Appendix A. UCI-DataFrameQA Dataset Generation with GPT-4

You are given a dataframe and are tasked with generating real-world questions and corresponding Pandas queries for specific roles (customer, data analyst, and company owner, etc.). The dataframe is described as follows:

- Name of dataframe: Automobile
- Description of dataframe: This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) its assigned insurance risk rating, (c) its normalized losses in use as compared to other cars. The second rating
... *(omit dataset description to save space)*

- Column information of dataframe:

#	Column	Dtype
0	normalized-losses	float
1	make	str
2	fuel-type	str

... *(omit column information to save space)*

- Sample rows of dataframe in CSV format:
normalized-losses,make,fuel-type,aspiration,num-of-doors,body-style,drive-wheels,engine-location,wheel-base,length,width,height,curb-weight,engine-type,num-of-cylinders,engine-size,fuel-system,bore,stroke,compression-ratio,horsepower,peak-rpm,city-mpg,highway-mpg,price,symboling
134.0,toyota,gas,std,2.0,hardtop,rwd,front,98.4,176.2,65.6,52.0,2540,ohc,4,146,mpfi,3.62,3.5,9.3,116.0,4800.0,24,30,8449.0,2
168.0,toyota,gas,std,2.0,hatchback,rwd,front,94.5,168.7,64.0,52.6,2204,ohc,4,98,2bbl,3.19,3.03,9.0,70.0,4800.0,29,34,8238.0,1
... *(omit sample rows of dataframe to save space)*

- Task:
Using the provided information about the dataframe, formulate 10 natural language questions for the data scientist category.

- Characteristics of the questions from data scientist category:
Questions tailored for individuals with an in-depth understanding of the dataset, possessing expertise in statistical and mathematical analysis. These questions should challenge their analytical skills, encouraging the use of advanced data manipulation and interpretation techniques. The focus is on extracting complex insights and patterns from the data.

- Guidelines:
- All questions must be solvable using the Pandas library in Python.
- Questions should encompass a wide range of Pandas operations, from basic to advanced functionalities.
- Questions must reflect the real-world interests of the specified role.

- Assumptions:
- The Pandas library has been imported as `pd`. You can reference it directly.
- The dataframe `df` is loaded and available for use.

- Response:
- Store each answer in a variable named `result`.
- Do NOT include comments or explanations in your response.
- Present your questions and code within the list.

- Expected Response Format:

```python
[
    {
        "question": "...",
        "query": "result = ",
    },
    {...},
    ...
]
```

Figure 1: **Sample Prompt for DataFrame QA Dataset Generation with GPT-4.**

Table 1: Characteristics of Questions from Three Different Roles Used in the Prompt.

Role	**Description of Question Characteristics**
Data Scientist	Questions tailored for individuals with an in-depth understanding of the dataset, possessing expertise in statistical and mathematical analysis. These questions should challenge their analytical skills, encouraging the use of advanced data manipulation and interpretation techniques. The focus is on extracting complex insights and patterns from the data.
General User	Questions designed for users who may not have specialized data analysis skills but are interested in the practical, consumer-oriented aspects of the data. These questions should be formulated based on the nature and context of the data, requiring inferential thinking about its potential end-users. Questions and queries should be structured to be somewhat open-ended, avoiding direct references to specific column names, thus introducing a level of interpretative ambiguity.
Data Owner	Questions aimed at individuals or entities who own or have created the data, with a focus on business-oriented insights. These questions should cater to their interest in understanding the broader business implications, trends, and strategic insights that can be derived from the data. The emphasis is on leveraging the data for decision-making, performance tracking, and identifying opportunities or areas for improvement within the business context.

Table 2: Sample Generated Question/Pandas Query Pairs of UCI-DataFrameQA Dataset.

Role	**Question**	**Pandas Query**
Data Scientist	How has the average weight of cars changed over the model years?	`result = df.groupby('model_year')['weight'].mean()`
Data Scientist	What is the distribution of tumor size for cases with recurrence events?	`result = df[df['Class'] == 'recurrence-events']['tumor-size'].value_counts()`
General User	Which cars have more than 6 cylinders?	`result = df[df['cylinders'] > 6]`
General User	What is the most common tumor size observed in the data?	`result = df['tumor-size'].mode()[0]`
Data Owner	What are the names of the cars with the top 3 highest fuel efficiencies in our dataset?	`result = df.nlargest(3, 'mpg')['car_name']`
Data Owner	What is the frequency of tumor sizes in the age group 50-59?	`result = df[df['age'] == '50-59']['tumor-size'].value_counts()`

Appendix B. Error Classification with GPT-3.5

Table 3: Description of Eight Pre-defined Error Classes.

Abbreviation	Error Classes	Description
String Error	String Matching and Comparison Errors	Errors in this class arise from improper handling of string comparisons, such as failing to use appropriate matching methods, not accounting for case sensitivity, whitespace, or special characters, and using exact matching where pattern recognition is required.
Access Error	Data Access and Bounds Errors	This class is for errors when data is accessed using an incorrect index or key, or when the index exceeds the bounds of the data structure, leading to `'index out of bounds'` or `'key not found'` errors.
Condition Error	Query Condition and Value Errors	This class covers errors where query conditions do not reflect the data accurately or the wrong values are used, resulting in no matches or incorrect results. It includes using incorrect column names or values and failing to match the query criteria with the dataset.
Type Error	Data Type and Operation Errors	This class includes errors from attempting operations between incompatible data types, using methods unsuitable for the data type, and applying aggregation functions incorrectly, often leading to type mismatches or operation errors on non-compatible data types.
Expectation Error	Expectation and Interpretation Errors	This class encompasses errors from a discrepancy between expected outcomes and actual results, which may stem from misinterpreting the output, data, or having incorrect expectations of the data's structure, leading to incorrect assumptions and results.
Structure Error	Data Structure Reference Errors	This class refers to errors arising from incorrect assumptions or references to the data's structure, such as referencing non-existent columns or misinterpreting the content of the data, leading to queries that do not align with the actual data format or content.
Function Error	Function and Method Usage Errors	Errors in this category result from misusing functions or methods outside their intended purpose, such as using a function designed for a specific operation in a context where it does not apply, or calling methods on objects they are not designed for.
Others	Others	The category to cover any errors that do not fit into the specific categories above, such as general mistakes in code logic or implementation that leads to unexpected results or errors.

- Task Description:
I am trying to process a DataFrame with Python to find out the answer to the question. My code is not producing the expected result, and I need assistance to identify the error type.

- Question:
how many schools or teams had jalen rose

- Sample Rows from DataFrame `df`:
,Player,No.,Nationality,Position,Years in Toronto,School/Club Team
0,aleksandar radojević,25,serbia,center,1999-2000,barton cc (ks)
1,shawn respert,31,united states,guard,1997-98,michigan state
2,quentin richardson,n/a,united states,forward,2013-present,depaul
3,alvin robertson,"7, 21",united states,guard,1995-96,arkansas
4,carlos rogers,"33, 34",united states,forward-center,1995-98,tennessee state

- My Code:
result = df.loc[df['Player'] == 'Jalen Rose', 'School/Club Team'].nunique()

- Actual Output of My Code:
0

- Expected Output:
1.0

- Error Class and Description
Error Class	Description
Query Condition and Value Errors	This class covers errors where query ...
...
(This table is omitted to save space. Details can be found in Table 6)

- Assumption:
- All the string values in the `df` are converted to lowercase.
- All the words in the question are converted to lowercase.
- I can only access the dataframe column names when I write the Python code. I don't have access to the data itself. So my code may not be correct if the question is unclear or the column names are unclear.

- Requirement:
- Classify the error type in my code from the error class table above. Choose all error classes from above and place your answer between the [ERROR] and [/ERROR] tags. For example:
[ERROR]
Query Condition and Value Errors
[/ERROR]

- Error could belong to multiple error classes.
- Provide a concise justification for your error classification decision. Place your answer between the [EXPLANATION] and [/EXPLANATION] tags. For example:
[EXPLANATION]
Your explanation
[/EXPLANATION]

- Don't include comments or explanations in your answer.

Figure 2: **Sample Prompt for Error Classification with GPT-3.5.** Provided are the question, sample rows in the target dataframe, generated Pandas query, execution result, ground truth answer, and 8 pre-defined error classes with definitions. GPT-3.5 is tasked with classifying the errors present in this example. In this case, the incorrect Pandas query generated by CodeLlama-34B erroneously capitalizes the player's name when querying the dataframe.

[ERROR]
Query Condition and Value Errors
String Matching and Comparison Errors
[/ERROR]

[EXPLANATION]
The error in the code is due to a combination of query condition and value errors and string matching and comparison errors.

In the code, the query condition `df['Player'] == 'Jalen Rose'` is used to filter the DataFrame based on the 'Player' column. However, the actual value in the 'Player' column is in lowercase, while the query condition is in title case. This results in no matches being found and the output being 0.

Additionally, the string matching and comparison error arises from not accounting for case sensitivity. The code is looking for an exact match of 'Jalen Rose', but the actual value in the DataFrame may be 'jalen rose' or 'JALEN ROSE' due to the assumption that all string values are converted to lowercase.

To fix the error, the query condition should be modified to `df['Player'].str.lower() == 'jalen rose'` to ensure case-insensitive matching.
[/EXPLANATION]

Figure 3: **Sample Response for Error Classification with GPT-3.5.** We demonstrate the outcome of employing GPT-3.5 to assist in identifying the type of error. The response accurately categorizes the error, identifying it as a String Matching and Comparison Error, as well as a Query Condition and Value Error.
The analysis by GPT-3.5 underscores the importance of using lowercase search criteria for proper nouns (e.g., `'jalen rose'` instead of `'Jalen Rose'`), and recommends the implementation of the `.str.lower()` function as a standard practice. This approach ensures uniform conversion of strings to lowercase within dataframes, thereby mitigating potential case sensitivity issues. This specific case serves as a prominent example of the effectiveness of LLMs in conducting detailed error classification and analysis, demonstrating their significant value in addressing complex data processing challenges in natural language processing research.

Appendix C. Examples of Challenges in DataFrame QA and Potential Solutions

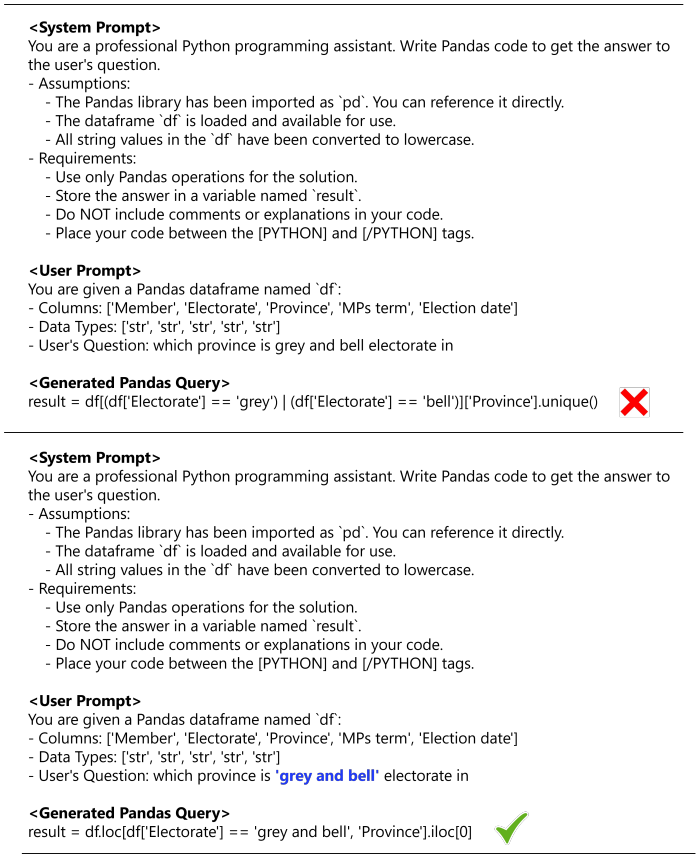

Figure 4: **Value Retrieval Error by GPT-4.** This type of error occurs in pandas queries when an incorrect value is retrieved. It often arises due to ambiguities in the user's question, leading to multiple possible interpretations. For instance, consider the example where the query targets the term `'grey and bell'` While the intended search might be for a combined entity, `'grey and bell'`, it could also be misinterpreted as two separate searches for `'grey'` and `'bell'` respectively. Another common occurrence in this error category involves GPT models inadvertently omitting special characters or symbols, such as Roman number `'I'` or hyphens, during query execution. This usually happens because the model mistakenly identifies these characters as typographical errors in the sentence, rather than integral parts of the search value. This represents one of the most common error categories in DataFrame QA tasks.
Solution: Enclosing query terms in quotation marks can significantly reduce Value Retrieval Errors. For instance, using quotations to specify `'grey and bell'` as a single entity, aiding in precise and accurate value retrieval.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']
- Data Types: ['int', 'int', 'int', 'int', 'int', 'int', 'int', 'int', 'int', 'float', 'int', 'float', 'float', 'int']
- User's Question: what is the variance in resting blood pressure (trestbps) among different heart disease severity levels?

<Generated Pandas Query>
result = df.groupby('ca')['trestbps'].var() ❌

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']
- Data Types: ['int', 'int', 'int', 'int', 'int', 'int', 'int', 'int', 'int', 'float', 'int', 'float', 'float', 'int']
- Column 'ca' includes the number of major vessels (0-3) colored by flourosopy.
- Column 'num' includes the diagonals of heart disease.
- User's Question: what is the variance in resting blood pressure (trestbps) among different heart disease severity levels?

<Generated Pandas Query>
result = df.groupby('num')['trestbps'].var() ✅

Figure 5: **Column Reference Error by GPT-4**. This error manifests when a query erroneously targets an incorrect column, commonly attributable to inadequately defined or ambiguous column names. Predominant in DataFrame QA tasks, this error underscores the imperative for a comprehensive understanding of table headers to facilitate precise data retrieval. LLMs, lacking prior domain-specific knowledge, are particularly susceptible to misidentifying columns, thereby yielding inaccurate outcomes. In the above example, the LLM fails to discern the significance of the `'num'` column, which is indicative of the diagnosis of heart disease. This oversight underscores the pivotal role of appropriate table header naming in ensuring the accuracy of DataFrame QA tasks.
Solution: Clarifying ambiguous columns in prompts can greatly reduce Column Reference Errors. For instance, stating `''Column 'ca' represents major vessels colored by fluoroscopy''` and `''Column 'num' indicates heart disease diagnoses''` guides LLMs to the correct data, enhancing query accuracy.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
- Data Types: ['str', 'float', 'float', 'float', 'float', 'float', 'float', 'float', 'int']
- User's Question: what are the mean and median lengths of abalone by each sex category?

<Generated Pandas Query> ✗
```
import pandas as pd

# Group the dataframe by sex
grouped_df = df.groupby('Sex')

# Calculate the mean and median lengths for each sex category
mean_lengths = grouped_df['Length'].mean()
median_lengths = grouped_df['Length'].median()

# Store the results in a dictionary
result = {'Mean Lengths': mean_lengths, 'Median Lengths': median_lengths}
```

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - **Do not import Pandas library.**
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
- Data Types: ['str', 'float', 'float', 'float', 'float', 'float', 'float', 'float', 'int']
- User's Question: what are the mean and median lengths of abalone by each sex category?

<Generated Pandas Query> ✓
```
result = df.groupby('Sex')['Length'].agg(['mean', 'median'])
```

Figure 6: **Instruction Misalignment Error by CodeLlama-7B.** This error type emerges when a LLM fails to follow or comprehend given instructions. A case in the above figure, despite explicit instructions in the prompt that the pandas library can be directly utilized and that the output should not include comments, the LLM deviates from this guideline. Case sensitive cases discussed in this paper also belong to this class. This indicates a misalignment with the provided instructions, reflecting a potential bias or conflict stemming from the LLM's training on datasets where import statements and comments are standard. In our specific DataFrame QA task, such inclusions are redundant and contrary to the task requirements. This scenario exemplifies the importance of a LLM's ability to adapt to the specific nuances and requirements of a given task, distinguishing between standard programming practices and task-specific directives.
Solution: Enhancing prompts with clear directives can effectively prevent Instruction Misalignment Errors. For example, specifying `'Do not import Pandas library'` alongside `'Pandas is pre-imported as pd'` emphasizes the need for LLMs to strictly follow given instructions, improving task adherence and accuracy.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Component', 'Integrated', 'Allied-Related', 'Allied-Unrelated', 'Holding']
- Data Types: ['str', 'str', 'str', 'str', 'str']
- User's Question: what is the total amount of allied-unrelated where the component is human capital?

<Generated Pandas Query>
result = df[df['Component'] == 'human capital']['Allied-Unrelated'].sum() ❌

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Component', 'Integrated', 'Allied-Related', 'Allied-Unrelated', 'Holding']
- Data Types: ['str', 'str', 'str', 'str', 'str']
- Column 'Allied-Unrelated' has categorical values.
- User's Question: what is the total amount of allied-unrelated where the component is human capital?

<Generated Pandas Query>
result = df[df['Component'] == 'human capital']['Allied-Unrelated'].count() ✅

Figure 7: **Aggregation Error by GPT-4.** This error arises in dataframe queries when a LLM incorrectly applies an aggregation function due to a lack of clear understanding of the specific meaning and content of a column, compounded by the ambiguity of the question. The crux of the error lies in the LLM's inability to distinguish whether a query requires the use of a summation (sum) or a count function. For instance, in response to a query like `‘What is the total amount under certain conditions?’`, it's vital to discern whether the query seeks the sum of all data meeting the criteria (using the sum() function) or merely the number of instances that satisfy the conditions (using the count() function). If a LLM does not fully grasp the nuances of the column's specific meaning and the data characteristics, or if it fails to accurately interpret the intent of the question, it may select an inappropriate aggregation function, leading to results that do not align with the actual requirements.
Solution: Providing clear column information and specific query formulations can effectively prevent Aggregation Errors. For instance, stating `‘‘Column ‘allied-unrelated’ holds categorical data’’` or rephrasing a query to `‘‘Count ‘allied-unrelated’ entries for human capital’’` guides the LLM to apply the correct aggregation method, enhancing result accuracy.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['State', 'Preliminaries', 'Interview', 'Swimsuit', 'Evening Gown', 'Average']
- Data Types: ['str', 'str', 'str', 'str', 'str', 'str']
- User's Question: what is the average score when the swimsuit score is 8.503?

<Generated Pandas Query>
result = df[df['Swimsuit'] == '8.503']['Average'].mean()

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['State', 'Preliminaries', 'Interview', 'Swimsuit', 'Evening Gown', 'Average']
- Data Types: ['str', 'str', 'str', 'str', 'str', 'str']
- User's Question: what is the **'average'** score when the swimsuit score is 8.503?

<Generated Pandas Query>
result = df[df['Swimsuit'] == '8.503']['Average']

Figure 8: **Function-Column Ambiguity Error by GPT-4.** This error manifests in dataframe queries when there is ambiguity between column names and function names, leading to erroneous interpretations and executions by the LLM. A typical instance of this error is seen when a column name includes terms like `'Average'` or `'Avg'` which can also be names of aggregation functions. This leads to an unnecessary aggregation operation instead of the simple data retrieval that the query required.
Solution: Renaming columns, such as changing to `'average score'`, may not rectify Function-Column Ambiguity Errors. A preferable approach is to encapsulate column names in quotation marks within queries, clearly differentiating them from function commands, thereby guiding accurate LLM interpretation.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Game', 'Date', 'Team', 'Score', 'High points', 'High rebounds', 'High assists', 'Location Attendance', 'Record']
- Data Types: ['float', 'str', 'str', 'str', 'str', 'str', 'str', 'str', 'str']
- User's Question: which team played on december 5?

<Generated Pandas Query>
```
df['Date'] = pd.to_datetime(df['Date'])
result = df[df['Date'] == 'december, 5']['Team']
```

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Game', 'Date', 'Team', 'Score', 'High points', 'High rebounds', 'High assists', 'Location Attendance', 'Record']
- Data Types: ['float', 'str', 'str', 'str', 'str', 'str', 'str', 'str', 'str']
- Column 'Date''s format is 'january, 1'.
- User's Question: which team played on december 5?

<Generated Pandas Query>
```
result = df[df['Date'] == 'december, 5']['Team']
```

Figure 9: **Insufficient Column Data/Format Information Error by GPT-4.** This error arises when there is a mismatch between the LLM's assumptions about a dataset's structure and the actual data format, leading to incorrect dataframe operations. A notable instance of this occurs when handling date-related queries without clear information on the date format in the dataset. LLM erroneously assumes that the `'Date'` column in the dataset contains full date information including the year. However, in reality, the dataset's `'Date'` column only contains month and day, without the year, leading to a failure in correctly applying the *to_datetime* function. In this case, the LLM's error stems from a gap in understanding the specific format of the `'Date'` column, highlighting the importance of having accurate and sufficient information about the data's structure and format. **Solution:** Specifying column formats in prompts, such as stating `''Column 'Date' follows the 'january, 1' format''` effectively addresses Insufficient Column Data/Format Information Errors, ensuring precise LLM data handling.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
- Data Types: ['str', 'float', 'float', 'float', 'float', 'float', 'float', 'float', 'int']
- User's Question: how does the average shell weight vary across different numbers of rings?

<Generated Pandas Query>
result = df.groupby('Rings').mean()['Shell_weight'] ✗ **CodeLlama-7B**

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
- Data Types: ['str', 'float', 'float', 'float', 'float', 'float', 'float', 'float', 'int']
- User's Question: how does the average shell weight vary across different numbers of rings?

<Generated Pandas Query>
result = df.groupby('Rings')['Shell_weight'].mean() ✓ **CodeLlama-34B**

Figure 10: **Coding Syntax Error by CodeLlama-7B.** This type of error highlights disparities in the coding capabilities of LLMs, particularly in structuring and executing DataFrame queries. It occurs when the syntax used in a query is incorrect or suboptimal, impacting the query's functionality and efficiency. In the above example, the `.mean()` function is applied across all columns in the grouped DataFrame before selecting the `'Shell_weight'` column. Such an approach is not just inefficient but also potentially problematic. If the DataFrame contains non-numerical columns, computing the mean for all columns initially can lead to errors, as the mean function is not applicable to non-numerical data. This kind of error emphasizes the challenges LLMs face in coding proficiency, particularly regarding the optimization of code for data manipulation tasks.
Solution: Addressing Coding Syntax Errors depends on the LLM's coding expertise. Solutions include choosing a base LLM with enhanced coding abilities, such CodeLlama-34B or GPT-4 or training the LLM on DataFrame QA datasets for better query optimization and data handling skills.

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
- Data Types: ['str', 'float', 'float', 'float', 'float', 'float', 'float', 'float', 'int']
- User's Question: i'm interested in knowing the most common age of abalone. can you find that for me?

<Generated Pandas Query>
result = df['Age'].mode()

<System Prompt>
You are a professional Python programming assistant. Write Pandas code to get the answer to the user's question.
- Assumptions:
  - The Pandas library has been imported as `pd`. You can reference it directly.
  - The dataframe `df` is loaded and available for use.
  - All string values in the `df` have been converted to lowercase.
- Requirements:
  - Use only Pandas operations for the solution.
  - Store the answer in a variable named `result`.
  - Do NOT include comments or explanations in your code.
  - Place your code between the [PYTHON] and [/PYTHON] tags.

<User Prompt>
You are given a Pandas dataframe named `df`:
- Columns: ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings']
- Data Types: ['str', 'float', 'float', 'float', 'float', 'float', 'float', 'float', 'int']
- The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope.
- User's Question: i'm interested in knowing the most common age of abalone. can you find that for me?

<Generated Pandas Query>
result = df['Rings'].mode()

Figure 11: **Hallucination Error by CodeLlama-7B.** This error type arises when a LLM generates responses based on incorrect assumptions or fabricated details, often due to a lack of domain-specific knowledge. In DataFrame queries, this manifests as references to non-existent columns or data points that the LLM 'hallucinates' or incorrectly infers. In the illstruated example, the accurate approach should involve using the `'Rings'` column, which typically represents the age of abalone. However, a Hallucination Error occurs when the LLM creates a query based on an imaginary `'Age'` column that doesn't exist in the dataset. This error is a result of the LLM's lack of understanding that in the context of abalone, age is commonly denoted by the number of rings, not a separate `'Age'` column. It demonstrates a significant gap in domain-specific knowledge, where the LLM fails to accurately interpret the data context and instead relies on incorrect or made-up information.
Solution: Enhancing prompts with detailed data and column information, like specifying `'Abalone age is assessed by counting rings on the shell'` helps bridge domain knowledge gaps in LLMs, effectively reducing Hallucination Errors and improving query accuracy.