ZOGRASCOPE: A New Benchmark for Semantic Parsing over Property Graphs

Francesco Cazzaro[†], **Justin Kleindienst**[§], **Sofia Márquez Gomez**[§], **Ariadna Quattoni**[†§]

[†]Universitat Politècnica de Catalunya, Barcelona, Spain

[§]dMetrics, Jackson Wyoming USA

`francesco.cazzaro@upc.edu, aquattoni@cs.upc.edu`

`{justin.kleindienst, sofia.marquez, ariadna.quattoni}@dmetrics.com`

Abstract

In recent years, the need for natural language interfaces to knowledge graphs has become increasingly important since they enable easy and efficient access to the information contained in them. In particular, property graphs (PGs) have seen increased adoption as a means of representing complex structured information. Despite their growing popularity in industry, PGs remain relatively underrepresented in semantic parsing research with a lack of resources for evaluation. To address this gap, we introduce ZOGRASCOPE, a benchmark designed specifically for PGs and queries written in Cypher. Our benchmark includes a diverse set of manually annotated queries of varying complexity and is organized into three partitions: iid, compositional and length. We complement this paper with a set of experiments that test the performance of different LLMs in a variety of learning settings.

1 Introduction

Knowledge graphs (KGs) have become increasingly prominent for representing complex relations between entities across domains and several graph data models have been proposed. Accessing the information contained in a PG typically requires significant expertise in query languages such as Cypher (Francis et al., 2018) or Gremlin (Rodriguez, 2015). To reduce this barrier, natural language interfaces to KGs have emerged as a critical tool. This has driven research in semantic parsing and knowledge-based question answering, which aims to translate natural language questions into executable graph queries (Gu and Su, 2022; Li et al., 2023b; Agarwal et al., 2024; Liang et al., 2024; Li et al., 2024). Despite advances, the task remains challenging due to the inherent complexity of natural language and the variability of graph schemas (Groschwitz et al., 2023; Li et al., 2023a).

Annotated data is essential for advancing development in the area, but acquiring it is both expensive and labor-intensive. Besides, most available benchmarks have focused primarily on RDF-style graphs. Unlike RDF, property graph nodes can have an arbitrary list of associated properties enabling the specification of rich and semantically intuitive data models. PGs clearly distinguish between entity properties and entity relations, whereas RDF graphs conflate the two concepts. Resources for PGs remain severely limited; many available Cypher benchmarks are either synthetically generated without supervision or adapted from non-property graph sources.

To address these challenges, we introduce ZOGRASCOPE, a benchmark designed for property graphs built from an open-access crime investigation graph using Cypher queries. The dataset consists of annotated pairs of natural language questions and Cypher queries. Unlike most prior work, each example is manually annotated by experts. LLMs were used only for intermediate paraphrasing to enhance lexical variety, with all outputs manually verified for correctness. ZOGRASCOPE comprises 5k samples, featuring a range of query types, and includes iid, compositional and length train/test partitions. In addition, the paper includes a wide range of experiments to asses LLMs performance in a variety of learning settings. We hope that ZOGRASCOPE will serve as a valuable resource for advancing research in semantic parsing over property graphs.

We release ZOGRASCOPE at `https://github.com/interact-erc/ZOGRASCOPE`.

2 Related Work

Few benchmarks exist for evaluating semantic parsers on property graphs, most of which are adapted from RDF and tabular datasets, or synthetically generated by LLMs. Using the first approach Zhao et al. (2023a) and Tiwari et al. (2025) adapt SQL benchmarks. Feng et al. (2025) and Nie

Query
| ```
MATCH (x0:Person)-[:KNOWS]-(x1:Person)-[:FAMILY_REL]-(x2:Person WHERE x2.
    surname = "[x2.surname.VALUE]")
RETURN x0.nhs_no
``` |

Proto-NL
List all national health service number of People who knows People that has a family relation with People with last name that is [x2.surname.VALUE]

Human annotation
List all national health service numbers of people who knows someone that is family related to a person with surname [x2.surname.VALUE]

Values Instantiation
List all national health service numbers of people who knows someone that is family related to a person with surname Nichols

LLM paraphrasing
What are the NHS numbers of individuals who know a family member of someone named Nichols?

Human verification and correction
What are the NHS numbers of individuals who know a family member of someone with last name Nichols?

Table 1: Illustration of the annotation process for one ZOGRASCOPE sample

et al. (2022) create benchmarks by converting RDF-graphs to property graphs: the former converts a subset of WikiData (Vrandečić and Krötzsch, 2014) while the latter converts MetaQA (Zhang et al., 2018). Following the second approach Tiwari et al. (2025) and Zhong et al. (2025) use LLMs to build pipelines capable of synthetically producing Cypher benchmarks without human supervision. Ozsoy et al. (2025) gathers synthetically generated samples from various available online resources to construct a benchmark, but the underlying graph to execute queries is absent.

The main difference of our approach with respect to previous works is that we create a manually annotated Cypher benchmark built directly over a native property graph. In contrast to adapted benchmarks, ours exhibits all the characteristics that are unique to the property graph data model.

Zhao et al. (2023b) employ LLMs to build a benchmark directly on top of a PG adding a final human verification step. The main difference with our benchmark creation process is that in our case all the initial annotations are provided by human annotators. Last, the dataset of Guo et al. (2022) resembles our methodology with the difference that it is in Chinese while ours is in English.

3 Dataset Creation

The ZOGRASCOPE benchmark is based on

the publicly available Pole graph[1], which contains a total of $61,521$ nodes and $105,840$ edges. It includes 11 entity classes, 32 unique properties, and 17 relation types. The graph focuses on crime investigations, modeling the relationships between people, objects, locations and events. For instance, a query might ask for the name of an officer investigating a particular crime that occurred at a specific location and involved a particular vehicle. Creating the benchmark involves a five step process: 1) We automatically generate variable anonymized Cypher queries. 2) Each anonymized Cypher query is manually annotated by an expert with a natural language realization. 3) For each query we instantiate the values of the anonymized variables in both the Cypher query and its natural language realization. 4) An LLM is used to suggest paraphrases of the instantiated natural language realizations and 5) A human annotator verifies and corrects the instantiated paraphrases.

Table 1 illustrates the annotation process. We use four human annotators, all proficient in the Cypher language. At any point, the annotator is free to discard a sample. This quality control mechanism ensures that only meaningful, well-formed queries and their corresponding natural language realizations are included in the final dataset.

To generate Cypher queries, we employ an au-

---
[1] https://github.com/neo4j-graph-examples/pole

	NL	*What is the number of shoplifting investigations conducted by Police Constables?*

	Query	`MATCH (x0:Crime WHERE x0.type = "Shoplifting")-[:INVESTIGATED_BY]-(`
		`    x1:Officer WHERE x1.rank = "Police Constable")`
		`RETURN COUNT(DISTINCT x0)`

	Extra Info	x0.Crime.type:Shoplifting = shoplifting
		x1.Officer.rank:Police Constable = Police Constables
	NL	*How long was the last call to the phone of a person linked to a crime on 11/08/2017?*

	Query	`MATCH (x0:PhoneCall)-[:CALLER]-(x1:Phone)-[:HAS_PHONE]-(x2:Person)`
		`    -[:PARTY_TO]-(x3:Crime WHERE x3.date = "11/08/2017")`
		`RETURN x0.call_duration`
		`ORDER BY x0.call_date DESC`
		`LIMIT 1`

	Extra Info	x3.Crime.date:11/08/2017 = 11/08/2017
	NL	*Can you list those connected to a crime handled by Officer Rockall, with the investigation finished and no suspect found?*

	Query	`MATCH (x0:Person)-[:PARTY_TO]-(x1:Crime WHERE x1.last_outcome = "`
		`    Investigation complete; no suspect identified")-[:`
		`    INVESTIGATED_BY]-(x2:Officer WHERE x2.surname = "Rockall")`
		`RETURN x0`

	Extra Info	x1.Crime.last_outcome:Investigation complete; no suspect identified = investigation finished and no suspect found
		x2.Officer.surname:Rockall = Rockall

Table 2: Examples of ZOGRASCOPE data samples.

	Tot.	%2n.	%3n.	%4n.	%5n.
TRAIN	2905	51.9	28.9	15.2	3.9
IID	767	50.1	29.3	16.4	4.2
COMP	1350	12.7	47.3	34.6	5.4
LEN_{trn}	3769	54.8	45.2	0	0
LEN_{test}	1253	0	0	82.5	17.5

Table 3: Total number of samples in each partition and their distribution among different query lengths.

tomatic procedure. We start by constructing a tree pattern, which is essentially a small subgraph, where the root node represents the answer, and the other nodes are connected through edges that represent relations. These tree patterns are designed to be consistent with the underlying graph, ensuring that nodes are connected by valid relations. Nodes in the tree can be unconstrained or constrained by property-value pairs. For example, an unconstrained node might correspond to a *Person* and have no property constraints. Each tree pattern can be deterministically mapped to a corresponding Cypher-query and that is how we generate the anonymized Cypher queries.

For each query, we also associate a determinis-

tically generated proto-natural language sentence realization, this is only used to aid the human annotator who can decide to ignore it if he prefers to only look at the Cypher query. The annotator is then asked to generate a realistic natural language realization of the query. Annotators are provided with the graph schema, which includes information on entity classes, properties, and relations. The result of this step is an anonymized Cypher query and its corresponding anonymized natural language realization. We then instantiate the anonymized variables with valid values from the graph, making sure that the query yields non-empty results. We generate three instantiations per query. Subsequently, to enrich the lexical diversity of the natural language queries we use an LLM (GPT4o) to produce three paraphrases for each instantiation. In the final step a human annotator reviews each paraphrase verifying and correcting them if needed.

The final dataset generated using this approach has around 5k samples and the query lengths range from 2 to 5 nodes. We show some examples in Table 2. The queries in the benchmark span a variety of aggregation types, including set operations, set size, attribute value sets, maximum and minimum attribute value sets, and both argmax and

	Model	iid	compositional	length
Fine-tuning	Mistral 7B	97.87	74.97	23.46
	Qwen3 4B	98.04	72.42	20.19
	Llama 3.2-3B	97.90	70.99	25.88
Dynamic ICL	GPT-4o	96.74	66.79	50.19
	Mistral 7B	95.18	39.95	13.57
	Qwen3 4B	94.79	42.10	18.27
	Llama 3.2-3B	97.00	31.73	10.93
Fixed ICL	GPT-4o	66.40	63.71	22.66
	Mistral 7B	44.44	25.30	11.33
	Qwen3 4B	43.27	28.76	12.69
	Llama 3.2-3B	31.03	15.49	5.98
Zero-shot	GPT-4o	41.67	32.91	16.28
	Mistral 7B	8.85	4.82	0.56
	Qwen3 4B	10.41	7.04	0.88
	Llama 3.2-3B	3.38	1.48	0.24

Table 4: Experimental results on each ZOGRASCOPE partition.

argmin attribute values. We split the data into training (~3k) and test (~2k) sets, organizing the test into an iid and a compositional partition. We define a template as a Cypher query with anonymized properties, classes, relations, and values, retaining only structural information. Test samples that correspond to templates not observed in the training set form the compositional split, while test samples corresponding to templates observed in the training set comprise the iid split. We also provide a length-based split: the training set includes queries with 2–3 nodes, while the test set includes those with 4–5 nodes. In Table 3 we present some statistics.

4 Experiments

We ran a series of experiments to examine semantic parsing performance on ZOGRASCOPE. Specifically, we assess different transformer-based models under four distinct settings:

• **FINE-TUNING**: The model is fine-tuned on the benchmark's training set. Details about the hyperparameters are provided in Appendix A.

• **DYNAMIC ICL** In-context learning with 5 examples dynamically retrieved from the training set based on the test question. We follow Agarwal et al. (2024) in using a sentence transformer as encoder and retrieving based on cosine similarity.

• **FIXED ICL** In-context learning with 5 fixed examples randomly sampled from the training set.

• **ZERO-SHOT** An LLM is prompted to generate the Cypher query without any example.

Note that for the last three settings we include in the LLM prompt the schema description of the graph with information about classes, properties and relations (see Appendix B).

We present the results of these evaluations using a range of LLMs including small models that can run locally: *Llama-3.2-3B* (Grattafiori et al., 2024), *Qwen3-4B* (Yang et al., 2025), and *Mistral-7B-v0.1* (Jiang et al., 2023) as well as a larger proprietary model GPT-4o: (*gpt-4o-2024-08-06*).

For the evaluation metric, we use execution accuracy, where a prediction is deemed correct if it yields identical results to the reference query when executed on the knowledge graph. Since we are interested in the semantic parsing task, we run these experiments assuming perfect entity linking by passing the gold entities to the models.

Table 4 presents the results of our experiments. As expected, we observe that the iid partition is the easiest, while performance decreases on the compositional partition. The length partition is the most challenging, with models struggling to generalize to longer queries. Regarding the different learning settings we observe that performance improves as more supervision becomes available. In the zero-shot setting, with no supervision, performance is low, but just by including a few examples accuracy improves substantially. Dynamically selecting examples further improves the accuracy, with the

Figure 1: Learning curve for the compositional partition.

	iid	compositional	length
Fine-tuning	94.14	68.49	21.38
Dynamic ICL	95.05	27.58	10.85
Fixed ICL	21.93	10.08	5.15
Zero-shot	4.16	1.44	0.15

Table 5: Experimental results for *Llama-3.2-3B* without gold entity linking.

iid partition almost achieving perfect performance. However, the performance of GPT4o on the compositional partition does not exhibit the same type of improvement. Finally, fine-tuning with supervised data remains the most effective approach achieving the best performance for each of the models.

The performance of the different models is usually correlated with their size, with larger models faring better. GPT-4o, being substantially larger, outperforms all other models demonstrating better capabilities. However, we observe that fine-tuning smaller models can still achieve better performance than using GPT4o with in-context learning alone. In the length partition this does not occur, but it is quite difficult for a fine-tuned model to generalize to longer query structures.

Figure 1 shows learning curves for the fine-tuned models over the compositional partition. Models are trained on randomly sampled subsets of the training set with sizes of 100, 200, 400, 800 and 1500 examples. We observe a typical learning curve with a sharp rise and then a flattening.

Finally in Table 5 we report results for the *Llama-3.2-3B* model without passing the gold entity linking values as input.

5 Conclusions

In this paper, we introduced ZOGRASCOPE, a novel benchmark designed to evaluate the performance of semantic parsers over Cypher property graphs. Built on the Pole graph, which models crime investigations, the benchmark includes a set of Cypher queries paired with natural language realizations, manually annotated by expert annotators. We hope that ZOGRASCOPE will contribute to advancing research in automatic text-to-Cypher translation, addressing the limited availability of similar resources and fostering progress in the field.

Limitations

One limitation of our work concerns the complexity of the query patterns. While we address a range of realistic query structures, certain types such as nested sub-queries are not currently included. In future work, we plan to extend the dataset to cover these more complex cases. A second limitation regards the entity values in the dataset. Because we instantiate entity values in a second step, even if their wording gets modified, their mapping might still be relatively straightforward. Thirdly, we acknowledge that the dataset only tackles the English language while it would be useful for these kind of resources to be available for multiple languages.

Risks and Ethical Considerations

Our dataset is constructed from a graph that has been rendered publicly available and is based on open data provided by the UK government. Any sensitive information, such as people names, phone numbers, and emails was removed and replaced with fictitious content within the graph prior to its release. As such we believe that our derived dataset does not pose privacy risks nor contain potentially harmful content.

Acknowledgments

References

Dhruv Agarwal, Rajarshi Das, Sopan Khosla, and Rashmi Gangadharaiah. 2024. Bring your own KG: Self-supervised program synthesis for zero-shot KGQA. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 896–919, Mexico City, Mexico. Association for Computational Linguistics.

Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. 2025. CypherBench: Towards precise retrieval over full-scale modern knowledge graphs in the LLM era. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8934–8958, Vienna, Austria. Association for Computational Linguistics.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1433–1445, New York, NY, USA. Association for Computing Machinery.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jonas Groschwitz, Shay Cohen, Lucia Donatelli, and Meaghan Fowlie. 2023. AMR parsing is far from solved: GrAPES, the granular AMR parsing evaluation suite. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10728–10752, Singapore. Association for Computational Linguistics.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Aibo Guo, Xinyi Li, Guanchen Xiao, Zhen Tan, and Xiang Zhao. 2022. Spcql: A semantic parsing dataset for converting natural language into cypher. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 3973–3977. ACM.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Bingzhi Li, Lucia Donatelli, Alexander Koller, Tal Linzen, Yuekun Yao, and Najoung Kim. 2023a. SLOG: A structural generalization benchmark for semantic parsing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3213–3232, Singapore. Association for Computational Linguistics.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023b. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Zhuoyang Li, Liran Deng, Hui Liu, Qiaoqiao Liu, and Junzhao Du. 2024. Unioqa: A unified framework for knowledge graph question answering with large language models. *Preprint*, arXiv:2406.02110.

Yuanyuan Liang, Keren Tan, Tingyu Xie, Wenbiao Tao, Siyuan Wang, Yunshi Lan, and Weining Qian. 2024. Aligning large language models to a domain-specific graph database for nl2gql. *Preprint*, arXiv:2402.16567.

Lunyiu Nie, Shulin Cao, Jiaxin Shi, Jiuding Sun, Qi Tian, Lei Hou, Juanzi Li, and Jidong Zhai. 2022. GraphQ IR: Unifying the semantic parsing of graph query languages with one intermediate representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5848–5865, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. Text2Cypher: Bridging natural language and graph databases. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, pages 100–108, Abu Dhabi, UAE. International Committee on Computational Linguistics.

Marko A. Rodriguez. 2015. The gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages*, DBPL 2015, page 1–10, New York, NY, USA. Association for Computing Machinery.

Aman Tiwari, Shiva Krishna Reddy Malay, Vikas Yadav, Masoud Hashemi, and Sathwik Tejaswi Madhusudhan. 2025. Auto-cypher: Improving LLMs on cypher generation via LLM-supervised generation-verification framework. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 623–640, Albuquerque, New Mexico. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Ziyu Zhao, Wei Liu, Tim French, and Michael Stewart. 2023a. Cyspider: A neural semantic parsing corpus with baseline models for property graphs. In *Australasian Joint Conference on Artificial Intelligence*, pages 120–132. Springer.

Ziyu Zhao, Wei Liu, Tim French, and Michael Stewart. 2023b. Rel2graph: Automated mapping from relational databases to a unified property knowledge graph. *arXiv preprint arXiv:2310.01080*.

Zijie Zhong, Linqing Zhong, Zhaoze Sun, Qingyun Jin, Zengchang Qin, and Xiaofan Zhang. 2025. SyntheT2C: Generating synthetic data for fine-tuning large language models on the Text2Cypher task. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 672–692, Abu Dhabi, UAE. Association for Computational Linguistics.

A Hyperparameter details

Regarding the fine-tuning experiments we do an initial parameter selection by reserving a small subset of the data, and then proceed to train on the full dataset for up to 20 epochs. The parameters that are usually selected are the following: for *Llama-3.2-3B* batch size of 48 and learning rate of 0.00003, for *Qwen3-4B* batch size of 48 and learning rate of 0.00005 and for *Mistral-7B-v0.1* batch size of 64 and learning rate of 0.000005. We train the model on a single A100 GPU and every result reported is the average of multiple runs with different seeds. In the worst case a training run would usually take 2 hours.

For the zero-shot and ICLs settings where models are used in inference-only we run experiments on a V100 GPU. For the fixed ICL setting we also repeat the experiments sampling different set of fixed ICL examples and then report the average.

B LLM Prompt

```
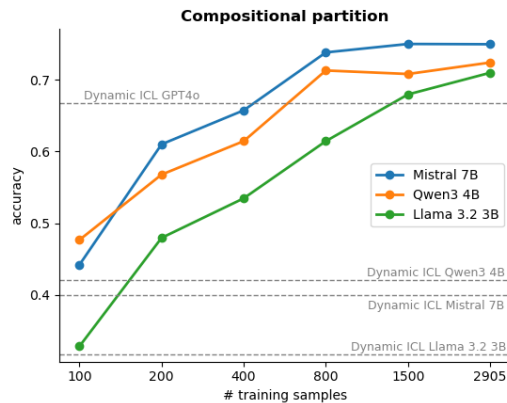###
Cypher schema:

CLASS: description
Person: People
Location: Locations
Phone: Phone
Email: Email
Officer: Officers
PostCode: PostCode
Area: Areas
PhoneCall: Phone calls
Crime: Crimes
Object: Criminal Objects
Vehicle: Vehicles

PROPERTY: description
year: year
postcode: post code
call_time: time
nhs_no: national health service number
address: address
name: name
phoneNo: phone number
model: model
badge_no: badge number
areaCode: area code
rank: rank
type: type
call_date: call date
call_duration: call duration (seconds)
email_address: email
make: car brand
date: date
surname: last name
code: code
last_outcome: processing status
age: age

RELATION: description - domain -> range
CURRENT_ADDRESS: that lives in - Person -> Location
HAS_PHONE: which has - Person -> Phone
HAS_EMAIL: which has - Person -> Email
KNOWS_SN: that is friends with - Person -> Person
KNOWS: who knows - Person -> Person
HAS_POSTCODE: which has - Location -> PostCode
POSTCODE_IN_AREA: that is in - PostCode -> Area
INVOLVED_IN: that is involved in - Vehicle -> Crime
CALLER: that were made to - PhoneCall -> Phone
CALLED: that were received a - PhoneCall -> Phone
KNOWS_PHONE: knows the phone of - Person -> Person
OCCURRED_AT: that occurred at - Crime -> Location
INVESTIGATED_BY: that is investigated by - Crime -> Officer
PARTY_TO: which is involved in - Person -> Crime
FAMILY_REL: that has a family relation with - Person -> Person
KNOWS_LW: that lives with - Person -> Person
LOCATION_IN_AREA: that is included - Location -> Area
###

# Task
Give me the cypher neo4j query for the following question. (Answer with the query
only, do not add anything else).

[[ICL EXAMPLES IF AVAILABLE]]

# Sample
Question: [[QUESTION]]

Query:
```