# Towards Visualizing Electronic Medical Records via Natural Language Queries

**Anonymous ACL submission**


## Abstract

Electronic medical records (EMRs) contain critical data for patient care and clinical research, but their complexity—blending structured data and unstructured clinical notes—hinders intuitive analysis. Data visualization could address this, yet the lack of large-scale, domain-specific text-to-visualization datasets (compounded by high manual annotation costs) has stalled progress. To bridge the gap, we propose an innovative approach using large language models (LLMs) for generating visualization data without labor-intensive manual annotation. We introduce a new pipeline for building text-to-visualization benchmarks suitable for EMRs, enabling users to visualize EMR statistics through natural language queries (NLQs). We release MedicalVis, the first large-scale EMR text-to-visualization benchmark (35,374 examples) comprising paired NLQs, EMR data, and diverse visualizations. Additionally, we introduce MedCodeT5, a multi-task model fine-tuned on MedicalVis, which outperforms various strong baselines. Our work establishes a critical benchmark for EMR visualization, enabling standardized evaluation and accelerating progress in clinical decision support and medical research via intuitive NLQ-driven EMR exploration.


## 1 Introduction

As a critical component of the modern healthcare system, electronic medical records (EMRs)(Williams and Boren, 2008; Fleming et al., 2024) documenting patients' personal information, medical histories, diagnoses, treatments, and medication use in detail. This structured and unstructured data offers invaluable insights for improving clinical decisions and medical research. However, efficiently analyzing EMR data remains an open challenge. The free-form nature of EMR entries such as clinical notes, and the diversity of data types make it challenging to extract key information and trends.

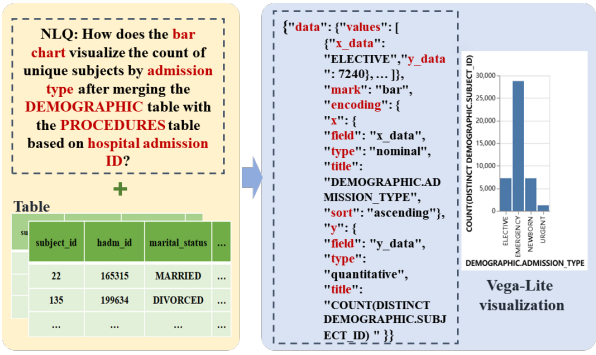



Figure 1: A text-to-visualization example from our proposed MedicalVis dataset, demonstrates how users can visually explore electronic medical record (EMR) data through natural language queries (NLQs).

Although text-to-SQL methods in the medical field have made some progress, such as the development of MIMICSQL (Wang et al., 2020) and EHRSQL (Lee et al., 2024), the data retrieved by structured query language (SQL) statements still lack intuitive clarity, necessitating further analysis and exploration. One promising solution is applying data visualization techniques that transform EMR data into intuitive graphical summaries. A crucial step in obtaining data visualization on a database involves writing visualization specifications in a declarative visualization language (DVL, e.g., Vega-Lite (Satyanarayan et al., 2017), ECharts (Li et al., 2018)), outlining the necessary data and its visual representation. However, proficiency in DVLs requires in-depth knowledge of domain-specific data and expertise in these languages, which is a significant barrier for beginners and non-technical users, and existing medical visualization tools are limited to focusing on a single data type such as genomic sequences (Pandey et al., 2023) and the domain of epidemiology (Crisan et al., 2022). Moreover, current text-to-visualization benchmarks focus on cross-domain applications, but there is a lack of research on unified EMR visualization that can handle the diverse

content of medical records. While methods like WSRP (Cai et al., 2023) offer solutions in the absence of large-scale labeled datasets, a significant gap remains in effective visualization tools for exploring and analyzing EMR databases. There is an urgent need to establish a benchmark for systematically comparing EMR visualization techniques. Overall, directly applying existing data visualization methods on EMRs suffers two main challenges: (1) Medical terminology abbreviations. (2) Lack of large-scale healthcare text-to-visualization dataset.

To advance automatic data visualization technology in the medical field, we propose a novel data creation pipeline method that leverages LLMs (e.g. GPT-4 (OpenAI et al., 2024)) to develop a text-to-visualization benchmark tailored for EMRs. Fig. 1 illustrates an example of text-to-visualization functionality provided within our MedicalVis dataset. In this instance, an NLQ and a database schema serve as inputs, requiring the text-to-visualization models to effectively comprehend the text, extract pertinent information for visualization, and ensure coherence and consistency among the resultant infographic elements. Specifically, the dataset construction pipeline includes three critical steps: (i) *DVQs generation*[1]. By crafting appropriate SQL-to-DVQ prompts, we leverage LLMs to transform SQL statements from the EMR domain into potential DVQs. (ii) *DVQs filter*. The data visualizations generated from DVQs may contain inadequate or unsuitable components, To address these potential issues, we have implemented a comprehensive set of criteria to filter and refine the candidate DVQs based on their data visualizations. (iii) *NLQs generation*. In the final step, through the design of a suitable DVQ-to-NLQ prompt, we employ LLMs as annotators to generate imperative and interrogative NLQs corresponding to each DVQ.

After the above-mentioned pipeline, we obtain the first large-scale text-to-visualization dataset for EMRs, entitled **MedicalVis**, comprising 35,374 examples. Following a thorough analysis and evaluation, we scrutinize the performance of various text to visualization models, namely Seq2Vis (Luo et al., 2021), ncNet (Luo et al., 2022), and CodeT5 (Wang et al., 2021), utilizing the MedicalVis dataset. The experimental findings indicate the efficacy of the text-to-visualization approach in facilitating the visualization analysis of the EMR database. This not only unveils the information encapsulated within EMRs but also validates the dataset's high quality.

The contributions of this paper are summarized as follows:

- We release MedicalVis, the first large-scale benchmark for text-to-visualization in the EMR domain. This dataset will facilitate further research in automated data visualization and analysis of EMRs.
- We introduce an innovative data creation pipeline that leverages LLMs to generate a large-scale, high-quality text-to-visualization dataset, thereby mitigating the need for labor-intensive manual annotation efforts.
- We present MedCodeT5, a multi-task pre-trained model based on CodeT5, and extensively evaluate its performance against various text-to-visualization baselines. The experiment validates the feasibility of analyzing and visualizing EMR data through natural language interaction.

The rest of the paper is organized as follows. Section 2 reviews the related work. Moving on to Section 3, we delve into the intricacies of the fundamental concept of the text-to-visualization task within the EMR domain. In Section 4, we introduce our newly proposed dataset, MedicalVis. Section 5 presents the details of the designed methods. Section 6 shows experimental results, followed by a conclusion in Section 7.

## 2 Related Work

### 2.1 EMR Data Analysis

EMRs and medical images are key sources of medical big data. Eladio et al. (Eladio and Wu, 2024) built a dataset with the SQuAD V2.0 framework to support EMR question answering. Deep learning has improved EMR analysis: Liang et al. (Liang et al., 2014) propose a model that jointly learns feature representations and decisions; Gao et al. (Gao et al., 2021) present SMP-Graph, an unsupervised semantic graph with structural cues that improves procedure coding; and Jia et al. (Jia et al., 2017) design a spatiotemporal autoencoder that learns compact patient representations from large, partially missing records, improving classification.

[1]The data visualization utilizes an intermediate form known as data visualization query (DVQ) that could be executed to obtain the specifications and charts, following existing practice of general text-to-visualization studies (Song et al., 2022; Luo et al., 2021).

GenoREC (Pandey et al., 2023) recommends genomic visualizations based on the data and task.

Text-to-SQL is widely used to query EMRs (Wang et al., 2020; Raghavan et al., 2021; Lee et al., 2024), but it mainly retrieves fields for specific questions and does not support broader analysis. Text-to-visualization targets analysis directly: it helps users find patterns without handling schema or chart details and can support non-experts. However, existing datasets and methods rarely focus on the medical domain. Given the richness of EMRs, specialized text-to-visualization resources are needed to improve clinical analysis and research.

### 2.2 Text-to-Visualization

Early systems relied on rule-based NLP (heuristics, grammars, probabilistic models). FlowSense (Yu and Silva, 2020) applies semantic parsing to data-flow editing and provides dataset and chart hints. NL4DV (Narechania et al., 2021) parses natural language into a JSON visualization specification. These approaches are limited by fixed rules. Recent work adopts neural methods: ADVisor (Liu et al., 2021) uses a two-stage pipeline for SQL retrieval and visualization grammar generation; NvBench (Luo et al., 2021) enables training transformer models such as ncNet (Luo et al., 2022) for cross-domain visual query translation. Hybrid neurosymbolic methods also emerge: RGVisNet (Song et al., 2022) combines retrieval over code patterns with neural generation, and Prompt4Vis (Li et al., 2024) uses context-aware few-shot prompting.

Visualization for electronic medical records remains underexplored. We address this gap by building a benchmark for medical data visualization and evaluating DNN- and LM-based methods on it, providing a foundation for future work on EMR visualization.

## 3 Dataset Construction

In this section, we introduce our proposed framework that leverages the capabilities of LLMs to enable the automatic generation of data for visualizing EMRs.

### 3.1 The Overall Pipeline

The overall data generation pipeline, as shown in Fig. 2, consists of the following three main steps. The first stage is DVQ candidate generation, where we use prompt techniques from LLMs to translate structured SQL queries into understandable and analyzable DVQs. The next stage is DVQ filtering and checking, where we employ a rule-based filter to screen the candidate visualization queries. This step involves filtering out overly complex, cumbersome, and unreasonable DVQ statements, ensuring that only high-quality and informative visualization results are retained. Through this process, potentially misleading or incomplete visualization queries generated by the model are eliminated. The final stage is NLQ generation, which focuses on transforming the generated data visualization results into understandable natural language questions. In this stage, we manually align the generated natural language with the subset of filtered DVQ and conduct matching tests to ensure that the generated natural language is fluent and the syntax directing to the DVQs is entirely correct. In the following sections, we provide detailed descriptions and explanations for each of the above steps.

### 3.2 DVQ Candidate Generation

The use of LLMs provides a convenient way for us to transform SQL queries. We use prompt methods to convert SQL queries into DVQs, as depicted in Step 1 of Fig. 2. The initial transformation of an SQL query $Q_{sql}$ into a visualization query $Q_{dvq}$ can be formally described as follows: an SQL query $Q_{sql}$ is provided as input to the model, using LLMs such as GPT-4, the query $Q_{sql}$ is parsed and restructured into a visualization query $Q_{dvq}$. This process can be represented as:

$$Q_{dvq} = \text{LLM}(Q_{sql}; P_{S2D}) \quad (1)$$

Where LLM denotes the Large Language Model that interprets and converts the SQL query and $P_{S2D}$ denotes our appropriately crafted SQL-to-DVQ prompt. The resulting $Q_{dvq}$ provides the instructions for generating the desired data visualization.

To achieve this, the instructional prompt $P_{S2D}$ is utilized, as shown in the SQL-to-DVQ Prompt example in Fig. 2. The prompt guides the model on how to transform SQL queries into DVQs, with the aim of preserving the fundamental elements of the SQL query while reformatting it into a DVQ suitable for visualization purposes. For instance, we convert SQL statements like : *'SELECT **"DISCHTIME"**, count("DAYS_STAY") FROM **DEMOGRAPHIC** WHERE "SUBJECT_ID" = "9575"*

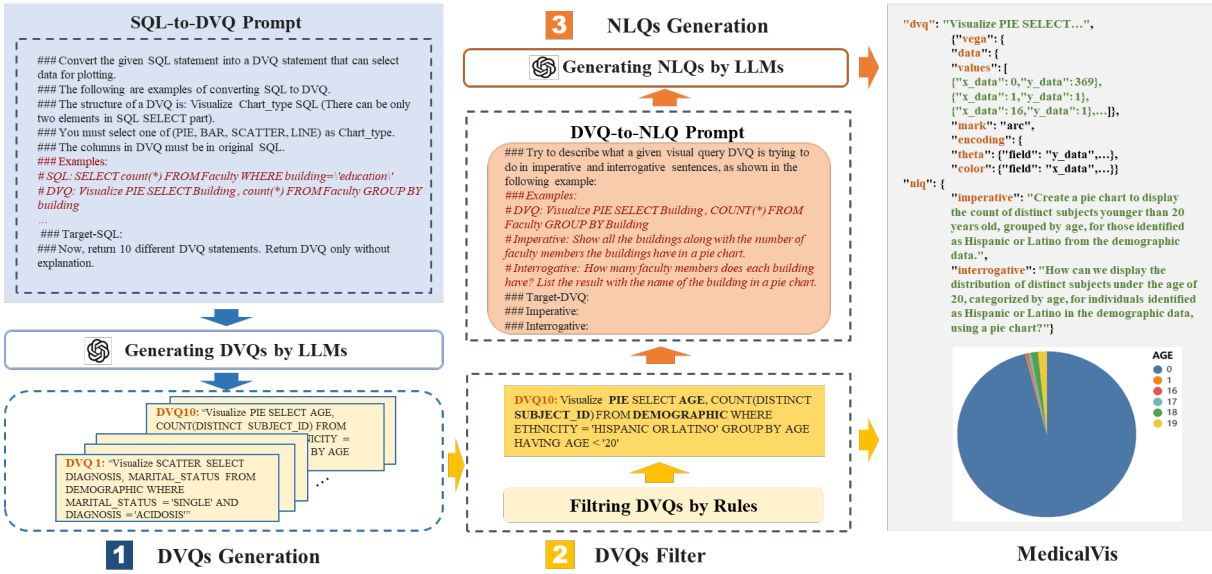



Figure 2: The proposed pipeline for creating an EMR text-to-visualization benchmark dataset. The pipeline comprises three key stages: (i) *DVQs Generation* from EMR data, (ii) *DVQs Filter* to check quality and ensure high-quality visualizations, and (iii) *NLQs Generation* to translate from data visualizations to textual descriptions by LLMs as annotators.

*GROUP BY "DISCHTIME" '* into DVQ statements like ***'Visualize SCATTER** SELECT **"DISCHTIME"**, count ("DAYS_STAY") FROM **DEMOGRAPHIC** WHERE "SUBJECT_ID" = "9575" GROUP BY "DISCHTIME" **BIN "DISCHTIME" BY MONTH'***. In this example, the SQL query $Q_{sql}$ is to count the number of "DAYS STAY" corresponding to DISCHTIME. The LLM transforms it into a visualization query $Q_{dvq}$, specifically a scatter chart, and additional pairs of months to group DISCHTIME.

### 3.3 DVQs Filtering and Checking

We apply a rule-based filter $R(\cdot)$ to enforce quality screening on candidate DVQs, whose decision logic is defined as: for a given DVQ $q$, if $R(q) =$ true, it means that the visualization meets the quality criteria, otherwise it is low quality visualization. This method establishes the valid interval of data points $\mathcal{D}_{\text{valid}} = [d_{\min}, d_{\max}]$, Where $d_{\min}, d_{\max}$ are selected based on the distribution characteristics of historical datasets, ensuring that the number of data points used for generating visualizations is constrained to a constant order of magnitude, rather than growing exponentially. During the processing of candidate set $T_Q$, the system dynamically eliminates all low quality instances satisfying $R(q) =$ false by executing $R(q)$ judgment item by item. The resulting refined collection $T'_Q$. This process ensures that the output collection satisfies domain specification conformance.

### 3.4 NLQs Generation

To generate NLQ text from visualizations, a systematic and comprehensive analysis of their constituent elements is necessary to articulate their meaning accurately.

First, it is crucial to identify the main theme or purpose of the visualization by carefully examining components such as the title, axis labels, and other textual elements. Subsequently, breaking down the visual components—such as bar charts, line graphs, pie charts, etc.—and analyzing their relationships, along with the scales used, becomes essential. For instance, in a bar chart, taller bars signify higher values. Trends and patterns can be discerned by observing changes in values over time. These observations form the foundation for constructing sentences and paragraphs that elucidate the key insights in the visualization. It is important to describe patterns, trends, and insights using precise language, referencing specific data points

| Data | Value |
|---|---|
| # of tables involved | 5 |
| # of columns involved | 49 |
| # Average DVQ length | 21 |
| # Average aggregation columns | 1.10 |
| # Average conditions | 1.41 |

Table 1: Statistics of MedicalVis dataset.

and comparison values. The ultimate objective is to convey complex visual information clearly and concisely. The DVQ-to-NLQ prompt design, illustrated in Fig. 2, aims to align natural language with visual effects while being compatible with human usage patterns.

## 4 Dataset Analysis

### 4.1 Overall Statistics

MedicalVis dataset includes 35,374 (NLQ, DVQ) pairs, and basic statistics are provided in Table 1. The dataset consists of DVQs distributed over 5 tables with a total of 49 columns. This variation allows researchers to explore different levels of data granularity. On average, each DVQ has 1.10 aggregated columns and includes 1.41 conditions. This enables comprehensive summary and analysis operations, including calculations for averages, sums, and other statistical measures essential for medical analytics.

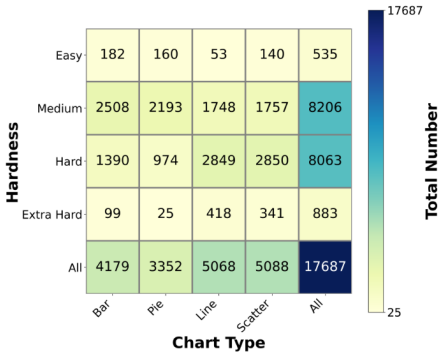



Figure 3: Visualization types vs. Hardness

### 4.2 Detailed Analysis

The MedicalVis dataset contains 35,374 (NLQ, DVQ) pairs and 17,687 distinct visualizations. Each DVQ is associated with two corresponding NLQs: one for an imperative sentence and one for an interrogative sentence. The following sections provide a more detailed discussion of the dataset.

**Visualization types.** MedicalVis contains 17,687 distinct visualizations across four types: bar (histogram), pie, line, and scatter. The dataset is carefully balanced to ensure that each visualization type is well-represented, with 4,179 bar charts, 3,352 pie charts, 5,068 line charts, and 5,088 scatter plots. Scatter plots are the most common, comprising nearly 29% of the total, while pie charts are the least frequent, accounting for 19%.

**Visualization hardness.** Intuitively, not all NLQ2DVQ tasks are equally difficult. Similar to nvBench, we categorize DVQs into four levels: easy, medium, hard, and extra hard. To evaluate the complexity of DVQs, we introduce a scoring framework that categorizes DVQs into different levels based on their structural components. Each component contributes to the overall complexity score $S_{\text{total}}$, which is calculated by aggregating scores from the following clauses: $S_{\text{select}}$ (number of aggregators and distinct columns in the SELECT part), $S_{\text{table}}$ (number of JOIN operations in the FROM part), $S_{\text{group}}$ (number of clauses in the GROUP BY part), $S_{\text{order}}$ (number of clauses in the ORDER BY part), $S_{\text{conditions}}$ (number of conditions in the WHERE, HAVING and LIMIT part), and $S_{\text{bin}}$ (1 if there is a BIN BY part in DVQ, otherwise 0).

$S_{total}$ is computed by summing up the scores of all the clauses we mentioned about. The difficulty levels of the questions can be categorized as: Easy ($S_{total} \leq 2$) , Medium ($2 < S_{total} \leq 4$), Hard ($4 < S_{total} \leq 6$), and Extra Hard ($S_{total} > 6$).

| Vis Types | #-vis | Avg.#-W | Max#-W | Min#-W |
|---|---|---|---|---|
| **Bar** | 4179 | 23 | 52 | 10 |
| **Pie** | 3352 | 21 | 52 | 11 |
| **Line** | 5068 | 24 | 63 | 10 |
| **Scatter** | 5088 | 25 | 60 | 11 |
| **All Type** | 17687 | 23 | 63 | 10 |

Table 2: NLQ Types and Word Counts (#-W) in MedicalVis

**NLQs.** Table 2 displays the number of DVQs of different visualization types and the length of the NLQs of the corresponding type. On average, each NLQ consists of approximately 23 words. Some NLQs have higher word counts due to the complexity of the visual query, which may involve filtering and join operations. At the same time, in the medical domain, efforts to ensure the unambiguity of NLQ expressions generally lead to longer NLQs.

## 5 Method

In this section, we explore the feasibility of generating visualizations from natural language descriptions in EMRs. As illustrated in Fig. 4, we conducted a mixed-task pretraining of the MedCodeT5 model, incorporating four key tasks: including NLQ2DVQ, DVQ2NLQ, sequence-to-sequence causal language modeling (CLM) and masked span prediction (MSP). This comprehensive pretraining approach enables the model to learn meaningful context representations and effectively recover missing information within complex DVQs.

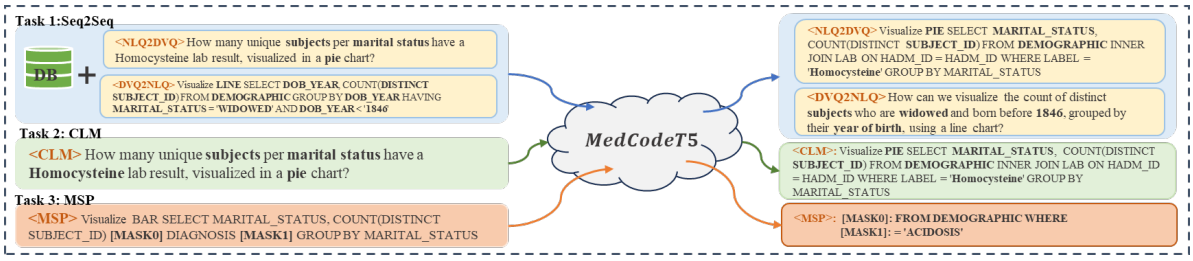



Figure 4: Illustration of the multi-task fine-tuning approach for the MedCodeT5 model. The approach encompasses: (i) sequence-to-sequence (Seq2Seq) transformations, which convert natural language queries to data visualization queries (NLQ2DVQ) and vice versa (DVQ2NLQ), (ii) sequence-to-sequence causal language modeling (CLM), and (iii) Masked Span Prediction (MSP)

Given the widespread adoption of LMs in NLP tasks, the CodeT5 model developed by Wang et al. (Wang et al., 2021) is well-suited for code-related tasks and shows strong potential for improving performance. To further enhance its capabilities in medical data visualization, we optimized CodeT5 using the MedicalVis dataset, resulting in a specialized version called MedCodeT5, focused on generating both medical DVQs and NLQs. In training MedCodeT5, we employed a multi-task learning approach, incorporating sequence-to-sequence transformation, CLM (Wu and Varshney, 2024; Tay et al., 2023), and MSP (Lee et al., 2021) tasks to improve language generation.

To enhance the generative capabilities of our MedCodeT5 model, we implemented a multi-task learning (MTL) framework inspired by the gradient signal-to-noise ratio (GSSNR) algorithm proposed by Peng et al. (Peng et al., 2024). The GSSNR for each task measures the consistency between the training sample gradients and the true gradient. A larger GSSNR indicates a stronger alignment between the sample gradients and the true gradient, while a smaller GSSNR suggests a more scattered relationship. This serves as a task reweighting mechanism based on sample gradient similarity. However, unlike (Peng et al., 2024) approach, we utilize only task weights, omitting sample weights. This decision is based on the fact that, in our multi-task training setup, the same sample is used across different tasks. As a result, the gradient similarities for the same sample may become too similar across tasks, leading to nearly identical sample weights, which can negatively impact the accuracy of the overall loss calculation. In this MTL setup, we optimize the model using a combination of sequence-to-sequence, CLM, and MSP tasks, with task weights dynamically adjusted during training to improve convergence and task synergy. Our approach incorporates the following steps for parameter updates:

$$\alpha_t = \langle \nabla_\theta L_t, \nabla J_u \rangle \tag{2}$$

where $\alpha_t$ represents the gradient similarity between the loss $L_t$ for task $t$ and the overall loss $J_u$. Based on this similarity, we update the task weights $w_t$ as follows:

$$w_t = \frac{\text{GSSNR}_t}{C}, \quad \text{GSSNR}_t = \frac{\mathbb{E}[\alpha_t]^2}{\text{Var}[\alpha_t]} \tag{3}$$

These weights are used to modify the model parameters $\theta$ through gradient descent updates at each iteration:

$$\theta^{k+1} = \theta^k - \eta \sum_{t=1}^{T} w_t \nabla_\theta L_t \tag{4}$$

By leveraging this dynamic weighting approach, the model adjusts its focus on tasks, optimizing for better overall performance.

In MedCodeT5, we leverage a sequence-to-sequence framework integrated with CLM tasks to enhance the generation of coherent and contextually accurate queries. The seq2seq tasks employ an encoder-decoder architecture based on Transformer models to map input queries to output visualizations. Meanwhile, the CLM tasks focus on predicting subsequent tokens and improving fluency and coherence in text generation. By training both tasks concurrently, the model not only learns to map sequences effectively but also generates fluent and contextually consistent language. In line with strategies from CodeT5+(Wang et al., 2023), we randomly select pivot positions within the CLM tasks, constraining them to between 10% and 80% of the sequence length. A special token [CLM] is prefixed to each sequence to signify the task, further optimizing the model's ability to generate accurate visualizations from medical

| Method | Validation Set | | | | Test Set | | | |
|---|---|---|---|---|---|---|---|---|
| | Data | Axis | Vis_type | Overall | Data | Axis | Vis_type | Overall |
| Seq2Vis | 31.6 | 19.1 | 70.7 | 3.8 | 33.6 | 22.6 | 68.2 | 4.5 |
| ncNet | 84.1 | - | 100.0 | 58.1 | 85.3 | - | 100.0 | 55.9 |
| CodeT5 | 97.6 | 52.4 | 100.0 | 66.0 | 97.5 | 52.0 | 100.0 | 64.3 |
| **MedCodeT5** | **97.8** | **52.4** | **100.0** | **66.2** | **97.6** | **52.0** | **100.0** | **65.9** |
| **MedCodeT5** (w/o GSSNR) | 97.0 | 52.7 | 100.0 | 61.8 | 96.7 | 52.0 | 100.0 | 61.5 |

Table 3: Matching Accuracy (%) on Validation and Test Sets

queries. These combined methodologies have enabled MedCodeT5 to be robustly trained to meet the distinct challenges of medical data visualization and NLQ generation, resulting in improved accuracy and more resilient language production capabilities. This optimization empowers MedCodeT5 to deliver superior performance, specifically tailored to medical data visualization and NLQ generation. To further refine the model's ability, we introduce an MSP task, where spans of tokens in the input sequence are masked and the model is tasked with predicting the masked portions. This further enhances the model's ability to generate structured, contextually accurate queries.

## 6 Experiments and Analysis

In this section, we detail our experimental setup and present the evaluation results. These experiments demonstrate the effectiveness and feasibility of the MedicalVis dataset in tackling complex text-to-visualization tasks, as well as highlight the superiority of the MedCodeT5 model in handling EMR-related tasks.

### 6.1 Experimental Setup

#### 6.1.1 Dataset

According to the segmentation of the MIMICSQL dataset, we performed a similar partitioning of our MedicalVis dataset, dividing it into three subsets: a training set, a validation set, and a test set. The training set comprised 80% of the total dataset, while the validation and test set each accounted for 10%.

#### 6.1.2 Methods

We choose the following methods since they are the most popular ones in text-to-visualization fields.

- *Seq2Vis (Luo et al., 2021)*: Seq2Vis is a widely used model that utilizes a standard encoder-decoder architecture with an attention mechanism for enhanced performance.
- *ncNet (Luo et al., 2022)*: ncNet is a Transformer-based model that allows users to select chart templates as additional input. This enables the model to incorporate information from the chosen chart templates into the natural language processing.
- *CodeT5 (Wang et al., 2021)*: CodeT5 is a Transformer-based model designed for code generation and understanding, leveraging multi-task pre-training to excel in tasks such as code completion and translation.
- *MedCodeT5*: MedCodeT5 excels in converting NLQ to DVQ and vice versa, facilitating seamless interaction with medical databases. Its robust pretraining on diverse tasks ensures high performance in understanding and generating medical data visualizations.

#### 6.1.3 Evaluation Metrics

In order to better explore the performance of our dataset across different models, we have employed exact match accuracy as a measure for the components used in visualizations, which include visualization matching accuracy, data value matching accuracy, axis matching accuracy, and overall matching accuracy.

$$M_{\text{vis}} = \frac{V_{\text{correct}}}{N_{\text{total}}}, M_{\text{data}} = \frac{D_{\text{correct}}}{N_{\text{total}}}, M_{\text{axis}} = \frac{A_{\text{correct}}}{N_{\text{total}}}$$

where $M$ means match accuracy, $V$, $D$ and $A$ represent visualization chart type, data, horizontal and vertical axes. $N_{\text{total}}$ indicates the total number of data items, that is, the total number of DVQs in the dataset or the overall data items after Vega-Lite transfer. The overall matching accuracy calculation follows the same procedure, where we determine the overall matching degree by assessing the exact matching between the predicted DVQ and the

| Method | Validation Set | | | | Test Set | | | |
|---|---|---|---|---|---|---|---|---|
| | Easy | Medium | Hard | Ex. Hard | Easy | Medium | Hard | Ex. Hard |
| Seq2Vis | 29.2 | 12.5 | 1.56 | 0 | 33.3 | 14.9 | 1.86 | 0 |
| ncNet | 45.8 | 35.2 | 16.5 | 0 | 44.7 | 36.7 | 14.1 | 0 |
| CodeT5 | 82.3 | **68.6** | 65.3 | 37.3 | 79.8 | **69.6** | 63.2 | 28.7 |
| **MedCodeT5** | **84.4** | 68.4 | **65.6** | **39.4** | **78.1** | 69.5 | **64.6** | **29.8** |
| **MedCodeT5** (w/o GSSNR) | 81.3 | 66.2 | 58.8 | 26.8 | 76.3 | 67.6 | 60.2 | 21.8 |

Table 4: Hardness Comparison (%)

ground truth. It is worth mentioning that we are using a strict matching metric, even if the data value in the SQL part of the DVQ is wrong, or if the column name A and column name B appear in reverse order, we will determine a mismatch as a whole.

## 6.2 Experimental Analysis

### 6.2.1 Performance Comparison

We conducted a comparative analysis of various benchmarks for DNN-based and LM-based models using our dataset, including Seq2Vis, ncNet, CodeT5, and MedCodeT5. A key strength of our study lies in constructing a diverse and challenging dataset for medical DVQ generation. By incorporating real-world cases, we ensure that models are evaluated under practically relevant conditions, reflecting challenges in clinical decision support.

Table 3 presents the performance of different methods in generating data visualization queries. CodeT5 and others adopt single-task fine-tuning on NLQ2DVQ, while MedCodeT5 employs multi-task training. MedCodeT5 (w/o GSSNR) omits task weighting, simply summing multi-task losses. Overall, MedCodeT5 achieves the highest accuracy (65.9%) with an axis accuracy of 97.2%. Both MedCodeT5 and CodeT5 attain high data-specific accuracy, reflecting the robustness of pre-trained models. However, naïve multi-task training (MedCodeT5 w/o GSSNR) underperforms CodeT5, indicating that multi-task learning alone is insufficient. GSSNR optimizes task generalization, enabling MedCodeT5 to handle diverse queries effectively. Notably, MedCodeT5 excels in more complex queries requiring reasoning and multi-table joins, a challenge explored further in the next section.

### 6.2.2 Detailed Analysis

As shown in Table 4, MedCodeT5 consistently outperforms all other methods across difficulty levels. It achieves a 65.9% overall score on the test set and a notable 29.8% on extremely hard problems, demonstrating its robustness. In contrast, CodeT5 falters on complex tasks, ncNet is effective only on simpler queries, and Seq2Vis struggles significantly, achieving the lowest scores. Further details are in Appendix A.

These performance gaps are rooted in the inherent complexity of medical database query (DVQ) generation, which requires reasoning about **implicit clinical logic**, **multi-table joins**, and **time series analysis**. These challenges expose critical limitations in existing models: Seq2Vis lacks a proper syntactic framework; ncNet suffers from domain knowledge deficiencies, leading to incorrect field and table selection; and even the general-purpose CodeT5 exhibits semantic mapping biases for critical medical terms.

MedCodeT5 addresses these issues by integrating domain-specific knowledge. This design directly improves medical term alignment, table relationship understanding, and logical consistency. Our experiments confirm that this domain-aware architecture provides a significantly more robust and accurate solution for medical DVQ generation.

# 7 Conclusion

We propose a novel pipeline to create a text-to-visualization benchmark dataset tailored for EMRs. This dataset enables users to generate visualizations from NLQs about EMR data. It consists of paired medical records, NLQs, and corresponding visualizations. To alleviate manual annotation efforts, we leverage LLMs for data creation. We evaluated state-of-the-art text-to-visualization techniques and LLM-based approaches on these new datasets, confirming the feasibility of generating EMR visualizations from NLQs. Our work facilitates standardized assessment of EMR visualization methods, advancing this impactful application field and promoting the elicitation of medical insights through visualization.

## 8 Limitations

Despite providing the first large-scale benchmark and methodological validation for EMR-oriented text-to-visualization, this study has several limitations that warrant further work.

**Data construction and annotation quality.** MedicalVis primarily relies on large language models (LLMs, e.g., GPT-4) to convert SQL to DVQ and then to generate NLQ, with rule-based filtering producing the final samples. This pipeline reduces manual effort but introduces risks of systematic bias and noise inherent in synthetic data: for example, the model may conflate "plausible" visualizations with factually correct ones; it may be insufficiently sensitive to nuances in clinical abbreviations, temporal semantics, and medical terminology; and filtering thresholds and heuristics can shift the sample distribution, weakening coverage of long-tail and complex queries. In addition, dependence on closed-source models imposes practical constraints on reproducibility and cost.

**Potential biases in difficulty tiers and filtering.** We use a structure-driven difficulty partition (easy/medium/hard/very hard) and quality filters to improve data consistency and executability. However, this design may underrepresent high-complexity tasks encountered in *real-world* settings, limiting the evaluation's coverage of intricate temporal constraints, cross-table relations, and implicit clinical logic. Moreover, difficulty is defined mainly by syntactic structure rather than semantic ambiguity or uncertainty in user intent—factors that are crucial for clinical QA and exploratory analysis.

**System integration and compliance.** This work verifies the technical feasibility of translating NLQ to DVQ, but it does not provide end-to-end integration with production EMR systems or assessments of interactive, online use. Requirements for *deployment-level* privacy, security, and compliance—such as de-identification, principles of minimal visual disclosure, and audit trails—have not been addressed systematically. Alignment with best practices in data visualization libraries (DVLs, e.g., Vega-Lite, ECharts)—including annotation, color usage, and safeguards against misleading encodings—as well as editable and traceable mechanisms tailored to clinical workflows, also remains to be strengthened.

In summary, while this study establishes a benchmark and methodological foundation for EMR text-to-visualization, supporting high-stakes clinical scenarios will require advances in data diversity, human-factors evaluation, compliance modules, system integration, and cross-domain robustness.

## References


Fengquan Cai, Dong Cao, and Hui Ye. 2023. Graph Representation of Chinese EMRs and its Short-text Classification Application1 . In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 4513–4519, Los Alamitos, CA, USA. IEEE Computer Society.

Anamaria Crisan, Shannah E. Fisher, Jennifer L. Gardy, and Tamara Munzner. 2022. Gevitrec: Data reconnaissance through recommendation using a domain-specific visualization prevalence design space. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4855–4872.

Jimenez Eladio and Hao Wu. 2024. emrqa-msquad: A medical dataset structured with the squad v2.0 framework, enriched with emrqa medical information. *Preprint*, arXiv:2404.12050.

Scott L. Fleming, Alejandro Lozano, William J. Haberkorn, Jenelle A. Jindal, Eduardo Reis, Rahul Thapa, Louis Blankemeier, Julian Z. Genkins, Ethan Steinberg, and et al. Nayak. 2024. Medalign: A clinician-generated dataset for instruction following with electronic medical records. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22021–22030.

Yue Gao, Xiangling Fu, Xien Liu, Kaiyin Zhou, and Ji Wu. 2021. Smp-graph: Structure-enhanced unsupervised semantic graph representation for precise medical procedure coding on emrs. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1303–1308.

Yao Jia, Chongyu Zhou, and Mehul Motani. 2017. Spatio-temporal autoencoder for feature learning in patient data with missing observations. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 886–890.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jongyeup Kim, and Edward Choi. 2024. Ehrsql: a practical text-to-sql benchmark for electronic health records. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Gyubok Lee, Seongjun Yang, and Edward Choi. 2021. Improving lexically constrained neural machine translation with source-conditioned masked span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and*

*the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 743–753, Online. Association for Computational Linguistics.

Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, and Wei Chen. 2018. Echarts: A declarative framework for rapid construction of web-based visualization. *Visual Informatics*, 2(2):136–146.

Shuaimin Li, Xuanang Chen, Yuanfeng Song, Yunze Song, and Chen Zhang. 2024. Prompt4vis: Prompting large language models with example mining and schema filtering for tabular data visualization. *Preprint*, arXiv:2402.07909.

Znaonui Liang, Gang Zhang, Jimmy Xiangji Huang, and Qmming Vivian Hu. 2014. Deep learning for healthcare decision making with emrs. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 556–559.

Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20.

Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 1235–1247, New York, NY, USA. Association for Computing Machinery.

Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2022. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226.

Arpit Narechania, Arjun Srinivasan, and John Stasko. 2021. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Aditeya Pandey, Sehi L'Yi, Qianwen Wang, Michelle A. Borkin, and Nils Gehlenborg. 2023. Genorec: A recommendation system for interactive genomics data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):570–580.

Xinyu Peng, Cheng Chang, Fei-Yue Wang, and Li Li. 2024. Robust multitask learning with sample gradient similarity. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(1):497–506.

Preethi Raghavan, Jennifer J Liang, Diwakar Mahajan, Rachita Chandra, and Peter Szolovits. 2021. emrKBQA: A clinical knowledge-base question answering dataset. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 64–73, Online. Association for Computational Linguistics.

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350.

Yuanfeng Song, Xuefang Zhao, Raymond Chi-Wing Wong, and Di Jiang. 2022. Rgvisnet: A hybrid retrieval-generation neural framework towards automatic data visualization generation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1646–1655, New York, NY, USA. Association for Computing Machinery.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*.

Ping Wang, Tian Shi, and Chandan K. Reddy. 2020. Text-to-sql generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*, WWW '20, page 350–361, New York, NY, USA. Association for Computing Machinery.

Yue Wang, Hung Le, Akhilesh Gotmare, Nghi Bui, Junnan Li, and Steven Hoi. 2023. CodeT5+: Open code large language models for code understanding and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1069–1088. Association for Computational Linguistics.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Faustine Williams and Suzanne Austin Boren. 2008. The role of the electronic medical record (emr) in care delivery development in developing countries: a systematic review. *Informatics in primary care*, 16(2).

Xinbo Wu and Lav Varshney. 2024. A meta-learning perspective on transformers for causal language modeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15612–15622, Bangkok, Thailand. Association for Computational Linguistics.

Bowen Yu and Claudio T. Silva. 2020. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–11.

# A Appendix

## A.1 Dataset Description

In addition to the MedicalVis dataset, which was constructed using the LLM described in the main text, we also created a subset of MedicalVis called H-MedicalVis. In this section, we provide a detailed explanation of the creation process of the artificial dataset and present the visualization results across different difficulty levels, as shown in Figure 5. Furthermore, we discuss the manual step involved in generating the dataset.

First, we constructed an artificial dataset containing 2,000 entries. The difficulty distribution of the artificial dataset closely aligns with that of the generated dataset. This ensures the rationality and aesthetics of the generated Vega-Lite visualizations, building on the successful rendering of Vega-Lite in the original generated dataset. The DVQs in the artificial dataset were corrected based on the original generated dataset. Moreover, to address the issue of similar syntactic structures in the generated dataset, we added one manually written NLQ to the original two NLQs, as shown in the H-Data of Table 5, to ensure diversity in semantic interpretations. Furthermore, to resolve potential confusion between the "WHERE" and "HAVING" clauses in the DVQs of the generated dataset, we conducted thorough screening and manual corrections to ensure the quality of the dataset.

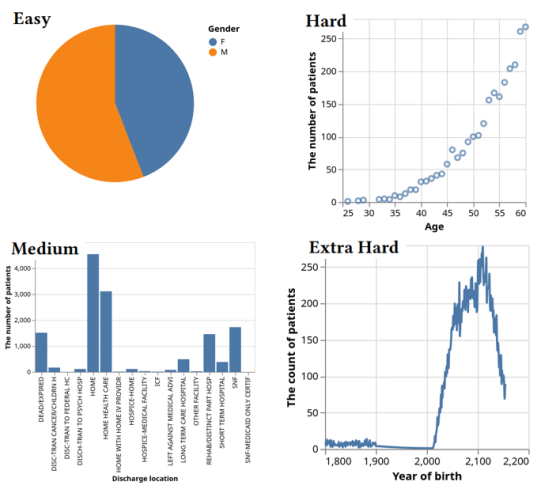



Figure 5: Visualization of DVQ from MedicalVis with Different Levels of Difficulty

Table 5: Comparison of Generated Data and Artificial Data

| Dataset Version | Generate Data | Artificial Data |
|---|---|---|
| DVQ | Visualize BAR SELECT EXPIRE_FLAG, COUNT(DISTINCT SUBJECT_ID) FROM DEMOGRAPHIC INNER JOIN DIAGNOSES ON HADM_ID = HADM_ID GROUP BY EXPIRE_FLAG ORDER BY COUNT(DISTINCT SUBJECT_ID) | Visualize PIE SELECT ADMISSION_TYPE, COUNT(DISTINCT SUBJECT_ID) FROM DEMOGRAPHIC GROUP BY ADMISSION_TYPE |
| NLQ | Generate a bar chart to show the count of unique subjects grouped by their expire flag after joining DEMOGRAPHIC and DIAGNOSES tables based on hospital admission ID. | How can we visualize the distribution of unique patients by admission type using a pie chart? |
| | | Create a pie chart showing the number of unique patients by admission type from the DEMOGRAPHIC table. |
| | How can one visualize the number of unique subjects distinguished by their expire flag, utilizing a bar chart, after merging DEMOGRAPHIC and DIAGNOSES tables on the hospital admission ID? | Show me a pie chart of the number of unique subjects for each admission type. (H-Data) |

## A.2 Implementation Details

The MedCodeT5 model was trained using the Adam optimizer, with CodeT5-small as the base model. The mini-batch size was set to 8, and the learning rate to 1e-5, with hyperparameter settings based on previous studies. Since CodeT5 serves as the base model for MedCodeT5, which excels in multi-task training, we did not separate the task-specific and parameter-sharing layers in the model. The gradient similarity loss for the MedCodeT5 model is computed using only the top-level parameters, such as those from the final Transformer block. The experiments were conducted on a server equipped with 8 NVIDIA L40 GPUs and 439 GB of memory, featuring a multi-core high-performance CPU and running the CentOS operating system. All methods were implemented using Python 3.10.

## A.3 Case Study

In this section, we demonstrate the practical application of our EMR visualization dataset and compare the performance of MedCodeT5 with other baselines through a concrete case study in Table 8. Given the NLQ: "Create a scatter plot to display the count of hospital admissions for each age group under 80 with a history of venous thrombosis or embolism", our DVQ evaluation focuses on several key aspects. First, identifying patients under 80 with a history of venous thrombosis or embolism is critical. The generated DVQ must then accurately summarize and analyze admissions by age group before transforming the data into a scatter plot for clear visualization.

Seq2Vis generates an incomplete DVQ, missing a condition and the GROUP BY clause, reflecting its broader challenges in handling DVQ syntax and query complexity. Meanwhile, ncNet struggles with incorrect column and condition selection. CodeT5 produces a nearly correct DVQ but errs in condition column selection. In contrast,

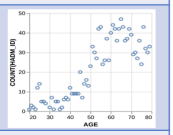
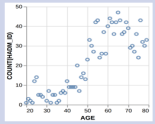
Table 6: Validation Set Error Analysis by Error Type

| Method | Err_value | Err_element | Err_condition | Err_keyword |
|---|---|---|---|---|
| Seq2Vis | 68.4 | 30.2 | 87.9 | 43.1 |
| CodeT5 | 2.4 | 9.6 | 17.1 | 10.1 |
| MedCodeT5 (w/o GSSNR) | 3.0 | 10.0 | 22.3 | **9.5** |
| **MedCodeT5** | **2.2** | **9.5** | **16.8** | 10.1 |

Table 7: Test Set Error Analysis by Error Type

| Method | Err_value | Err_element | Err_condition | Err_keyword |
|---|---|---|---|---|
| Seq2Vis | 66.4 | 32.9 | 85.8 | 44.8 |
| CodeT5 | 2.5 | 10.6 | 15.2 | 12.3 |
| MedCodeT5 (w/o GSSNR) | 3.3 | 10.7 | 20.2 | **12.0** |
| **MedCodeT5** | **2.4** | **10.4** | **14.3** | 12.2 |

Table 8: Case Study: DVQs generated by diverse text-to-visualization methods on MedicalVis, accompanied by their corresponding charts (Errors indicated in red).

| **NLQ** | **Create a scatter plot to display the count of hospital admissions for each age group under 80 years with a history of venous thrombosis or embolisms.** | |
|---|---|---|
| **Target DVQ** | Visualize **SCATTER** SELECT **AGE**, COUNT(**HADM_ID**) FROM **DEMOGRAPHIC** INNER JOIN **DIAGNOSES** ON **DEMOGRAPHIC.HADM_ID** = **DIAGNOSES.HADM_ID** WHERE **AGE** ≤ 80 AND **SHORT_TITLE** = **'Hx-ven thrombosis/embols'** GROUP BY **AGE** |  |
| **Seq2Vis** | Visualize SCATTER SELECT AGE , **COUNT ( * )** FROM DEMOGRAPHIC INNER JOIN DIAGNOSES ON DEMOGRAPHIC.HADM_ID = DIAGNOSES.HADM_ID WHERE AGE < 62 AND **SHORT_TITLE** = （✗） |  |
| **ncNet** | Visualize SCATTER SELECT AGE, COUNT(**AGE**) FROM DEMOGRAPHIC WHERE AGE < '80' AND **LONG_TITLE** = **'tietze's disease'** GROUP BY AGE （✗） |  |
| **CodeT5** | Visualize SCATTER SELECT AGE, COUNT(HADM_ID) FROM DEMOGRAPHIC INNER JOIN DIAGNOSES ON DEMOGRAPHIC.HADM_ID = DIAGNOSES.HADM_ID WHERE AGE < 80 AND **LONG_TITLE** = 'Hx-ven thrombosis/embols' GROUP BY AGE （✗） |  |
| **MedCodeT5** | Visualize **SCATTER** SELECT **AGE**, COUNT(**HADM_ID**) FROM **DEMOGRAPHIC** INNER JOIN **DIAGNOSES** ON **DEMOGRAPHIC.HADM_ID** = **DIAGNOSES.HADM_ID** WHERE **AGE** < 80 AND **SHORT_TITLE** = **'Hx-ven thrombosis/embols'** GROUP BY **AGE** （✓） |  |

MedCodeT5 generates a logically structured and accurate DVQ, aligning precisely with the NLQ requirements.

This case exemplifies broader trends observed in the error analysis presented in Table 6 and Table 7. Across both validation and test sets, MedCodeT5 consistently achieves the lowest error rates in numerical accuracy (Err_value), element matching (Err_element), condition parsing (Err_condition), and keyword placement (Err_keyword), thereby demonstrating its robust performance in DVQ generation. Conversely, Seq2Vis exhibits the highest error rates across all categories, particularly in Err_condition and Err_keyword, which is consistent with its inability to fully capture query conditions and structure as evidenced in the case study. CodeT5 delivers a balanced performance but falls slightly short of MedCodeT5, especially in terms of numerical precision and keyword handling. The absence of the GSSNR module in MedCodeT5 (w/o GSSNR) results in marginally higher error rates for conditions and keywords, underscoring the critical role of this module in refining syntax interpretation.

### A.4 Prompt

This section provides additional details about the prompt that are not fully presented in the main text. The prompt content is displayed in the Table 9.

Table 9: Prompt Details

**SQL to DVQ**
**Role:** SYSTEM
**Content:**
Convert the given SQL statement into a DVQ statement that can select data for plotting.
The following are examples of converting SQL to DVQ.
The structure of a DVQ is: Visualize Chart_type SQL (There can be only two elements in SQL SELECT part).
You must select one of (PIE, BAR, SCATTER, LINE) as Chart_type.
The columns in DVQ must be in original SQL.
Examples:
1. SQL: SELECT count(*) FROM Faculty WHERE building = “education”
DVQ: Visualize PIE SELECT Building , count(*) FROM Faculty GROUP BY building

2. SQL: SELECT count(*) FROM Faculty WHERE Sex = “femal" and Rank = “AssocProf"
DVQ: Visualize BAR SELECT Rank, count(*) FROM Faculty GROUP BY Sex, Rank

3. SQL: SELECT COUNT(*) FROM Apartments where apt_type_code = “Studio" ORDER BY COUNT(*) ASC
DVQ: Visualize PIE SELECT apt_type_code , COUNT(*) FROM Apartments GROUP BY apt_type_code ORDER BY COUNT(*)ASC

4. SQL: SELECT Classroom , Grade FROM list
DVQ: Visualize SCATTER SELECT Classroom , Grade FROM list

5. SQL: SELECT COUNT(EMP_FNAME) FROM employee WHERE emp_jobcode = “PROF" GROUP BY EMP_FNAME ORDER BY emp_dob
DVQ: Visualize BAR SELECT EMP_FNAME , COUNT(EMP_FNAME) FROM employee WHERE emp_jobcode = “PROF" GROUP BY EMP_FN AME ORDER BY emp_dob

6. SQL: SELECT date_of_birth , COUNT(date_of_birth) FROM Guests WHERE gender_code = “Male"
DVQ: Visualize LINE SELECT date_of_birth , COUNT(date_of_birth) FROM Guests WHERE gender_code = “Male" BIN date_of_birth BY YEAR

**Role:** USER
**Content:**
Given SQL: {SQL}
Now, return 10 different DVQ statements. Return DVQ only without explanation.

**DVQ to NLQ**
**Role:** SYSTEM
**Content:**
Visualization query is like SQL, it is used to retrieve data and visualize it.
Try to describe what given visualization query is trying to do in an imperative sentence and an interrogative sentence.
Make sure the chart type in the visualization query is in the sentences you give.
Note that there is BIN BY in visualization query.

**Role:** USER
**Content:**
Make sure you include the chart type in the visualization query in the sentences you give.
Given visualization query: {DVQ}
Now, return an imperative sentence and an interrogative sentence.