

A Broader Discussion

A.1 Limitations

Although this work presents a comprehensive exploration of the long-context table understanding problem, certain limitations could be alleviated by future research. First, the proposed benchmark focuses on the single table in English. The long-context understanding in multi-table scenarios with broader language coverage deserves future explorations. Second, real-world applications such as document understanding could also require processing the hybrid content of tabular, textual, and image data. Therefore, it is valuable to explore the long-context understanding ability towards data consisting of multiple modalities. Third, we build the NIAT benchmark based on tables from public datasets to save cost, which primarily contain medium-sized tables. Future work could collect larger tables that span diverse structures to provide a longer context for long-context table understanding, especially tables with more than thousands of rows.

A.2 Ethical Considerations

The proposed NIAT benchmark is constructed based on the academic datasets like WTQ and TabFact, which are free and open datasets for research use with licenses like MIT License¹ or CC-BY-SA4.0 License². The resulting synthetic training data is also a free and open resource for the community to study long-structured table understanding. Considering that LLMs may generate harmful content, we used Llama3.1-70B-instruct to conduct LLM-as-a-judge to check the content of synthetic data and the generated data by API-available models, and we also randomly sampled a part of the data for manual checking and did not observe unsafe data in the synthetic samples. Thus, the authors foresee no ethical concerns with the research in this paper.

B Details of NIAT Benchmark

B.1 Data Statistics

To construct a benchmark for long-context understanding of tabular data at a low cost, we leveraged existing open-source datasets to sample tabular data. Table 1 represents the statistics results of sampled tables, we calculate the average length of tokenized tabular data. Figure 1 shows the distribution of input length of the NIAT benchmark.

Although the largest tables reach a length of 100k tokens, the average length of tables sampled from public dataset is not long enough. We anticipate that future researchers will build upon this work to develop more comprehensive benchmarks for long-table understanding tasks with longer tabular text.

Table Types	Format	Table Num.	Avg. Row Num.	Avg. Col Num.	Avg. Length	Max Length
Horizontal	HTML	325	6.43	26.78	1831.27	29791
	Markdown	358	6.38	25.77	981.92	16012
Flat	HTML	362	26.32	6.51	5226.04	103430
	Markdown	362	26.24	6.38	843.05	13203
Hierarchical	HTML	1086	19.81	8.87	878.39	5339
	Markdown	1086	19.81	8.87	1481	9420
Total	-	-	18.56	11.46	1594.32	103430

Table 1: Detailed table statistics of sampled tables of NIAT benchmark. The **Avg. Length** and **Max Length** represent the average and maximum token lengths of encoded tabular text for specific table categories. We use the tokenizer of Llama3.1-8B-Instruct to calculate the token length.

B.2 Prompt Templates of Evaluating LLMs

For text-LLMs and MLLMs, we adopt unified prompt templates to evaluate the performance of them. For Cell-Locating the prompt templates are shown in Figure 5, for Cell-Lookup task, the adopted template are provided in Figure 6.

¹<https://opensource.org/license/mit/>

²<https://creativecommons.org/licenses/by-sa/4.0/deed.en>

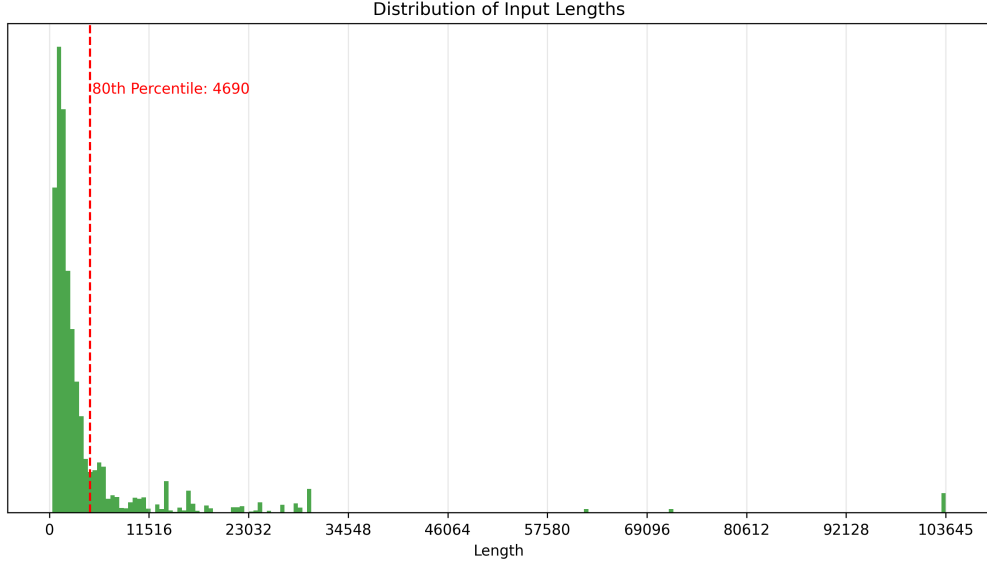


Figure 1: The distribution of input lengths for our proposed NIAT benchmark. The tokenizer of Llama3.1-8B-Instruct is adopted to calculate the token length.

Table 2: Performance of representative MLLMs on NIAT benchmark. ‘Flat’, ‘Hori.’ and ‘Hie.’ denote flat, horizontal and hierarchical tables, respectively. The best and suboptimal results are highlighted in **bold** and underlined.

Model	Cell-Locating				Cell-Lookup			
	Flat	Hori.	Hie.	Ave.	Flat	Hori.	Hie.	Ave.
GPT-4o	28.46	15.34	11.4	18.40	66.9	61.43	76.65	68.33
DeepSeek-VL2-tiny-3B	5.89	4.08	5.47	5.15	55.29	25.44	54.68	45.14
GLM-4V-9B-Chat	4.73	3.72	2.68	3.71	52.63	24.41	50.00	42.35
InternVL-2.5-8B-Chat	15.53	10.76	<u>12.44</u>	12.91	37.89	33.37	24.39	31.88
Llava-1.5-7B	0.69	0.56	0.93	0.73	2.71	4.63	1.50	2.95
Table-Llava-7B	7.65	4.90	11.62	8.06	9.58	5.59	2.68	5.95
Llava-1.6-7B	2.57	1.61	3.50	2.56	16.88	5.39	7.80	10.02
MiniCPM-V-2.6-8B	23.99	16.21	20.12	20.11	59.50	36.10	37.37	44.32
Phi-3.5-Vision-Instruct-4.2B	2.28	2.61	4.13	3.01	41.09	22.17	36.48	33.25
Phi-4-multimodal-instruct	4.06	4.49	6.47	5.01	57.86	31.22	26.87	38.65
Qwen2-VL-7B-Instruct	5.53	4.43	5.07	5.01	63.04	21.83	53.24	46.04
Qwen2.5-VL-3B-Instruct	8.01	5.45	7.77	7.08	59.71	38.02	65.96	54.56
Qwen2.5-VL-7B-Instruct	10.53	7.92	9.39	9.28	70.24	51.83	<u>72.32</u>	64.80
Llama-3.2-11B-Vision-Instruct	4.46	4.25	5.99	4.90	37.35	<u>36.46</u>	51.74	41.85

35 C Implementation Details

36 C.1 Implementation Details

37 To investigate the relationship between the fundamental table structure comprehension capabilities
38 of LLMs and their performance in downstream table understanding tasks, we selected four popular
39 datasets spanning various table understanding tasks. These datasets include **Table Question An-**
40 **swering** (HiTab, WTQ), **Tabular Numerical Reasoning** (TABMWP), and **Table Fact Verification**
41 (TabFact). Each language model was instructed to generate a chain-of-thought reasoning process
42 before providing its final answer. To evaluate performance, we calculated accuracy based on the last
43 30 characters of the generated content for open-source LLMs, which corresponds to the model’s final
44 answer.

45 We provide an example of a Cell-locating question and answer pair, along with the chain-of-thought
46 (CoT) reasoning process, in Figure 7.

47 C.2 Training Details

48 To efficiently fine-tune *Qwen2.5-7B-Instruct* and *Llama3.1-8B-Instruct* on our synthetic data, we
49 employ the NVIDIA Megatron framework. To avoid out-of-memory errors caused by long-context
50 input sequences, we configure tensor parallelism (tp) to 8 and model pipeline (pp) to 4. Each model
51 is fully fine-tuned on the mixed data for 2 epochs.

52 C.3 Evaluation Details

53 For the Data Mixing Strategy, we randomly sample 8,400 NIAT training instances and 6,497 table-
54 lookup question-answer pairs, combining them into the training dataset. The longest sample in
55 our training data reaches 92k tokens. **For evaluating all foundation models and the further**
56 **fine-tuned models**, we adopt the original generation configurations of the models and utilize the
57 vLLM framework for deployment on a machine equipped with 8 NVIDIA A100-80G GPUs, as the
58 input sequences can be extremely long.

59 C.4 Details of Cell-Lookup Synthesis Tasks

60 **Locate single cell** Given a row header and a column header, LLMs are tasked with identifying
61 the cell value at the intersection of these two headers. The uniqueness of the ground truth answer is
62 strictly enforced.

63 **Synthetic Data Generation Prompt Template:** f"In the table above, what is the element located in
64 the cell at the intersection of the row header «row header» and the column header «column header»?"

65 **Retrieve the Nth cell based on row header** Counting ability is crucial for LLMs when interpreting
66 tabular data. In this task, given a row header (a specific cell value in the first column), the LLM is
67 required to return the value of the Nth cell in the corresponding row.

68 **Synthetic Data Generation Prompt Template:** f"What is the Nth cell value with header row «row
69 header»?"

70 **Retrieve the Nth cell based on column header** : Similarly, LLMs are expected to locate the Nth
71 cell value within a column specified by a given column header. Both tasks require LLMs to retrieve
72 target cell contents simultaneously based on natural language queries and row/column IDs.

73 **Synthetic Data Generation Prompt Template:** f"What is the Nth cell value with header col «col
74 header»?"

75 **Retrieve all location IDs of a certain key** In some cases, cell values may be duplicated in a large
76 table. This task requires identifying the row and column IDs (location IDs) of all cells that contain
77 the target value.

78 **Synthetic Data Generation Prompt Template:** f"In the table above, how many cells contain the
79 value «cell to ask»?"

80 **Cell value counting** In this task, LLMs are asked to count the number of cells that contain a specific
81 value, further testing their ability to process and summarize tabular data.

82 **Synthetic Data Generation Prompt Template:** f"How many cell contains the value «random
83 element»? Please provide all row IDs and column IDs of all cells contain "«random element»".

84 **Table navigation** Understanding the relative positions of two different cells in a table is a chal-
85 lenging task for LLMs. Given a specific base cell position and row/column offsets, LLMs are
86 required to navigate the two-dimensional table by calculating the target position IDs and retrieving
87 the corresponding cell values.

88 **Synthetic Data Generation Prompt Template:**"You are provided with a two-dimensional table and
89 need to locate the content of a specific cell. The following information is given: Base Position: Row
90 index: «base row» Column index: «base col» Relative Position: Row offset: «relative row» Column
91 offset: «relative col» Search Instructions: If «relative row» > 0, move downwards from the base row
92 index. If «relative row» < 0, move upwards from the base row index. If «relative col» > 0, move
93 rightwards from the base column index. If «relative col» < 0, move leftwards from the base column
94 index. "Calculate the new target position («new row», «new col») using these offsets from the base
95 position, and return the content of the cell located at this new position"

96 **Prompt Template of 6 Synthesized Cell-Lookup tasks** We randomly sampled to examples of
 97 Cell-Lookup training data synthesized by GPT-4o are shown in Figure 8 (Retrieve the Nth cell based
 98 on column header) and 9 (Table navigation).

99 D Additional Experiments

100 D.1 Results on Cropped Tables

101 Table 6 presents the detailed performance metrics of various MLLMs. In Table 3, we compares
 102 LLMs fine-tuned on our proposed NIAT synthetic data against mainstream LLMs and TableLLMs
 103 on Cell-Locating tasks with cropped tables. This task demands a deep structural understanding of
 104 two-dimensional tabular data, so we evaluated model performance across varying context window
 105 sizes. Consistent with expectations, longer context windows improve accuracy in Cell-Locating tasks.

106 The **Qwen2.5 series** dominates performance, likely due to its specialized training on tabular data (as
 107 noted in its technical report). Notably, **Qwen2.5-7B-Instruct-1M**, a long-context variant, significantly
 108 outperforms its standard counterpart (**Qwen2.5-7B-Instruct**), suggesting benefits from structured
 109 pre-training and post-training data.

110 Among **Thinker LLMs**, DeepSeek-R1 achieves state-of-the-art results, leveraging its Long-CoT
 111 capability to accurately retrieve target cells in large 32×32 tables using row/column identifiers. This
 112 underscores the potential of test-time scaling techniques for processing long-structured tabular data.
 113 Surprisingly, Qwen2.5-7B-Coder-Instruct (with only 7B parameters) excels on NIAT tasks, likely
 114 due to: 1) Structural alignment between tabular data and code (both rely on separators for semantic
 115 organization). 2) Extensive pre-training on markdown, which may have included markdown tables,
 116 priming it for Cell-Locating NIAT tasks.

117 **Multi-Modal LLMs (MLLMs)**, which process table images directly, more closely mirroring human
 118 tabular understanding, demonstrate superior performance in Cell-Locating tasks. Notably, recent state-
 119 of-the-art models like Qwen2.5-VL-7B-Instruct and MiniCPM-V-2.6.8B significantly outperform
 120 earlier approaches (e.g., Llava-1.5-7B), while they still underperform GPT-4o with pure text input.

121 As illustrated in Figure 2, both mainstream **MLLMs** and **text-LLMs** exhibit the **lost-in-the-middle-**
 122 **tables** phenomenon—their performance degrades monotonically as table size increases. For MLLMs,
 123 this limitation arises because visual encoders struggle to accurately parse table structures when rows
 124 and columns expand beyond 32×32, leading to unreliable target cell retrieval. Nevertheless, recent
 125 MLLMs (e.g., MiniCPM-V-2.6-8B) show marked improvements in Cell-Locating tasks, suggesting
 126 advances in structural understanding.

127 D.2 Ablation Study

128 We conduct ablation experiments to investigate the impact of different training tasks systematically.
 129 The experimental results are presented in Table 5. We compare the foundation model fully trained on
 130 our proposed mixed synthetic data to variants trained on individual sub-tasks, as well as to models
 131 fine-tuned on the training splits of WTQ, TabFact, and HiTab, which serve as oracle experiments
 132 representing the performance ceiling.

133 **Fine-tuning on our synthesized training data achieves comparable performance to variants fine-**
 134 **tuned on three downstream datasets.** Surprisingly, when fine-tuned on our mixed synthetic data,
 135 *Llama3.1-8B-Instruct* achieves comparable accuracy on the TabFact dataset and even outperforms
 136 other models on the remaining downstream tasks. For variants trained solely on Cell-Locating and
 137 Cell-Lookup tasks (+ Cell-Locating and + Cell-Lookup), removing NIAT training data significantly
 138 drops performance across all four downstream tasks.

139 D.3 Case Study

140 We provide two cases of the model further fine-tuned on our synthesized data on **Cell-Locating** and
 141 **Cell-Lookup**, respectively in Figure 3 and 4.

Table 3: Performance results on Cropped Tables of mainstream text-LLMs, TableLLM, MLLMs, and foundation models fine-tuned on our synthetic data. The best results are highlighted in **bold**, and + *ours* denotes models further fine-tuned on our proposed synthetic data.

Model	Cell-Locating on Cropped Tables						
	8	12	16	20	24	28	32
Open Source LLMs							
GLM4-9b-chat	9.9	10.88	10.26	8.43	4.91	8.65	4.29
GLM4-9b-chat-1M	5.63	6.39	5.94	5.43	3.84	3.99	2.57
MiniCPM3-4B	1.98	4.35	5.73	5.75	4.28	9.51	5.02
InternLM2.5-7B	4.9	6.25	4.84	5.23	3.24	3.85	2.25
Mistral-7B-v0.3	8.54	10.65	9.17	6.2	3.47	5.49	5.81
Yi-1.5-9B-16K	12.81	8.98	5.55	5.67	3.32	4.53	2.12
Qwen2.5-Instruct							
Qwen2.5-7B-Instruct	26.35	18.98	15.91	12.83	7.55	9.72	2.9
Qwen2.5-7B-Instruct-1M	32.08	24.44	20.6	13.8	9.99	12.13	4.93
Qwen2.5-14B-Instruct	28.96	19.81	19.69	15.93	14.38	17.54	9.31
Qwen2.5-14B-Instruct-1M	39.69	33.84	33.83	26.7	20.98	25.54	16.54
Qwen2.5-7B-Instruct + ours	37.60	30.83	25.76	17.67	18.66	18.24	10.85
Llama3.1-Instruct							
Llama-3.1-8B-Instruct	16.88	13.56	12.94	8.98	6.34	7.25	3.83
Llama-3.1-8B-Instruct + ours	20.1	12.75	10.21	5.51	4.51	4.88	2.49
TableLLM							
TableLLM	-	-	-	-	-	-	-
StructLLM	5.62	5.23	4.29	4.02	2.07	3.26	0.92
TableGPT2	21.15	15.46	12.99	10.12	8.34	9.57	3.4
Thinker LLM and Coder LLM							
GPT-4o	50.21	33.36	30.89	24.15	27.54	26.01	17.32
QwQ-32B	51.15	45.23	43.91	14.57	6.01	6.32	6.56
DeepSeek-R1	73.33	64.63	64.60	41.87	38.07	54.06	51.17
Qwen2.5-7B-Coder	65.42	54.86	44.90	28.42	17.79	29.88	28.67
MLLMs							
GPT-4o(Text)	41.67	18.34	19.94	9.75	8.84	18.46	3.92
GLM-4V-9B	2.80	2.61	4.30	1.39	1.43	0.89	0.09
InternVL-2.5-8B	28.52	16.64	12.60	10.24	10.96	13.18	5.09
Llava-1.5-7B	2.68	4.31	1.19	0.20	1.26	0.39	0.30
Table-Llava-7B	1.67	2.37	2.65	2.12	2.97	1.82	2.01
Llava-1.6-7B	6.60	4.03	4.24	5.57	4.11	4.33	0.26
MiniCPM-V-2.6-8B	43.51	30.57	15.57	11.74	10.01	12.88	3.94
Phi-3.5-Vision-Instruct-4.2B	4.25	2.84	5.92	2.68	4.68	1.56	0.82
Phi-4-multimodal-instruct	6.04	8.86	8.84	5.77	6.12	8.69	3.68
Qwen2-VL-7B-Instruct	7.27	8.72	8.62	9.64	8.19	8.07	2.30
Qwen2.5-VL-3B-Instruct	15.66	13.60	10.95	10.53	9.70	8.11	2.67
Qwen2.5-VL-7B-Instruct	19.80	14.64	11.98	10.67	5.29	9.59	3.02
Llama-3.2-11B-Vision-Instruct	10.63	8.96	9.89	6.98	4.34	5.30	3.03

	Format	Avg. Queries Num.
Horizontal	HTML	317.56
	Markdown	376.33
Flat	HTML	290.57
	Markdown	372.67
Hierarchical	HTML	909
	Markdown	1041

Table 4: The average number of queries of question-based NIAT. We prompt GPT-4o to generate simple questions to extract cells of a given table and check the correctness of



Figure 2: The figure presents a NEEDLEINATABLE evaluation performance comparison on cropped tables between MLLMs and text-LLMs.

Table 5: Ablation results of different tasks further fine-tuned on *Llama3.1-8B-Instruct*. The best results are highlighted in **bold**. WTQ, TabFact, and HiTab are the three downstream datasets selected for fine-tuning with chain-of-thought (CoT) reasoning processes generated by GPT-4o on foundation LLMs, as listed in the table.

Model	NIAT							WTQ	TabFact	HiTab	TABMWP
	8	12	16	20	24	28	32				
<i>Llama3.1-8B-Instruct</i>	26.75	16.83	12.74	10.80	13.37	11.15	7.56	52.9	70	30.5	76.28
+ Cell-Locating & Cell-Lookup	24.35	17.33	14.75	13.29	15.83	14.27	8.89	67.43	78.57	49.41	72.99
+ Cell-Locating	19.76	13.49	12.06	12.96	14.17	11.44	8.29	67.33	67.45	33.44	70.5
+ Cell-Lookup	21.75	15.98	14.97	13.13	16.23	14.86	9.64	59	53.5	35	69.44
+ 4 downstream datasets	17.08	13.61	13.05	5.07	7.12	9.4	2.9	64.78	78.13	48.22	81.79

Table 6: Performance results of mainstream MLLMs on cropped tables. The best results are highlighted in **bold**.

Model	Cell-Locating NIAT task on cropped tables							
	8*8	12*12	16*16	20*20	24*24	28*28	32*32	Ave.
<i>Multi-Modal LLMs</i>								
<i>DeepSeek-VL2-tiny-3B</i>	2.8	4.08	2.46	4.27	3.47	2.49	1	2.94
<i>GLM-4V-9B</i>	2.8	2.61	4.3	1.39	1.43	0.89	0.09	1.93
<i>Llava-1.5-7B</i>	2.68	4.31	1.19	0.2	1.26	0.39	0.3	1.48
<i>Table-Llava-7B</i>	1.67	2.37	2.65	2.12	2.97	1.82	2.01	2.23
<i>Llava-1.6-7B</i>	6.6	4.03	4.24	5.57	4.11	4.33	0.26	4.16
<i>Phi-3.5-Vision-Instruct-4.2B</i>	4.25	2.84	5.92	2.68	4.68	1.56	0.82	3.25
<i>Phi-4-multimodal-instruct-5.6B</i>	6.04	8.86	8.84	5.77	6.12	8.69	3.68	6.86
<i>Qwen2-VL-7B-Instruct</i>	7.27	8.72	8.62	9.64	8.19	8.07	2.3	7.54
<i>Qwen2.5-VL-3B-Instruct</i>	15.66	13.6	10.95	10.53	9.7	8.11	2.67	10.17
<i>Qwen2.5-VL-7B-Instruct</i>	19.8	14.64	11.98	10.67	5.29	9.59	3.02	10.71
<i>Llama-3.2-11B-Vision-Instruct</i>	10.63	8.96	9.89	6.98	4.34	5.3	3.03	7.02
<i>MiniCPM-V-2.6-8B</i>	43.51	30.57	15.57	11.74	10.01	12.88	3.94	18.32
<i>InternVL-2.5-8B</i>	28.52	16.64	12.6	10.24	10.96	13.18	5.09	13.89
<i>GPT-4o</i>	41.67	18.34	19.94	9.75	8.84	18.46	3.92	17.27

An example of finetuned Llama3.1-8B-Instruct output:

To determine the value at the specified cell, we will follow these steps:

1. **Identify the Row:**
 - We are looking for the row with a row ID of 8. In the provided table, the rows are numbered from 1 to 8, corresponding to each province listed.
 - The row with a row ID of 8 corresponds to the province "Seville".
2. **Identify the Column:**
 - We are looking for the column with a column ID of 1. The columns are labeled as follows:
 - Column 1: Province
 - Column 2: Capital
 - Column 3: Population
 - Column 4: Density
 - Column 5: Municipalities
 - Column 6: Legal districts
3. **Locate the Cell:**
 - With the identified row (Seville, row ID 8) and column (Province, column ID 1), we locate the cell at the intersection of these two.
 - The cell at row ID 8, column ID 1 contains the value "Seville".
4. **Output the Result:**
 - The cell value located at the specified position is "Seville".

Therefore, the final answer is:

```

{
  "row_ID": 8,
  "column_ID": 1,
  "cell_value": "Seville"
}

```

This matches the provided ground truth answer, confirming that the value in the specified cell is "Seville".

Figure 3: A case response from Llama3.1-8B-Instruct, further fine-tuned on synthesized data, demonstrating its performance on the Cell-Locating task.

An example of finetuned Llama3.1-8B-Instruct output:

To determine the opponent in Week 16, we need to carefully examine the table provided. Let's go through a step-by-step reasoning process:

- 1. Identify Week 16:**
 - We need to locate Week 16 in the table. The weeks are listed sequentially from Week 1 to Week 17. We look for the entry that corresponds to Week 16.
- 2. Locate the Date for Week 16:**
 - In the table, the date for Week 16 is December 23, 2007.
- 3. Find the Opponent for Week 16:**
 - Once we have identified the correct week, we look at the "Opponent" column for that row. For Week 16, the opponent listed is "Philadelphia Eagles."
- 4. Verify the Details:**
 - To ensure accuracy, we check the other details in the same row to confirm consistency. The "Game Site" is "Louisiana Superdome," the "TV" channel is "Fox," and the "Result/Score" is "L 38-23," which matches the pattern of losses (L) followed by a score in the format of "Score - Score."
- 5. Conclusion:**
 - Based on the information extracted from the table, the opponent in Week 16 is confirmed to be the Philadelphia Eagles.

Final Answer:

```
```json
{{
 "answer": "Philadelphia Eagles"
}}
```
```

Figure 4: A case response from Llama3.1-8B-Instruct, further fine-tuned on synthesized data, demonstrating its performance on the Cell-Lookup task.

Read the table image carefully and retrieve the value from a specified cell, based on a given row ID and a column ID.

Let's first think step by step and then output a final answer in the following JSON format:

```
```json
{{
 "row_ID": "m",
 "column_ID": "n",
 "cell_value": "<Corresponding Cell Value>"
}}
```
```

Please note:

- Row and column IDs start from 1.
- The header row is considered as the first row, with a row_ID of 1, so ensure you include it when locating the target row.
- If the target cell is empty, the cell_value should be an empty string ("").

Now, retrieve the cell value at the position where the row_ID is {} and the column_ID is {}.

Your Response:

Figure 5: Prompt template for evaluating Cell-Locating Task.


```
Read the table image carefully and answer the following question.  
Let's first think step by step and then output a final answer in the JSON  
format:  
```json  
{
 "answer": "<the final answer to the question>"
}
```  
## Question:  
{}  
  
## Your Response:
```

Figure 6: Prompt template for evaluating Cell-Locating Task.

Question:

Retrieve the cell value at the position where the row_ID is 5 and the column_ID is 26.

Answer:

To find the target cell value in the given table, where the row_ID is 5 and the column_ID is 26, we should follow these detailed step-by-step instructions:

1. Identify the Header Row:

- The header row is the first row in the table and provides the names for each column. In the provided table, this is the first row of data:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---------|--|---------|--|-------------|--|----------------|--|----------|--|----------|--|--------------|--|----------|--|----------|--|----------|--|----------|--|--------------|--|----------|--|----------|--|------------|--|----------|--|----------|--|----------|--|----------|--|----------|--|----------------|--|----------|--|----------|--|--------------|--|---------|--|-------------|--|----------|--|---------|--|------------|--|--------------|--|----------|--|----------|--|
| | Name(s) | | Council | | Core County | | Other Counties | | 2006 Tot | | 2011 Tot | | % change Tot | | Tot area | | Tot dens | | 2006 Mun | | 2011 Mun | | % change Mun | | Mun area | | Mun dens | | 2009 elect | | 2006 Sub | | 2011 Sub | | Mun dens | | 2011 Tot | | 2006 Mun | | Other Counties | | 2011 Sub | | Tot area | | % change Mun | | Council | | Core County | | Mun area | | Name(s) | | 2009 elect | | % change Tot | | 2006 Sub | | 2011 Mun | |
|--|---------|--|---------|--|-------------|--|----------------|--|----------|--|----------|--|--------------|--|----------|--|----------|--|----------|--|----------|--|--------------|--|----------|--|----------|--|------------|--|----------|--|----------|--|----------|--|----------|--|----------|--|----------------|--|----------|--|----------|--|--------------|--|---------|--|-------------|--|----------|--|---------|--|------------|--|--------------|--|----------|--|----------|--|

2. Identify Row 5 (not skipping the header row):

- Based on the given instructions, the header should be counted as row 1. Thus, row_ID 5 corresponds to the fifth row in the table, including the header row.
- Looking at the provided table, the fifth row is:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|-----------|--|------|--|-----------|--|----------|--|--------|--|--------|--|-----|--|----|--|-------|--|--------|--|--------|--|-----|--|--------|--|-------|--|--------|--|-------|--|-------|--|-------|--|--------|--|--------|--|----------|--|-------|--|----|--|-----|--|------|--|-----------|--|-------|--|-----------|--|--------|--|-----|--|-------|--|--------|--|
| | Waterford | | City | | Waterford | | Kilkenny | | 49,213 | | 51,519 | | 4.7 | | 44 | | 1,171 | | 45,748 | | 46,732 | | 2.2 | | 38.35Z | | 1,219 | | 26,598 | | 3,465 | | 4,787 | | 1,219 | | 51,519 | | 45,748 | | Kilkenny | | 4,787 | | 44 | | 2.2 | | City | | Waterford | | 38.35 | | Waterford | | 26,598 | | 4.7 | | 3,465 | | 46,732 | |
|--|-----------|--|------|--|-----------|--|----------|--|--------|--|--------|--|-----|--|----|--|-------|--|--------|--|--------|--|-----|--|--------|--|-------|--|--------|--|-------|--|-------|--|-------|--|--------|--|--------|--|----------|--|-------|--|----|--|-----|--|------|--|-----------|--|-------|--|-----------|--|--------|--|-----|--|-------|--|--------|--|

3. Identify Column 26:

- Each vertical group of data is identified as a column. Starting from the first column, we count until we reach the 26th column as specified.
- According to the header, the columns are ordered as follows:
 1. Name(s)
 2. Council
 3. Core County
 4. Other Counties
 5. 2006 Tot
 6. 2011 Tot
 7. % change Tot
 8. Tot area
 9. Tot dens
 10. 2006 Mun
 11. 2011 Mun
 12. % change Mun
 13. Mun area
 14. Mun dens
 15. 2009 elect
 16. 2006 Sub
 17. 2011 Sub
 18. Mun dens
 19. 2011 Tot
 20. 2006 Mun
 21. Other Counties
 22. 2011 Sub
 23. Tot area
 24. % change Mun
 25. Council
 26. **Core County** <- Target Cell

4. Cell Location and Determination:

- In the fifth row, column 26 corresponds to the "Core County" column.
- The cell at the intersection of row 5 and column 26 in this row contains the value: "Core County."

Thus, following this step-by-step process, the value located at row_ID 5 and column_ID 26 is **Core County**.

Carefully review the table, question, and provided ground truth answer.

Table: `<table_str>`

Question: In the table above, what is the element located in the cell at the intersection of the row header 12 and the column header game?

Ground truth Answer: 12

Generate a detailed, step-by-step reasoning process to accurately answer the question. Conclude with the final answer.

GPT-4o Answer:

To determine the element located in the specified cell, follow these steps:

- 1. Understand the Table Structure:**
 - The table is organized with columns labeled as: `game`, `date`, `team`, `score`, `high points`, `high rebounds`, `high assists`, `location attendance`, and `record`.
 - Each row corresponds to a particular game's data, with the rows numbered from 1 to 13.
- 2. Identify the Cell to Examine:**
 - The question asks for the element at the intersection of the row header 12 and the column header `game`.
 - The row header refers to the game's sequence or identifier within this particular dataset; the row number is essentially the order of games played.
- 3. Locate Row 12:**
 - Go to the row labeled with the number 12 in the table. This corresponds to all the data associated with the 12th game played.
- 4. Find Column Header `game`:**
 - The `game` column is the first column in the table. This column indicates the game number or identifier for that particular set of data.
- 5. Extract the Information:**
 - At the intersection of row 12 and the column `game`, locate the number, which indicates the sequence or position of that game.
- 6. Interpret the Table Data:**
 - If we review the table, we can see that in row 12 under the column `game`, the cell contains the number "12". This means that the 12th row corresponds to the 12th game played.

Finally, using this step-by-step examination, the element located in the cell at the intersection of the row header 12 and the column header `game` is indeed:

Final Answer: 12

Figure 8: An example of the chain-of-thought (CoT) reasoning process for synthesized Cell-lookup tasks.

Carefully review the table, question, and provided ground truth answer.

Table: `<table_str>`

Question:
You are provided with a two-dimensional table and need to locate the content of a specific cell. The following information is given:

Base Position:
Row index: 32
Column index: 4
Relative Position:
Row offset: -17
Column offset: 0
Search Instructions:
If relative_row > 0, move downwards from the base row index.
If relative_row < 0, move upwards from the base row index.
If relative_col > 0, move rightwards from the base column index.
If relative_col < 0, move leftwards from the base column index.
Calculate the new target position (new_row, new_col) using these offsets from the base position, and return the content of the cell located at this new position

Ground truth Answer: One of the original 19 parishes.
Generate a detailed, step-by-step reasoning process to accurately answer the question. Conclude with the final answer.

GPT-4o Answer:
To determine the content of a specific cell using the given information, we need to follow the search instructions carefully. Let's go through each step:

1. **Identify the Base Position:**
 - The base row index is 32.
 - The base column index is 4.
2. **Understand the Relative Position:**
 - The row offset is -17, meaning we need to move 17 rows upwards from the base row.
 - The column offset is 0, meaning there is no movement in the columns, and we stay in the same column as the base column.
3. **Apply the Row Offset:**
 - Starting from the base row index of 32, move 17 rows upwards.
 - Calculate the new row index: $(32 - 17 = 15)$.
4. **Apply the Column Offset:**
 - The base column index is 4 and the column offset is 0, so the new column index remains 4.
5. **Locate the New Position in the Table:**
 - The new row index is 15.
 - The new column index is 4.
6. **Find the Content of the Cell at the New Position:**
 - Check the table at row index 15 and column index 4.
 - According to the table, this corresponds to the "Origin" column for the 16th parish in the list, which is East Baton Rouge Parish since the table index starts from zero.
7. **Verify and Extract the Content:**
 - The content at row index 15 and column index 4 is "One of the original 19 parishes."
8. **Conclusion:**
 - The content of the cell located at the new position is "One of the original 19 parishes."

Thus, the final answer is:
"One of the original 19 parishes."

Figure 9: An example of the chain-of-thought (CoT) reasoning process for synthesized Cell-lookup tasks.