Large Language Models as Generalizable Text-to-Table Systems

Steven Coyne[1]* Yuyang Dong[2]
[1]Tohoku University [2]NEC Corporation
coyne.steven.charles.q2@dc.tohoku.ac.jp
dongyuyang@nec.com

Abstract

Wu et al. [1] formalized the task of text-to-table generation, in which unstructured texts are summarized in table form. However, their experiments focus on fine-tuned, dataset-specific models that show limited performance outside of the data they were trained on. As the task assumes no user query, systems ideally generate valid tables across various input texts without favoring the layout or table count of a particular dataset. To address this, we apply prompted large language models (LLMs) to this task, assessing their suitability as generalizable text-to-table systems. To more accurately rate outputs, particularly in a zero-shot setting, we revise the evaluation script from [1] to align tables by column similarity before scoring.

1 Introduction

Text-to-table generation is a sequence-to-sequence task in which one or more structured tables are generated based on the content of an input text [1]. The input can be a sentence or a longer document. The output is a table that attempts to represent the information in the input text. An example using a basketball game report from the RotoWire dataset [2] can be seen in Figure 1.

In this task, there is no explicit user query. Tables are designed based only on the input text. This means that the task is at least partially open-ended, as there may be multiple valid tables or sets of tables that could accurately represent an input text. As a result, the "ground truth" reference tables in the datasets are not definitive.

As the goal of the text-to-table setting is to convert an input text into a "valid" table that makes sense with respect to the input, it is a high-level logical task that may depend on world knowledge and common sense reasoning. Examples of the sort of inferences that may be done as part of the table design process include the following:

Input Text:

The Wizards launched yet another comeback on Tuesday, this time feasting on the relatively inexperienced Los Angeles Lakers. Washington not only overcame a 13 - point fourth quarter deficit, but won by double digits as well. The team shot over 51 percent from the field on the night and outscored LA 37 - 13 in the fourth. At the crux of the win was All-Star point guard John Wall, who scored 34 points to go along with 14 assists and four steals. Wall was an excellent 14 - of - 25 from the field. LA, meanwhile, saw promising play from its own point guard. Sophomore D'Angelo Russell filled the stat sheet respectably, shooting 10 - of - 21 for 28 points to go along with nine assists and six rebounds. His backcourt partner, Jordan Clarkson, added another 22 points and shot 10 - of - 19.

Table:

Team:
	Percentage of field goals	Points in 4th quarter
Lakers	51	13
Wizards		37
Player:
	Assists	Field goals attempted	Field goals made	Points	Total rebounds	Steals
John Wall	14	25	14	34		4
D'Angelo Russell	9	21	10	28	6	
Jordan Clarkson		19	10	22		

Figure 1 Example of an input text and reference table in the text-to-table task. The table format is adapted from [1], with their <NEWLINE> separator replaced with line breaks for readability.

Entity Linking: An entity may be referred to in multiple ways due to nicknames, metonymy, and other phenomena. A basketball team may be referred to by its name or by its home city. There may be a switch between these partway through the text. The model must resolve these references.

Implicit Reasoning: Some information in a text may be implied by other facts. If a basketball game is won 120-90, and the winning team is named, then the model must assign 90 to the losing team even if no sentence explicitly describes this.

Contextual Understanding: Depending on the input text, the "best" way to represent information may differ significantly. In a biographical description, information may be best summarized as a one-row table in which other entities are listed only in relation to the main entity the text is focused on.

Such principles may be learned as part of a fine-tuning process with respect to a particular dataset of relatively similar texts and tables. This can be seen in previous works such as [1] and [3]. However, these models are limited to the specific datasets that they were trained on and may

* The work was done when Steven Coyne was an intern at NEC.

show poor performance on tables that do not resemble the training data's format or use similar header names.

Given their strong performance across a number of reasoning tasks [4], we hypothesize that LLMs such as GPT-3 [5] can perform this task in a generalizable manner without the need to fine-tune on hundreds or thousands of examples, taking advantage of principles such as chain-of-thought reasoning [6] or in-context learning [5] to address the aforementioned reasoning challenges. To this end, we experiment with LLM prompting strategies using several different task decomposition settings, which we apply uniformly to each of the text-to-table datasets designated in [1] to test the generalized performance of the selected models.

2 Related Work

Previous works have examined the use of LLMs to handle table-based tasks. Li et al. [7] apply base and fine-tuned GPT-3.5 models to diverse table tasks, showing strong performance and generalizability to new data and tasks. However, their study does not include the text-to-table task described in [1]. Tang et al. [8] analyze the text-to-table setting using a variety of prompted LLMs, generating tables in raw text, LaTeX, and HTML. Their work focuses on the challenges of evaluation in this task, rather than the generalizability of LLMs compared to seq2seq baselines across datasets. Our work also differs in using a JSON format and in its approach to subtask decomposition.

3 Methodology

3.1 Baseline and Datasets

As a baseline, we use a system in [1], a seq2seq model based on BART [9]. We use the same datasets as their work: E2E [10], WikiTableText [11], WikiBio [12], and RotoWire [2]. As WikiBio is large, we use a 10% subset in our experiments to control costs. We reproduce the experiments in [1] based on their code,[1)] utilizing the "table constraint" and table relation embedding features. However, to align with our other experiments, we modify the output format and scoring script as described below.

3.2 Output Format

The system described in [1] uses an idiosyncratic format that differs between the RotoWire dataset and the other three used in their study. To unify handling of all datasets, and hypothesizing that a more common structured format may be easier for LLMs to generate consistently, we use a JSON format for our experiments. Furthermore, a JSON format permits experimentation with JSON-specific output parameters in our selected LLMs.

3.3 Evaluation

For evaluation, we use the scorer released as part of [1]. This scorer parses tables into relations and scores them with three metrics: exact match, chrF [13], and BERTScore [14]. We modify this scoring script to handle JSON inputs as part of the format change. We also create a JSON conversion script to allow the baseline system outputs reproduced from [1] to be evaluated by the adapted scorer.

In addition to format adaptations, we further modify the scoring script to allow tables in a multiple-table setting to be aligned by similarity. The original script can only match such tables by name exactly, but this is unlikely in cases such as a zero-shot table generation, as a table name difference as small as "Team" and "team" would result in a score of zero. We implement a table-matching algorithm in which the candidate tables are aligned by the similarity of their column names. We use the highest average chrF metric across column names to align tables. We experimented with using BERTScore for this purpose, but the resulting scores were slightly lower. We hypothesize that cases like "Team" vs. "Teams," in which there is some surface similarity, outnumber cases such as "Player" and "Athlete" in which there are two distinct semantically similar words.

3.4 Our Method

We selected the OpenAI LLMs `gpt-3.5-turbo-1106` and `gpt-4-1106-preview` for our experiments. Since the outputs of our prompts are in a JSON format, we include experiments with the `response_format` parameter, performing the `gpt-3.5-turbo-1106` experiments in both text mode and JSON mode to compare them.

We experiment with decomposing the task of table generation, resulting in the following subtasks. Examples of the outputs of each can be seen in the Appendix.

Grouping: Entities in the text are listed and sorted into categories. The input is the text, and the output is one or more lists of entities wrapped in JSON.

Schema Generation: Based on the input text, and

1) Available at https://github.com/shirley-wu/text_to_table

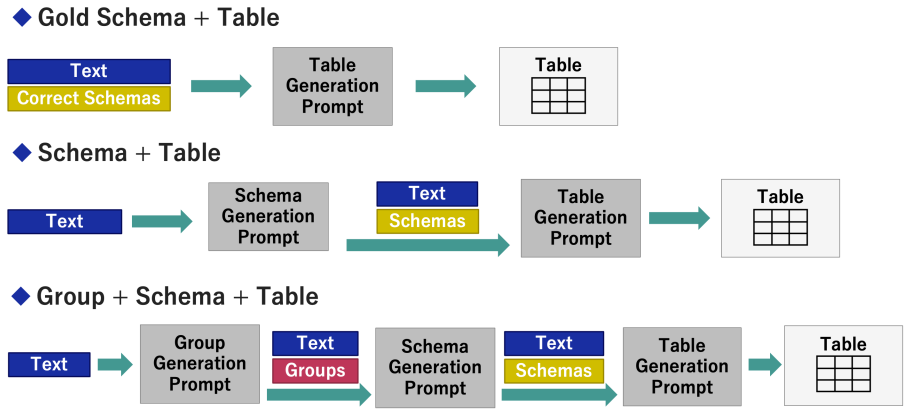

Figure 2 Prompt settings in our experiments. Experiments are performed with both few-shot and zero-shot prompts.

optionally the output of a grouping step, the model generates a list of one or more JSON table schemas. Each schema is associated with a planned output table.

Table Generation: Based on the input text and one or more input schemas, potentially the result of a schema generation step, the model generates tabular JSON. Each table row is represented by a JSON object, and all rows in a given table use identical fields. Multiple tables are nested within a top-level JSON object.

We used a prompt chaining approach in which each of these subtasks was handled by a separate call to the LLM. We defined both zero-shot and few-shot prompts for each step. The zero-shot prompts incorporated chain-of-thought [6]. In the few-shot prompts, example inputs and outputs were drawn randomly from the validation set and appended before the inputs. We used 5-shot prompting for all datasets in the GPT-3.5 experiments, plus additional 1-shot prompting for RotoWire. We created two sets of prompts for the schema generation step: one using grouping step outputs and one using only the input text.

4 Experiments

Our first experiments focus on the table generation step. To test the performance of the LLMs in this subtask alone, we establish a setting where the ground-truth "gold schema" is extracted from the reference file and provided along with the input text. Theoretically, this establishes an upper limit of performance for other pipelines. Following this, we performed pipeline experiments using generated schemas, both with and without grouping. This resulted in six total settings: zero-shot and few-shot versions of each of the three prompt settings seen in Figure 2.

Due to budgetary constraints, our GPT-4 experiments were limited to the RotoWire dataset using the schema + table and group + schema + table settings, using only JSON mode. We additionally limit few-shot experiments to one example to control costs. We selected RotoWire due to its small size and high complexity, being the only dataset with two-dimensional tables and multiple tables per input. We selected JSON mode after it showed slightly improved performance on RotoWire in the GPT-3.5 experiments.

5 Results and Discussion

Results can be seen in Tables 1 and 2. In the "gold schema" few-shot setting, the scores of the LLMs were comparable or even superior to those of the respective baseline fine-tuned model. However, it must be stressed that this setting involves a leak of ground-truth information.

In settings with generated schemas, the LLM scores are lower than the baseline model scores. However, the LLMs achieve moderate performance across all datasets, suggesting their capabilities are generalizable.

The LLMs display higher performance in few-shot settings compared to zero-shot, and in the case of RotoWire, higher performance at a higher number of examples.

The results for GPT-3.5 show relatively similar scores between the JSON and text output formats. Which format scores higher varies by dataset, with E2E showing higher scores in text mode and RotoWire showing higher scores in JSON mode. All differences are under two points, except for the gold schema zero-shot E2E exact match score.

Overall, the experiments that included a grouping step produced lower scores than the experiments using only schema and table steps. In RotoWire, the group + schema + table prompt chain performed better than the no-group chain in a few-shot setting, and the no-group chain per-

System/Prompts	Dataset	output_format: text						output_format: json					
		Header F1			Non-Header F1			Header F1			Non-Header F1		
		EM	Ch	BS	EM	Ch	BS	EM	Ch	BS	EM	Ch	BS
Baseline (Wu et al. 2022)	E2E	99.63	99.70	99.88	97.88	98.00	98.56	99.63	99.70	99.88	97.88	98.00	98.56
	WikiBio (subset)	78.70	83.37	91.98	66.83	75.25	74.63	78.70	83.37	91.98	66.83	75.25	74.63
	WikiTableText	76.14	81.97	93.13	57.34	66.95	78.18	76.14	81.97	93.13	57.34	66.95	78.18
	RotoWire	88.89	91.56	93.14	85.03	87.47	92.97	88.89	91.56	93.14	85.03	87.47	92.97
gpt-3.5-turbo-1106 zero-shot gold schema + table	E2E	**100**	**100**	**100**	18.46	56.28	69.56	**100**	**100**	**100**	18.54	56.37	69.55
	WikiBio (subset)	**99.17**	**99.35**	**99.63**	46.29	71.49	66.25	**99.27**	**99.46**	**99.78**	46.18	71.53	66.22
	WikiTableText	**99.78**	**99.81**	**99.93**	46.77	59.69	72.29	**99.78**	**99.81**	**99.96**	46.81	59.69	72.34
	RotoWire	**99.86**	**99.86**	**99.86**	75.61	83.74	**93.76**	**100**	**100**	**100**	75.35	83.62	**93.78**
gpt-3.5-turbo-1106 few-shot gold schema + table	E2E	**100**	**100**	**100**	72.63	84.25	88.45	**100**	**100**	**100**	69.01	83.04	87.11
	WikiBio (subset)	**99.70**	**99.70**	**99.71**	62.95	**80.36**	**77.49**	**99.96**	**99.97**	**99.97**	62.69	**80.23**	**77.38**
	WikiTableText	**99.76**	**99.79**	**99.94**	**59.03**	**69.63**	**78.99**	**99.89**	**99.90**	**99.98**	**58.24**	**69.02**	**78.38**
	RotoWire (1-shot)	**99.49**	**99.50**	**99.50**	**85.85**	**88.73**	**97.23**	**99.12**	**99.13**	**99.15**	**85.54**	**88.32**	**96.92**
	RotoWire (5-shot)	**100**	**100**	**100**	**88.32**	**90.37**	**98.37**	**100**	**100**	**100**	**88.09**	**90.19**	**98.32**
gpt-3.5-turbo-1106 zero-shot schema + table	E2E	21.36	42.11	77.46	2.38	18.08	41.87	21.27	42.04	77.42	2.40	18.09	41.94
	WikiBio (subset)	0.00	36.56	58.38	0.00	24.02	28.57	0.00	36.70	58.54	0.00	23.94	28.40
	WikiTableText	0.00	27.11	69.66	0.00	18.78	42.51	0.00	26.94	69.72	0.00	18.48	42.79
	RotoWire	23.57	49.61	55.10	16.40	40.93	44.19	23.67	49.31	55.19	16.60	40.79	43.96
gpt-3.5-turbo-1106 few-shot schema + table	E2E	91.70	93.42	98.89	70.23	80.12	86.27	91.65	93.40	98.73	68.47	79.90	85.48
	WikiBio (subset)	52.72	64.69	82.42	37.14	50.81	50.07	52.73	64.66	82.45	37.01	50.72	49.94
	WikiTableText	45.79	58.67	90.28	29.68	41.17	65.11	46.05	58.74	90.16	29.85	41.23	64.99
	RotoWire (1-shot)	52.70	64.69	67.98	51.47	60.96	67.53	53.84	65.67	68.60	52.79	61.95	68.53
	RotoWire (5-shot)	73.63	79.96	83.65	73.29	77.21	84.94	73.56	80.02	83.90	73.21	77.18	85.18
gpt-3.5-turbo-1106 zero-shot group + schema + table	E2E	23.84	41.95	74.00	1.83	16.45	30.86	23.79	42.04	74.11	1.93	16.65	31.23
	WikiBio (subset)	0.00	38.02	58.98	0.00	23.75	26.70	0.00	38.79	60.24	0.00	24.04	26.87
	WikiTableText	0.14	21.76	75.14	0.08	12.07	32.78	0.13	21.51	74.35	0.10	11.77	32.41
	RotoWire	11.16	43.69	39.75	8.43	35.03	28.05	10.95	43.03	39.25	7.97	34.10	27.47
gpt-3.5-turbo-1106 few-shot group + schema + table	E2E	90.20	92.21	98.52	68.44	78.50	85.43	90.20	92.20	98.38	66.76	78.38	84.61
	WikiBio (subset)	54.42	65.58	82.72	35.61	48.93	48.16	54.71	65.86	83.20	35.54	49.12	48.16
	WikiTableText	43.48	55.69	90.02	28.45	39.23	64.20	42.86	55.50	89.95	28.18	39.03	64.04
	RotoWire (1-shot)	61.29	72.04	76.11	60.53	68.84	76.62	61.51	72.33	76.18	60.53	68.92	76.84
	RotoWire (5-shot)	73.47	80.57	84.03	73.23	77.71	85.38	73.55	80.59	84.04	73.11	77.64	85.42

Table 1 Scores for `gpt-3.5-turbo-1106`. E, Ch, and BS refer to exact match, chrF, and BERTScore, respectively. output_format does not apply to baseline scores, so scores are simply repeated. Scores in bold are higher than the baseline, but it should be noted that the gold schema setting involves a leak of reference table information. Scores for RotoWire are the average of Team and Player scores.

System/Prompts	Header F1			Non-Header F1		
	EM	Ch	BS	EM	Ch	BS
Baseline	88.89	91.56	93.14	85.03	87.47	92.97
zero-shot S+T	23.59	53.11	52.70	19.38	43.91	44.14
one-shot S+T	43.35	64.33	66.00	42.26	60.23	63.63
zero-shot G+S+T	16.40	47.85	50.88	13.50	38.38	42.35
one-shot G+S+T	54.67	71.03	73.92	52.16	66.40	71.84

Table 2 Scores for `gpt-4-1106-preview` on the RotoWire dataset, using the average of Team and Player scores.

formed better in zero-shot. These trends were seen in both `gpt-3.5-turbo-1106` and `gpt-4-1106-preview`.

Interestingly, while GPT-4 outperformed GPT-3.5 in the zero-shot setting, the reverse was seen in the one-shot RotoWire experiments, where GPT-3.5 scored higher.

Finally, the models were able to reliably generate valid JSON in all experiments, even without using JSON mode.

6 Conclusion

The experiments suggest that our prompt settings can permit LLMs such as GPT-3.5 and GPT-4 to serve as relatively generalizable text-to-table generation systems. While their performance lagged behind the baseline fine-tuned systems in general, the LLMs show promise in this task, particularly in few-shot settings in which some basic information about the target table format can be provided. Additionally, the models' high performance in the gold schema experiments suggests that they may be able to follow user-defined schemas reliably. There remain many unexplored possibilities for prompting settings in this task, and our decomposition experiments are by no means exhaustive. Future works may explore additional approaches to prompting, format decisions, and table evaluation.

Acknowledgments

This work is mainly supported by NEC Corporation and partially supported by JSPS KAKENHI Grant Number JP22H00524.

References

[1] Xueqing Wu, Jiacheng Zhang, and Hang Li. Text-to-table: A new way of information extraction. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, **Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 2518–2533, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[2] Sam Wiseman, Stuart Shieber, and Alexander Rush. Challenges in data-to-document generation. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, **Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing**, pp. 2253–2263, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[3] Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. A sequence-to-sequence&set model for text-to-table generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, **Findings of the Association for Computational Linguistics: ACL 2023**, pp. 5358–5370, Toronto, Canada, July 2023. Association for Computational Linguistics.

[4] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. **arXiv preprint arXiv:2303.18223**, 2023.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, **Advances in Neural Information Processing Systems**, Vol. 33, pp. 1877–1901. Curran Associates, Inc., 2020.

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

[7] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table-tuned gpt for diverse table tasks, 2023.

[8] Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. Struc-bench: Are large language models really good at generating complex structured data?, 2023.

[9] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics.

[10] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In Kristiina Jokinen, Manfred Stede, David DeVault, and Annie Louis, editors, **Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue**, pp. 201–206, Saarbrücken, Germany, August 2017. Association for Computational Linguistics.

[11] Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. Table-to-text: Describing table region with natural language. **Proceedings of the AAAI Conference on Artificial Intelligence**, Vol. 32, No. 1, Apr. 2018.

[12] Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In Jian Su, Kevin Duh, and Xavier Carreras, editors, **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing**, pp. 1203–1213, Austin, Texas, November 2016. Association for Computational Linguistics.

[13] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, **Proceedings of the Tenth Workshop on Statistical Machine Translation**, pp. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[14] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In **International Conference on Learning Representations**, 2020.

A Appendix

Here, we present an example of the JSON format of our prompt chains, extending the example from Figure 1. This particular example is converted from the RotoWire development set, not generated by one of our models.

Input Text

The Wizards launched yet another comeback on Tuesday, this time feasting on the relatively inexperienced Los Angeles Lakers. Washington not only overcame a 13 - point fourth quarter deficit, but won by double digits as well. The team shot over 51 percent from the field on the night and outscored LA 37 - 13 in the fourth. At the crux of the win was All-Star point guard John Wall, who scored 34 points to go along with 14 assists and four steals. Wall was an excellent 14 - of - 25 from the field. LA, meanwhile, saw promising play from its own point guard. Sophomore D'Angelo Russell filled the stat sheet respectably, shooting 10 - of - 21 for 28 points to go along with nine assists and six rebounds. His backcourt partner, Jordan Clarkson, added another 22 points and shot 10 - of - 19.

Grouping Output Format

```
{
    "Team":[
        "Lakers",
        "Wizards"
    ],
    "Player":[
        "John Wall",
        "D'Angelo Russell",
        "Jordan Clarkson"
    ]
}
```

Schema Output Format

```
[
    {
        "title":"Team",
        "type":"object",
        "properties":{
            "Team":{
                "type":"string"
            },
            "Percentage of field goals":{
                "type":"integer"
            },
            "Points in 4th quarter":{
                "type":"integer"
            }
        }
    },
    {
        "title":"Player",
        "type":"object",
        "properties":{
            "Player":{
                "type":"string"
            },
            "Assists":{
                "type":"integer"
            },
            "Field goals attempted":{
                "type":"integer"
            },
            "Field goals made":{
                "type":"integer"
            },
            "Points":{
                "type":"integer"
            },
            "Total rebounds":{
                "type":"integer"
            },
            "Steals":{
                "type":"integer"
            }
        }
    }
]
```

Table Output Format

```
{
    "Team":[
        {
        "Team":"Lakers",
        "Percentage of field goals":51,
        "Points in 4th quarter":13
        },
        {
        "Team":"Wizards",
        "Percentage of field goals":null,
        "Points in 4th quarter":37
        }
    ],
    "Player":[
        {
        "Player":"John Wall",
        "Assists":14,
        "Field goals attempted":25,
        "Field goals made":14,
        "Points":34,
        "Total rebounds":null,
        "Steals":4
        },
        {
        "Player":"D'Angelo Russell",
        "Assists":9,
        "Field goals attempted":21,
        "Field goals made":10,
        "Points":28,
        "Total rebounds":6,
        "Steals":null
        },
        {
        "Player":"Jordan Clarkson",
        "Assists":null,
        "Field goals attempted":19,
        "Field goals made":10,
        "Points":22,
        "Total rebounds":null,
        "Steals":null
        }
    ]
}
```