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.

1. Flexible schemas: Attributes can store arbitrary key-value data like parking details and ambience without needing predefined columns, for example, "Ambience".
2. Nested objects: Hours can be stored as nested key-value documents instead of separate hour columns, for example, "Hours".

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 Postgres `left join` query took only a few milliseconds, while the Mongo `$lookup` query took a few hundred milliseconds. Postgres was clearly faster than Mongo, likely because SQL databases are efficient for relational joins between tables and query planning, while MongoDB is efficient for implementing flexible schemas and having nested documents, which makes joins more expensive in time.

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.

The dataframe has the same attributes corresponding to the features and has an attribute for each nested feature of the document representation in order to accomodate to the features that are represented by a set of more values rather than just one value. Values not found in every document are substituted with `NaN` in the pandas dataframe.