Precision@k of top-k selection by DPXPlain.
needs to evaluate each sub-query. For the step noise quantification and confidence interval of rank, the time usage is not significant since the first only needs to find the image of two intervals and the second is a binary search. Figure 10b shows that the runtime is linearly proportional to the size of explanations $k$, and the difference between questions is due to the difference of group sizes. We also find the runtime grows exponentially with the number of conjuncts $l$ as the number of explanation predicates grows exponentially: for $l = 1, 2, 3$, the runtime about question I1 is $67, 3078$ and $79634$, and for question G1 it is $40, 1587$ and $39922$ seconds.

Figure 10: Runtime analysis of DPXPlain.

6 RELATED WORK

We next survey related work in the fields of DP and explanations for query results. To the best of our knowledge, DPXPlain is the first work that explains aggregate query results while satisfying DP.

Explanations for query results. The database community has proposed several approaches to explaining aggregate and non-aggregate queries in multiple previous works. Proposed approaches include provenance [17, 26, 27, 53, 54, 62, 63, 93], intervention [81, 82, 98], entropy [43], responsibility [73, 74], Shapley values [67, 78], counterbalance [75] and augmented provenance [65], and several of these approaches have used predicates on tuple values as explanations like DPXPlain, e.g., [43, 65, 82, 98]. We note that any approaches that consider individual tuples or explicit tuple sets in any form as explanations (e.g., [26, 63, 67, 73]) cannot be applied in the DP setting since they would violate privacy. Among the other summarization or predicate-based approaches, Scorpion [98] explains outliers in query results with the intervention of most influential predicates. Our influence function (Section 4.2) is inspired by the influence function of Scorpion, but has been modified to deliver accurate results while satisfying DP. Another intervention-based work [82] that also uses explanation predicates, models inter-dependence among tuples from multiple relations with causal paths. DPXPlain does not support joins in the queries, which is a challenging future work (see Section 7).

Differential privacy. Private SQL query answering systems [32–34, 36, 59, 60, 69, 90, 97] consider a workload of aggregation queries with or without joins on a single or multi-relational database, but none supports explanation under differential privacy. The selection of private top-$k$ candidates is well-studied by the community [8, 10, 11, 15, 18, 30, 37, 61, 66, 70, 71, 77, 91]. We adopt One-shot Top-$k$ mechanism [77] since it is easy to understand. Private confidence interval is a new trend of estimating the uncertainty under differential privacy [12, 21, 45], however, the current bootstrap based methods measure the uncertainty from both the sampling process and the noise injection, of which the first part is unnecessary, and we only focus on the second part. The most relevant work to the private rank estimation is private quantile [4, 20, 39, 48, 57, 64, 86], which is to find the value given a position such as median, but the problem of rank estimation in our setting is reversed.

Privacy and provenance. As mentioned earlier, data provenance is often used for explaining query results, mainly for non-aggregate queries. Within the context of provenance privacy [6, 9, 19, 22, 23, 83, 85, 88], one line of work [22–24] studied the preservation of workflow privacy (privacy of data transferred in a workflow with multiple modules or functions), with a privacy criterion inspired by $l$-diversity [68]. A recent work [28] explored what can be inferred about the query from provenance-based explanations and found that the query can be reversed-engineered from the provenance in various semirings [49]. To account for this, a follow-up paper [25] proposed an approach for provenance obfuscation that is based on abstraction. This work uses $k$-anonymity [87] to measure how many ‘good’ queries can generate concrete provenance that can be mapped to the abstracted provenance, thus quantifying the privacy of the underlying query. Devising techniques for releasing provenance of non-aggregate and aggregate queries while satisfying DP is an interesting research direction.

7 FUTURE WORK

There are several interesting future directions. Extending DPXPlain to more general queries (like joins) and questions is an important future work. Unlike standard explanation frameworks like [98] where the join results can be materialized before running the explanation mechanism, a careful sensitivity analysis of adding/removing tuples from multiple tables is needed in the DP settings [90].

Second, the complexity of the top-$k$ selection algorithm links to the number of explanation predicates that could be exponentially large, leaving room for future improvements. Additionally, other interesting notions of explanations for query answers (e.g., [65, 67, 75]) can be explored in the DP setting. Finally, evaluating our approach with a comprehensive user study and examining different metrics of understandability of the explanations generated by DPXPlain is also an important direction for future investigation.

ACKNOWLEDGMENTS

This work was supported by the NSF awards IIS-2147061, IIS-2016393, IIS-2008107, IIS-1703431, and IIS-1552538.