What are **two** benefts of storing this data in MongoDB with JSON over a relational database management system such as Postgres? Please reference specific examples from the `business` collection to back up your claims. - Format your answer as follows: 1. Benefit #1, Example #1. 2. Benefit #2, Example #2.

Limit each benefit to 1 sentence and each example to 1 sentence for a total of at most four sentences.

MongoDB offers the significant advantage of accommodating data in various formats, including deeply nested elements within subdocuments and arrays. This flexibility surpasses that of Postgres, as MongoDB supports storing multi-valued attributes using JSON. For example, within a single column of the business collection, it's straightforward to store nested elements like "GoodForMeal" and "hours".

Additionally, the document-based data model of MongoDB is a major plus. It allows for the representation of complex structures and hierarchical relationships. In contrast to Postgres, MongoDB provides enhanced versatility and capability to present a unified view of large, diverse datasets through its document model. This feature is particularly useful for viewing all attributes related to a single business in the business collection at once, streamlining processes such as indexing, updating, and searching data.

0.0.1 Question 2d

In the last question, you performed equivalent left joins in both Postgres and Mongo. Now, examine their query plans, paying special attention to `executionTimeMillis`. Which join was faster? What gives that database system you chose an advantage over the other? Keep your response to at most three sentences.

The join in Postgres was faster, with an execution time of 4.036 ms, compared to the MongoDB query for which the execution time wasn't provided but typically would be slower for such operations. Postgres has a more efficient query optimizer for joins, leveraging advanced techniques like hash joins and sequential scans which are optimized for structured, tabular data. Additionally, Postgres's ability to use indexes effectively in joins provides a significant performance advantage over MongoDB, which is more oriented towards flexibility in data representation than optimizing join operations.

What do you notice about how the columns of `business_df` are constructed? How are values that are not found in every document handled in the pandas dataframe? Compare and contrast this dataframe representation with the document representation we saw with Mongo. Keep your response to at most two sentences.

In the business_df DataFrame, columns are constructed to represent each field in the MongoDB documents, including nested attributes, which are flattened. Values not present in every document are represented as NaN (Not a Number), indicating missing data; this contrasts with MongoDB's flexible schema where fields absent in some documents simply do not exist in those documents, highlighting the fixed-schema nature of DataFrames compared to the schema-less nature of MongoDB documents