Saama Technologies at EHRSQL 2024: SQL Generation through Classification Answer Selector by LLM

Mohammed Jabir, Kamal Raj Kanakarajan, Malaikannan Sankarasubbu
Saama Technologies
{mohammed.jabir, kamal.raj, malaikannan.sankarasubbu}@saama.com

Abstract

The EHRSQL task aims to develop a dependable text-to-SQL model for Electronic Health Records (EHR) databases, which are crucial sources of clinical data that store patients' medical histories in hospitals. Large language models (LLM) have been proven to exhibit state-of-the-art performance for text-to-SQL tasks across various domains. To this end, we have developed a framework, SQL Generation through Classification Answer Selector by LLM (SCAS), which comprises two modules. The CAS module determines the answerability of the question, while the SG model generates the SQL query exclusively for answerable questions. Our system ranked 7th on the leaderboard with a Reliability Score of 53.21 on the official test set.

1 Introduction

Electronic Health Records (EHRs) are an essential component of modern healthcare. They store a patient's complete medical history, allowing hospital staff to make better clinical decisions (Wang et al., 2020; Bardhan et al., 2022) by quickly accessing relevant patient information. However, accessing this information can be time-consuming, especially when complex queries are involved. The traditional way of accessing EHRs involves using a predefined rule conversion system to convert user queries to SQL and retrieve the relevant information. This process can become a bottleneck for users who need to build custom queries or deal with complex queries. To address this issue, the EHRSQL (Lee et al., 2022) task aims to develop a system that can automatically translate user questions into corresponding SQL queries, making retrieving the information they need easier and quicker. The system's objective is to build a text-to-SQL system that converts natural language queries to SQL and informs users whether their queries are answerable.

Text-to-SQL tasks (Katsogiannis-Meimarakis and Koutrika, 2023) involve mapping natural language questions onto a given relational database into SQL queries. Early studies (Dong and Lapata, 2016; Wang et al., 2019) tackled this task with pre-defined rules or as a sequence-to-sequence task. However, recent advancements in large language models (LLMs) (Brown et al., 2020; OpenAI et al., 2024; Touvron et al., 2023) have become a milestone for natural language processing and machine learning. LLMs are pre-trained on massive text corpus, which enables them to perform various natural language tasks, and their ability to do in-context learning (Liu et al., 2021) makes them most suitable for text-to-SQL generation.

In this paper, we present our approach to tackling the EHRSQL 2024 (Lee et al., 2024) shared task, which involves a complex dataset of electronic health records. Our proposed framework, the SQL Generation through Classification Answer Selector by LLM (SCAS), helps avoiding incorrect SQL generation by using the Classification Answer Selector (CAS) module. The CAS module uses an LLM prompting method that incorporates the output of other classification models to generate the final classification output, thereby abstaining from incorrect responses. The SCAS framework generates SQL queries only for necessary questions by utilizing other LLM models. Our system achieved a 7th position on the leaderboard, with a Reliability Score of 53.21 on the official test set. The code to reproduce the experiments mentioned in this paper is publicly available[1].

2 Background

2.1 Task and Dataset Description

EHRSQL is a text-to-SQL task aiming to convert natural language queries into corresponding SQL queries while identifying untranslatable ones. The

[1] https://github.com/upjabir/ehrsql_2024

Proceedings of the 6th Clinical Natural Language Processing Workshop, pages 655–671

original EHRSQL (Lee et al., 2022) task was built on MIMIC-III (Johnson et al., 2016) and EICU (Pollard et al., 2018) datasets, which are available from Physionet (Goldberger et al., 2000). The EHRSQL dataset contains a wide range of questions across various domains in EHR, including Demographics, Prescription, Vital signs, and more, as well as Time Sensitive questions. The dataset for the task comprises 5124 training, 1163 validation, and 1162 testing samples.The competition dataset is derived from the MIMIC-IV (Johnson et al., 2023) open-access database demo subset and includes both answerable and unanswerable questions in the training set. The system should output corresponding SQL queries for answerable questions and null for unanswerable questions.

The Reliability Score (RS) is a new evaluation metric for text-to-SQL models used in the EHRSQL task. It rewards accurate SQL generation for certain types of questions while penalizing incorrect SQL generation for others. It does not assign any reward or penalty for abstaining from answering certain questions. The competition uses several scoring systems, including RS(0), RS(5), RS(10), and RS(N), with RS(10) being the primary metric for the leaderboard. In RS(10), correct predictions receive one positive point, while incorrect predictions receive -10 points. N in RS(N) represents the size of the test set.

2.2 Related Works

The text to SQL task poses a significant challenge and has previously been approached as a sequence-to-sequence task. (Brunner and Stockinger, 2021) utilized the BERT (Devlin et al., 2019) model as an encoder architecture to achieve state-of-the-art results in this task. They incorporated user questions and employed a neural network architecture to extract values and generate SQL queries. Another study of (Qi et al., 2022) demonstrates an innovative approach by incorporating various types of existing relations and co-references, thereby introducing new parameters to the encoder-decoder (Sutskever et al., 2014) architecture model.

Researchers have been utilizing the Language Model LLM for text-to-SQL since its emergence. Downstream tasks for LLM can be achieved through in-context learning and fine-tuning methods. (Wei et al., 2023) proposed a Chain of Thought style prompting technique to enhance the capabilities of LLM. (Pourreza and Rafiei, 2023) proposed a decomposed in-context learning method where the text-to-SQL task is divided into subtasks. On the other hand (Tai et al., 2023) introduced a new CoT-style prompting method specifically for text-to-SQL parsing, which showed significant improvements compared to standard prompting methods and the least-to-most prompting method. Additionally, a new prompt engineering method called DIAL SQL was proposed by (Gao et al., 2023), demonstrating the potential of fine-tuning LLMs for Text-to-SQL while highlighting the degeneracy of in-context learning capability after fine-tuning.

Text classification is a crucial task in machine learning, and it can be accomplished using classical machine learning models such as Random Forest and Deep learning models like Transformer (Vaswani et al., 2023), which is also effective in handling complex language tasks. With the emergence of LLM, which is trained on large text corpus, (Wang et al., 2023) suggests that the efficiency of text classification has been increased by using LLM as a zero-shot classifier. Although using LLM for downstream tasks is quite challenging. (Sun et al., 2023) addresses the difficulty of using LLM for downstream tasks by implementing effective prompting techniques, thereby improving the efficiency of LLM in text classification. On the other hand, (Zhang et al., 2024) overcomes this challenge by fine-tuning LLM, resulting in impressive performance surpassing in-context zero-shot learning capabilities of pre-trained LLM models like GPT 4 (OpenAI et al., 2024) in the healthcare domain.

3 System Overview

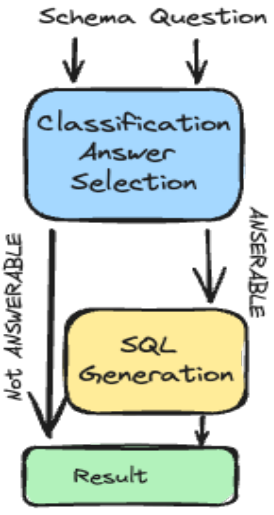

Figure 1: SCAS framework input and output flow.

Our research paper presents a novel framework

comprising two modules: The Classification Answer Selector (CAS) module and the SQL generation (SG) module. The CAS module is responsible for determining whether a question can be answered, while the SG module is designed to generate an SQL query for questions.

3.1 Classification Answer Selector Module

The CAS (Classification Answer Selector) module is a powerful tool that includes two distinct classification models and a selector to generate the final classification answer. The selector utilizes the advanced capabilities of Azure's OpenAI GPT-3.5-turbo (Brown et al., 2020) LLM to ensure the accuracy and comprehensiveness of the final answer.

The classification model uses a classical machine learning approach and methodology for the classification task. Feature selection is used to reduce the dataset's dimensionality by eliminating irrelevant features, with TF-IDF (Sparck Jones, 1972) as an effective methodology for text classification. Multiple classification models were employed, including MultinomialNB (Lewis, 1998), SGD (Robbins, 1951; Kiefer and Wolfowitz, 1952), and CatBoost (Dorogush et al., 2018), for ensemble classification using a weighted ensemble approach. Predictions were based on probabilities, with a threshold of 0.4 established to convert predicted probabilities into class labels.

$$ClassLabel = \begin{cases} 1 & \text{if probability} > 0.4 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The Second classification model is a fine-tuned LLM specifically for classification tasks. We utilize a pre-trained Large language model from the Codellama family (Rozière et al., 2024), specifically CodeLlama-7b-Instruct-hf[2], for this task. In this task, we have a large language model M and a training dataset D=$\{x_i,y_i\}$, where x_i represents the input prompt and y_i is the class label. The goal is to minimize the weighted cross-entropy loss, which is calculated by dividing the total number of instances in the training data by twice the count of positive or negative target values:

$$l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \quad (2)$$

[2] https://huggingface.co/meta-llama/CodeLlama-7b-Instruct-hf

The selector takes the outputs from both the first and second classification models and utilizes an in-context learning method to determine the final classification result. The prompt used for the selector is shown in figure 2. A full listing of examples is available in Appendix E

> Based on the database schema and table description, determine which AI assistant's answer accurately identifies whether the given question can generate an SQL query or not.
>
> ### Database Table Description
> The table name and its corresponding description are as follows:
> {table description}
>
> ### Database Schema
> This query will run on a database whose schema is represented in this string:
> {schema}
>
> {few shots}
> Question: "{question}"
> Ai Assitant 1's Answer: {model1 answer}
> Ai Assitant 2's Answer: {model2 answer}
> Answer: Let's think step by step.

Figure 2: Prompt for Classification Answer Selector.

3.2 SQL Generation Module

We employ the same pre-trained large language model of the Codellama family, which is used in classifier tasks for SQL generation. We perform instruction tuning only by considering the answerable questions from the EHRSQL 2024 dataset. The dataset, denoted as D=$\{x_i\}$, consists of input prompts where x_i represents the input prompt. The training objective is Causal language modeling. A full listing of prompts and examples are shown in Appendix D

4 Finetuning

We fine-tuned the model using the efficient parameter tuning method LoRA (Hu et al., 2022) and the HuggingFace library (Mangrulkar et al., 2022). The finetuning process for both the CAS and SQL generation modules involved using the AdamW optimizer and a cosine learning rate scheduler, targeting all linear layers within the model. We em-

Exp No	Development Phase	RS(0)	RS(5)	RS(10)	RS(N)
1	gpt-3.5-turbo	38.60	-261.47	-561.56	-69761.39
2	codellama FD	40.84	-250.21	-541.27	-67659.15
3	codellama SA + CL1	**74.97**	46.17	17.36	-6625.02
4	codellama SA + CL1*	42.73	40.15	37.57	**–557.26**
5	codellama SA + CL2	43.59	39.29	34.99	956.40
6	codellama SA + CAS	55.97	**49.52**	**43.07**	-1444.02
Exp No	**Test Phase**				
1	codellama SA + CAS	**77.03**	-36.07	-149.18	-26322.96
2	codellama SA + CAS*	53.21	**44.64**	**36.07**	**-1946.786**

Table 1: Experimental results during development phase and test phase. FD means Full data set used for training, SA means only Selected Answerable Questions. CL1 is the classical model, CL2 is codellama model, and CAS is the Classical Answer Selector Module. * Adjusted threshold.

ployed a maximum sequence length of 4096 tokens for training and inference for the SQL generation task, using beam-search decoding strategies with 4 beams during inference. The hyperparameters utilized for the finetuning process are outlined in Appendix A, while Appendix B and C detail the dataset preprocess and postprocess methods employed for fine-tuning. Additionally, every fine-tuning process was done using a 4× Quadro RTX 8000 (48GB VRAM) card.

5 Results

We tried both fine-tuning and in-context learning of LLM for this task. We established a baseline for our experiment using the gpt-3.5-turbo model, which received a RS(10) score of -561.56 points. To enable in-context learning for the model, we used few-shot prompting. In the second experiment, we fine-tuned the codellama model, and despite having only 7B parameters, it outperformed the gpt-3.5-turbo model. Notably, both the question classifier and SQL generation in both 1 & 2 experiments used the same model. Experiment 3 showcased the robustness of our SQL generation module, as it achieved an impressive RS(0) score of 74.97. This indicates that our system can correctly generate executable queries 74.97% of the time. Experiment 4 focused on improving the question classification model by adjusting the threshold to identify unanswerable questions better. While this enhanced the RS(10) score by 23.48 points, it caused a decrease in the RS(0) score due to misclassification. Experiment 5 evaluated the performance of the codellama-based question classifier, which showed no significant improvement over classical models. Finally, in experiment 6, we used the CAS module that combines the result of two classification models to enhance the gpt-3.5-turbo model's performance. The input the CAS module is detailed in section 2. After completing the development phase, we submitted our top-performing model for the testing phase, scoring RS(0) of 53.21 and RS(10) of 36.07 points. The SQL generator module achieved an impressive RS(0) score of 77.03 points during the test phase, without any adjustments made to the Question Classifier threshold, which shows the capabilities of the SQL generator module.

After performing an error analysis on the CAS module, it was discovered that false negatives were higher. This indicates that some answerable questions were incorrectly classified as unanswerable. Since our system is designed as a pipeline model, only the questions classified as answerable will advance to the SQL generation model. This resulted in a decrease in the RS(0) score. Notably, most of the false negative predictions were observed in queries related to test procedures, hospitals and departments.

6 Conclusion

We developed a sophisticated system that can generate SQL queries from user queries in the EHR dataset, provided they are convertible to SQL. Our system was able to achieve an impressive rank of 7 on the EHRSQL task. Our experiments have shown that fine-tuning an LLM for task-specific SQL generation significantly enhances its performance compared to in-context learning. However, we acknowledge that our system needs improvement in identifying which user queries can be successfully converted to SQL. This is crucial for ensuring the reliability of our SQL converter system.

To facilitate the reproducibility of our work, we have made available instruction templates, code, and pre-trained models as open-source resources.

Limitations

Our research specifically focused on Codellama models, and we discovered that models fine-tuned on text-to-SQL tasks, such as SQLCoder[3], did not perform well in the EHRSQL task. Additionally, there is a limited amount of data available to train for unanswerable questions within the provided training data, with only 450 out of 5124 questions being unanswerable. In future work, generating synthetic data for unanswerable questions using models like GPT-4 could potentially improve performance. It is important to note that all experiments were conducted using Codellama 7B models.

References

Jayetri Bardhan, Anthony Colas, Kirk Roberts, and Daisy Zhe Wang. 2022. Drugehrqa: A question answering dataset on structured and unstructured electronic health records for medicine related queries. *arXiv preprint arXiv:2205.01290*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Ursin Brunner and Kurt Stockinger. 2021. Valuenet: A natural language-to-sql system that learns from database information. *Preprint*, arXiv:2006.00888.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *Preprint*, arXiv:1601.01280.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. Catboost: gradient boosting with categorical features support. *Preprint*, arXiv:1810.11363.

[3] https://huggingface.co/defog/sqlcoder-7b-2

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *Preprint*, arXiv:2308.15363.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. 2023. Mimic-iv clinical database demo.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32(4):905–936.

Jack Kiefer and Jacob Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. 2022. Ehrsql: A practical text-to-sql benchmark for electronic health records. *Advances in Neural Information Processing Systems*, 35:15589–15601.

Gyubok Lee, Sunjun Kweon, Seongsu Bae, and Edward Choi. 2024. Overview of the ehrsql 2024 shared task on reliable text-to-sql modeling on electronic health records. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, Mexico City, Mexico. Association for Computational Linguistics.

David D Lewis. 1998. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Preprint*, arXiv:2107.13586.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):1–13.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Preprint*, arXiv:2304.11015.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Yu Cheng, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *Preprint*, arXiv:2205.06983.

Herbert E. Robbins. 1951. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. *Preprint*, arXiv:2305.08377.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Preprint*, arXiv:1409.3215.

Chang-You Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain-of-thought style prompting for text-to-sql. *Preprint*, arXiv:2305.14215.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

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*, pages 350–361.

Zhiqiang Wang, Yiran Pang, and Yanbin Lin. 2023. Large language models are zero-shot text classifiers. *Preprint*, arXiv:2312.01044.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Xiaodan Zhang, Nabasmita Talukdar, Sandeep Vemulapalli, Sumyeong Ahn, Jiankun Wang, Han Meng, Sardar Mehtab Bin Murtaza, Dmitry Leshchiner, Aakash Ajay Dave, Dimitri F. Joseph, Martin Witteveen-Lane, Dave Chesla, Jiayu Zhou, and Bin Chen. 2024. Comparison of prompt engineering and fine-tuning strategies in large language models in the classification of clinical notes. *medRxiv*.

A Hyperparameter

Hyperparameter	Codellama SA	CL2
Learning rate	2.5e-5	1e-4
Batch size	4	4
Epochs	3	3
Weight decay	0.01	0.01
Weight for loss	-	0.54, 5.69
Lora rank (r)	16	16
Lora rank alpha (α)	32	32
Lora rank dropout	0.05	0.1

Table 2: Hyperparameter used for the best performing model.

Hyperparameters used by the best performing pre-trained language model are listed in Table 2, and the total hyperparameter search space is listed in Table 3. Also, the hyperparameter for the classification model is listed in the Table 4.

Hyperparameter	Value
Learning rate	2.5e-5, 5e-5, 1e-4, 2e-4
Batch size	4, 8, 16, 32
Epochs	1-5
Weight decay	0.01, 0.02, 0.05, 0.1
Lora rank (r)	8, 16, 32, 64
Lora rank alpha (α)	16, 24, 32
Lora rank dropout	0.05, 0.08, 0.1

Table 3: Full list of hyperparameter search space for finetuning LLM

B Dataset Preprocess

For classification, we derive a binary dataset from the raw EHRSQL dataset, which contains only two classes based on the question's answerability. The class label is one if the question is answerable; otherwise, it is zero. We use the raw question and its answerability for the first classification model to create a dataset of questions and their respective class labels. For the second classification model, we map the input to the format of figure 3, which involves providing the SQL schema with foreign keys and the question itself. The class label determination remains the same as in the first classification model. Figure 2 showcases the prompt for the selector in the CAS module, which includes $< dti, dsi, qi, fi, cii, cij >$. We select a few-shot example based on the cosine similarity between the

Model	HyperParameters	values
MultinomialNB Classifier	$alpha$	0, 0.01, **0.02**, 1.0
	fit_prior	**True**,False
SGD classifier	$loss$	**modified_huber**, log_loss,huber
	max_iter	1000, 5000, **8000**
	tol	**1e-4**, 1e-5, 2e-5
	$penalty$	**l2**, l1
CatBoost classifier	$learning\_rate$	0.01, **0.0056**, 0.01, 0.2
	$depth$	4,5,**6**,8
	$l2\_leaf\_reg$	1, 4, **6.5**, 8.5, 10
	$subsample$	0.1, **0.3**, 0.5, 1
	$loss$	LogLoss, **CrossEntropy**

Table 4: Hyperparameter space for the classification experiments. Hyperparameters in bold are what we used for the our classification models

given question (qi) and the entire set of training questions using a pre-trained sentence transformer called all-mpnet-base-v2 [4]. From this process, we identify the four most similar training questions along with their corresponding SQL query, which will serve as our few-shot examples.For SQL generation, we formatted the raw data into the format of figure 2. The prompt includes $< qi, dti, dsi, qsi >$, where qi is the question, dti is the database table information, dsi is the schema with foreign key details, and qsi is the SQL query for the corresponding question.

C Dataset Postprocess

In order to guarantee that the classification models' outputs are effectively conveyed to the selector within the CAS module, a postprocessing step must be incorporated. This step entails modifying the classification output: if the output is 1, it is transformed to "Able to generate answer"; otherwise, it is converted to "Unable to generate answer". Furthermore, the generated output in the SQL generation module is trimmed to include only the section between the [SQL] and [SQL] keywords.

[4] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[INST] ### Task
Generate a SQL query to answer [QUESTION]question[/QUESTION].

Database Table Description
The table name and its corresponding description are as follows:
{table description}

Database Schema
This query will run on a database whose schema is represented in this string:
{schema}

Answer
Given the database schema, here is the SQL query that answers [QUESTION]question[/QUESTION]
[SQL]{sql}[/SQL]

Figure 3: Prompt for SQL generation.

D Prompt and Examples for SQL generation module

Prompt in the figure 3 is used to train and inference pre-trained large language model for the SQL generation task. Given below is a full-fledged example for SQL generation prompt.

Example 1

Based on the database schema and table description, determine which AI assistant's answer accurately identifies whether the given question can generate an SQL query or not.
Database Table Description
The table name and its corresponding description are as follows:
ADMISSIONS – Every unique hospitalization for each patient in the database
PATIENTS – Every unique patient in the database
D_ICD_DIAGNOSES – International Statistical Classification of Diseases and Related Health Problems (ICD-9) codes relating to diagnoses
D_ICD_PROCEDURES – International Statistical Classification of Diseases and Related Health Problems (ICD-9) codes relating to procedures
D_LABITEMS – Local codes ('ITEMIDs') appearing in the database that relate to laboratory tests
D_ITEMS – Local codes ('ITEMIDs') appearing in the database, except those that relate to laboratory tests
DIAGNOSES_ICD – Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
PROCEDURES_ICD – Patient procedures, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
LABEVENTS – Laboratory measurements for patients both within the hospital and in outpatient clinics
PRESCRIPTIONS – Medications ordered for a given patient
COST – All patients events cost
CHARTEVENTS – All charted observations for patients
INPUTEVENTS – Intake for patients monitored while in the ICU
OUTPUTEVENTS – Output information for patients while in the ICU
MICROBIOLOGYEVENTS – Microbiology culture results and antibiotic sensitivities from the hospital database
ICUSTAYS – Every unique ICU stay in the database
TRANSFERS – Patient movement from bed to bed within the hospital

Database Schema
This query will run on a database whose schema is represented in this string:
CREATE TABLE patients
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of the patient
subject_id INT NOT NULL UNIQUE, – Unique subject id of the patient
gender VARCHAR(5) NOT NULL, – Gender of the patient
dob TIMESTAMP(0) NOT NULL, – Date of birth of the patient
dod TIMESTAMP(0) – Date of death of the patient
);
CREATE TABLE admissions
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of the admission
subject_id INT NOT NULL, – Subject id of the admission
hadm_id INT NOT NULL UNIQUE, – Unique hospital admission id of the admission
admittime TIMESTAMP(0) NOT NULL, – Admit time of the admission
dischtime TIMESTAMP(0), – Discharge time of the admission
admission_type VARCHAR(50) NOT NULL, – Admission type of the admission
admission_location VARCHAR(50) NOT NULL, – Admission location of the admission
discharge_location VARCHAR(50), – Discharge location of the admission
insurance VARCHAR(255) NOT NULL, – Insurance of the admission
language VARCHAR(10), – Langauge of the admission
marital_status VARCHAR(50), – Marital status of the admission
age INT NOT NULL, – Age of the admission
);

```
CREATE TABLE d_icd_diagnoses
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of the icd diagnose
icd_code VARCHAR(10) NOT NULL
UNIQUE, – Unique icd code of the icd
diagnose
long_title VARCHAR(255) NOT NULL –
Title of the icd diagnose
);
CREATE TABLE d_icd_procedures
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of icd procedure
icd_code VARCHAR(10) NOT NULL
UNIQUE, – Unique icd code of the icd
procedure
long_title VARCHAR(255) NOT NULL –
Title of the icd procedure
);
CREATE TABLE d_labitems
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of the item relate to laboratory
tests
itemid INT NOT NULL UNIQUE, –
Unique item id of the item relate to
laboratory tests
label VARCHAR(200) – Label of the item
relate to laboratory tests
);
CREATE TABLE d_items
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of the item excepts item relate
to laboratory tests
itemid INT NOT NULL UNIQUE, –
Unique item id of the item excepts item
relate to laboratory tests
label VARCHAR(200) NOT NULL, –
Label of item excepts item relate to
laboratory tests
abbreviation VARCHAR(200) NOT NULL,
– Abbreviation of item excepts item relate
to laboratory tests
linksto VARCHAR(50) NOT NULL –
Event linked to item excepts item relate to
laboratory tests
);
CREATE TABLE diagnoses_icd
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of diagnose
subject_id INT NOT NULL, – Subject id
of diagnose
hadm_id INT NOT NULL, – Hospital
admission id of diagnose
icd_code VARCHAR(10) NOT NULL, –
ICD code of diagnose
charttime TIMESTAMP(0) NOT NULL, –
Chart time of diagnose
);
CREATE TABLE procedures_icd
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of procedures
subject_id INT NOT NULL, – Subject id
of procedures
hadm_id INT NOT NULL, – Hospital
admission id of procedures
icd_code VARCHAR(10) NOT NULL, –
ICD code of procedures
charttime TIMESTAMP(0) NOT NULL, –
Chart time of procedures
);
CREATE TABLE labevents
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of laboratory event
subject_id INT NOT NULL, – Subject id
of laboratory event
hadm_id INT NOT NULL, – Hospital
admission id of laboratory event
itemid INT NOT NULL, – Item id of
laboratory event
charttime TIMESTAMP(0), – Chart time of
laboratory event
valuenum DOUBLE PRECISION, – Nu-
merical value measured of laboratory event
valueuom VARCHAR(20), – Unit of
numerical value of laboratory event
);
CREATE TABLE prescriptions
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of prescription
subject_id INT NOT NULL, – Subject id
of prescription
hadm_id INT NOT NULL, – Hospital
admission id of prescription
```

```
starttime TIMESTAMP(0) NOT NULL, – Start time of prescription
stoptime TIMESTAMP(0), – Stop time of prescription
drug VARCHAR(255) NOT NULL, – Drug name of prescription
dose_val_rx VARCHAR(100) NOT NULL, – Dosage value of prescription
dose_unit_rx VARCHAR(50) NOT NULL, – Dosage unit of prescription
route VARCHAR(50) NOT NULL, – Intake method of prescription
);
CREATE TABLE cost
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of cost event
subject_id INT NOT NULL, – Subject id of cost event
hadm_id INT NOT NULL, – Hospital admission id of cost event
event_type VARCHAR(20) NOT NULL, – Event type of cost event
event_id INT NOT NULL, – Event id of cost event
chargetime TIMESTAMP(0) NOT NULL, – Charge time of cost event
cost DOUBLE PRECISION NOT NULL, – Cost of cost event
);
CREATE TABLE chartevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of chart event
subject_id INT NOT NULL, – Subject id of chart event
hadm_id INT NOT NULL, – Hospital admission id of chart event
stay_id INT NOT NULL, – Stay ID of chart event
itemid INT NOT NULL, – Item ID of chart event
charttime TIMESTAMP(0) NOT NULL, – Chart time of chart event
valuenum DOUBLE PRECISION, – Numerical value measured of chart event
valueuom VARCHAR(50), – Unit of numerical value of chart event
);
CREATE TABLE inputevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of input event
subject_id INT NOT NULL, – Subject id of input event
hadm_id INT NOT NULL, – Hospital admission id of input event
stay_id INT NOT NULL, – Stay id of input event
starttime TIMESTAMP(0) NOT NULL, – Start time of input event
itemid INT NOT NULL, – Item id of input event
amount DOUBLE PRECISION, – Amount of input event
);
CREATE TABLE outputevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of output event
subject_id INT NOT NULL, – Subject id of output event
hadm_id INT NOT NULL, – Hospital admission id of output event
stay_id INT NOT NULL, – Stay id of output event
charttime TIMESTAMP(0) NOT NULL, – Chart time of output event
itemid INT NOT NULL, – Item id of output event
value DOUBLE PRECISION, – Value of output event
);
CREATE TABLE microbiologyevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of microbiologyevent
subject_id INT NOT NULL, – Subject id of microbiologyevent
hadm_id INT NOT NULL, – Hospital admission id of microbiologyevent
charttime TIMESTAMP(0) NOT NULL, – Chart time of microbiologyevent
spec_type_desc VARCHAR(100), – Specimen name of microbiologyevent
test_name VARCHAR(100), – Test name of microbiologyevent
org_name VARCHAR(100), – Organism name of microbiologyevent
);
```

CREATE TABLE icustays
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of icu stay
subject_id INT NOT NULL, – Subject id of icu stay
hadm_id INT NOT NULL, – Hospital admission id of icu stay
stay_id INT NOT NULL UNIQUE, – Stay id of icu stay
first_careunit VARCHAR(20) NOT NULL, – first care unit of icu stay
last_careunit VARCHAR(20) NOT NULL, – Last care unit of icu stay
intime TIMESTAMP(0) NOT NULL, – In time of icu stay
outtime TIMESTAMP(0), – Out time of icu stay
);
CREATE TABLE transfers
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of transfer
subject_id INT NOT NULL, – Subject Id of transfer
hadm_id INT NOT NULL, – Hospital admission id of transfer
transfer_id INT NOT NULL, – Transfer Id of transfer
eventtype VARCHAR(20) NOT NULL, – Event type of transfer
careunit VARCHAR(20), – Care unit of transfer
intime TIMESTAMP(0) NOT NULL, – In time of transfer
outtime TIMESTAMP(0), – Out time of transfer
);
– admissions.subject_id can be joined with patients.subject_id
– diagnoses_icd.hadm_id can be joined with admissions.hadm_id
– diagnoses_icd.icd_code can be joined with d_icd_diagnoses.icd_code
– procedures_icd.hadm_id can be joined with admissions.hadm_id
– procedures_icd.icd_code can be joined with d_icd_procedures.icd_code
– labevents.hadm_id can be joined with admissions.hadm_id
– labevents.itemid can be joined with d_labitems.itemid
– prescriptions.hadm_id can be joined with admissions.hadm_id
– cost.hadm_id can be joined with admissions.hadm_id
– cost.event_id can be joined with diagnoses_icd.row_id
– cost.event_id can be joined with procedures_icd.row_id
– cost.event_id can be joined with labevents.row_id
– cost.event_id can be joined with prescriptions.row_id
– chartevents.hadm_id can be joined with admissions.hadm_id
– chartevents.stay_id can be joined with icustays.stay_id
– chartevents.itemid can be joined with d_items.itemid
– inputevents.hadm_id can be joined with admissions.hadm_id
– inputevents.stay_id can be joined with icustays.stay_id
– inputevents.itemid can be joined with d_items.itemid
– outputevents.hadm_id can be joined with admissions.hadm_id
– outputevents.stay_id can be joined with icustays.stay_id
– outputevents.itemid can be joined with d_items.itemid
– microbiologyevents.hadm_id can be joined with admissions.hadm_id
– icustays.hadm_id can be joined with admissions.hadm_id
– transfers.hadm_id can be joined with admissions.hadm_id

Answer
Given the database schema, here is the SQL query that answers [QUESTION]What was the drug that patient 10015931 was prescribed with within the same hospital visit after the replacement of aortic valve with zooplastic tissue, percutaneous approach since 5 months ago?[/QUESTION]
[SQL] SELECT admissions.subject_id, prescriptions.drug,prescriptions.starttime, admissions.hadm_id FROM prescrip-

tions JOIN admissions ON prescriptions.hadm_id = admissions.hadm_id WHERE admissions.subject_id = 10015931 [/SQL]

E Prompt and Examples for CAS Module

Following is the full example for the prompt in the figure 2

Example 2

Based on the database schema and table description, determine which AI assistant's answer accurately identifies whether the given question can generate an SQL query or not.
Database Table Description
The table name and its corresponding description are as follows:
ADMISSIONS – Every unique hospitalization for each patient in the database
PATIENTS – Every unique patient in the database
D_ICD_DIAGNOSES – International Statistical Classification of Diseases and Related Health Problems (ICD-9) codes relating to diagnoses
D_ICD_PROCEDURES – International Statistical Classification of Diseases and Related Health Problems (ICD-9) codes relating to procedures
D_LABITEMS – Local codes ('ITEMIDs') appearing in the database that relate to laboratory tests
D_ITEMS – Local codes ('ITEMIDs') appearing in the database, except those that relate to laboratory tests
DIAGNOSES_ICD – Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
PROCEDURES_ICD – Patient procedures, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
LABEVENTS – Laboratory measurements for patients both within the hospital and in outpatient clinics
PRESCRIPTIONS – Medications ordered for a given patient
COST – All patients events cost
CHARTEVENTS – All charted observations for patients
INPUTEVENTS – Intake for patients monitored while in the ICU
OUTPUTEVENTS – Output information for patients while in the ICU
MICROBIOLOGYEVENTS – Microbiology culture results and antibiotic sensitivities from the hospital database
ICUSTAYS – Every unique ICU stay in the database
TRANSFERS – Patient movement from bed to bed within the hospital

Database Schema
This query will run on a database whose schema is represented in this string:
CREATE TABLE patients
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of the patient
subject_id INT NOT NULL UNIQUE, – Unique subject id of the patient
gender VARCHAR(5) NOT NULL, – Gender of the patient
dob TIMESTAMP(0) NOT NULL, – Date of birth of the patient
dod TIMESTAMP(0) – Date of death of the patient
);
CREATE TABLE admissions
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of the admission
subject_id INT NOT NULL, – Subject id of the admission
hadm_id INT NOT NULL UNIQUE, – Unique hospital admission id of the admission
admittime TIMESTAMP(0) NOT NULL, – Admit time of the admission
dischtime TIMESTAMP(0), – Discharge time of the admission
admission_type VARCHAR(50) NOT NULL, – Admission type of the admission
admission_location VARCHAR(50) NOT NULL, – Admission location of the admission
discharge_location VARCHAR(50), –

```
Discharge location of the admission
insurance VARCHAR(255) NOT NULL, –
Insurance of the admission
language VARCHAR(10), – Langauge of
the admission
marital_status VARCHAR(50), – Marital
status of the admission
age INT NOT NULL, – Age of the
admission
);
CREATE TABLE d_icd_diagnoses
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of the icd diagnose
icd_code VARCHAR(10) NOT NULL
UNIQUE, – Unique icd code of the icd
diagnose
long_title VARCHAR(255) NOT NULL –
Title of the icd diagnose
);
CREATE TABLE d_icd_procedures
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of icd procedure
icd_code VARCHAR(10) NOT NULL
UNIQUE, – Unique icd code of the icd
procedure
long_title VARCHAR(255) NOT NULL –
Title of the icd procedure
);
CREATE TABLE d_labitems
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of the item relate to laboratory
tests
itemid INT NOT NULL UNIQUE, –
Unique item id of the item relate to
laboratory tests
label VARCHAR(200) – Label of the item
relate to laboratory tests
);
CREATE TABLE d_items
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of the item excepts item relate
to laboratory tests
itemid INT NOT NULL UNIQUE, –
Unique item id of the item excepts item
relate to laboratory tests
label VARCHAR(200) NOT NULL, –
Label of item excepts item relate to
laboratory tests
abbreviation VARCHAR(200) NOT NULL,
– Abbreviation of item excepts item relate
to laboratory tests
linksto VARCHAR(50) NOT NULL –
Event linked to item excepts item relate to
laboratory tests
);
CREATE TABLE diagnoses_icd
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of diagnose
subject_id INT NOT NULL, – Subject id
of diagnose
hadm_id INT NOT NULL, – Hospital
admission id of diagnose
icd_code VARCHAR(10) NOT NULL, –
ICD code of diagnose
charttime TIMESTAMP(0) NOT NULL, –
Chart time of diagnose
);
CREATE TABLE procedures_icd
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of procedures
subject_id INT NOT NULL, – Subject id
of procedures
hadm_id INT NOT NULL, – Hospital
admission id of procedures
icd_code VARCHAR(10) NOT NULL, –
ICD code of procedures
charttime TIMESTAMP(0) NOT NULL, –
Chart time of procedures
);
CREATE TABLE labevents
(
row_id INT NOT NULL PRIMARY KEY, –
Unique ID of laboratory event
subject_id INT NOT NULL, – Subject id
of laboratory event
hadm_id INT NOT NULL, – Hospital
admission id of laboratory event
itemid INT NOT NULL, – Item id of
laboratory event
charttime TIMESTAMP(0), – Chart time of
laboratory event
valuenum DOUBLE PRECISION, – Nu-
merical value measured of laboratory event
valueuom VARCHAR(20), – Unit of
```

```
numerical value of laboratory event
);
CREATE TABLE prescriptions
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of prescription
subject_id INT NOT NULL, – Subject id of prescription
hadm_id INT NOT NULL, – Hospital admission id of prescription
starttime TIMESTAMP(0) NOT NULL, – Start time of prescription
stoptime TIMESTAMP(0), – Stop time of prescription
drug VARCHAR(255) NOT NULL, – Drug name of prescription
dose_val_rx VARCHAR(100) NOT NULL, – Dosage value of prescription
dose_unit_rx VARCHAR(50) NOT NULL, – Dosage unit of prescription
route VARCHAR(50) NOT NULL, – Intake method of prescription
);
CREATE TABLE cost
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of cost event
subject_id INT NOT NULL, – Subject id of cost event
hadm_id INT NOT NULL, – Hospital admission id of cost event
event_type VARCHAR(20) NOT NULL, – Event type of cost event
event_id INT NOT NULL, – Event id of cost event
chargetime TIMESTAMP(0) NOT NULL, – Charge time of cost event
cost DOUBLE PRECISION NOT NULL, – Cost of cost event
);
CREATE TABLE chartevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of chart event
subject_id INT NOT NULL, – Subject id of chart event
hadm_id INT NOT NULL, – Hospital admission id of chart event
stay_id INT NOT NULL, – Stay ID of chart event
itemid INT NOT NULL, – Item ID of chart event
charttime TIMESTAMP(0) NOT NULL, – Chart time of chart event
valuenum DOUBLE PRECISION, – Numerical value measured of chart event
valueuom VARCHAR(50), – Unit of numerical value of chart event
);
CREATE TABLE inputevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of input event
subject_id INT NOT NULL, – Subject id of input event
hadm_id INT NOT NULL, – Hospital admission id of input event
stay_id INT NOT NULL, – Stay id of input event
starttime TIMESTAMP(0) NOT NULL, – Start time of input event
itemid INT NOT NULL, – Item id of input event
amount DOUBLE PRECISION, – Amount of input event
);
CREATE TABLE outputevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of output event
subject_id INT NOT NULL, – Subject id of output event
hadm_id INT NOT NULL, – Hospital admission id of output event
stay_id INT NOT NULL, – Stay id of output event
charttime TIMESTAMP(0) NOT NULL, – Chart time of output event
itemid INT NOT NULL, – Item id of output event
value DOUBLE PRECISION, – Value of output event
);
CREATE TABLE microbiologyevents
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of microbiologyevent
subject_id INT NOT NULL, – Subject id of microbiologyevent
hadm_id INT NOT NULL, – Hospital
```

admission id of microbiologyevent
charttime TIMESTAMP(0) NOT NULL, – Chart time of microbiologyevent
spec_type_desc VARCHAR(100), – Specimen name of microbiologyevent
test_name VARCHAR(100), – Test name of microbiologyevent
org_name VARCHAR(100), – Organism name of microbiologyevent
);
CREATE TABLE icustays
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of icu stay
subject_id INT NOT NULL, – Subject id of icu stay
hadm_id INT NOT NULL, – Hospital admission id of icu stay
stay_id INT NOT NULL UNIQUE, – Stay id of icu stay
first_careunit VARCHAR(20) NOT NULL, – first care unit of icu stay
last_careunit VARCHAR(20) NOT NULL, – Last care unit of icu stay
intime TIMESTAMP(0) NOT NULL, – In time of icu stay
outtime TIMESTAMP(0), – Out time of icu stay
);
CREATE TABLE transfers
(
row_id INT NOT NULL PRIMARY KEY, – Unique ID of transfer
subject_id INT NOT NULL, – Subject Id of transfer
hadm_id INT NOT NULL, – Hospital admission id of transfer
transfer_id INT NOT NULL, – Transfer Id of transfer
eventtype VARCHAR(20) NOT NULL, – Event type of transfer
careunit VARCHAR(20), – Care unit of transfer
intime TIMESTAMP(0) NOT NULL, – In time of transfer
outtime TIMESTAMP(0), – Out time of transfer
);
– admissions.subject_id can be joined with patients.subject_id
– diagnoses_icd.hadm_id can be joined with admissions.hadm_id
– diagnoses_icd.icd_code can be joined with d_icd_diagnoses.icd_code
– procedures_icd.hadm_id can be joined with admissions.hadm_id
– procedures_icd.icd_code can be joined with d_icd_procedures.icd_code
– labevents.hadm_id can be joined with admissions.hadm_id
– labevents.itemid can be joined with d_labitems.itemid
– prescriptions.hadm_id can be joined with admissions.hadm_id
– cost.hadm_id can be joined with admissions.hadm_id
– cost.event_id can be joined with diagnoses_icd.row_id
– cost.event_id can be joined with procedures_icd.row_id
– cost.event_id can be joined with labevents.row_id
– cost.event_id can be joined with prescriptions.row_id
– chartevents.hadm_id can be joined with admissions.hadm_id
– chartevents.stay_id can be joined with icustays.stay_id
– chartevents.itemid can be joined with d_items.itemid
– inputevents.hadm_id can be joined with admissions.hadm_id
– inputevents.stay_id can be joined with icustays.stay_id
– inputevents.itemid can be joined with d_items.itemid
– outputevents.hadm_id can be joined with admissions.hadm_id
– outputevents.stay_id can be joined with icustays.stay_id
– outputevents.itemid can be joined with d_items.itemid
– microbiologyevents.hadm_id can be joined with admissions.hadm_id
– icustays.hadm_id can be joined with admissions.hadm_id
– transfers.hadm_id can be joined with admissions.hadm_id

Question: 'What was the drug that patient 10015931 was prescribed with within the same hospital visit after the replacement of aortic valve with zooplastic tissue, percutaneous approach since 5 months ago?'
Answer: Able to generate SQL Query.

Question: 'Tell me the name of the prescription drug that patient 10015931 was prescribed in the same day after having a replacement of aortic valve with zooplastic tissue, percutaneous approach procedure since 4 months ago?'
Answer: Able to generate SQL Query.

Question: 'What was prescribed to patient 10015931 during the same hospital visit following their replacement of aortic valve with zooplastic tissue, percutaneous approach during this month?'
Answer: Able to generate SQL Query.

Question: 'What was the drug that patient 10025463 was prescribed for during the same hospital encounter after the procedure of excision or destruction of other lesion or tissue of heart, endovascular approach?'
Answer: Able to generate SQL Query.

Question: "What was the drug that patient 10015931 was prescribed with within the same hospital visit after the replacement of aortic valve with zooplastic tissue, percutaneous approach since 5 months ago?"
Ai Assitant 1's Answer: Able to generate SQL Query.
Ai Assitant 2's Answer: Able to generate SQL Query.
Answer: Let's think step by step."