Schema Design

There are two tables in the database: the `Restaurant` table and the `InspectionResults` table. Below are brief descriptions of their schemas.

Restaurant

The `Restaurant` table contains the following fields. These fields uniquely defining a restaurant were mapped from the original restaurant inspection NYC Open Data.

Name	Type	Description
`camis`	Integer	Unique restaurant ID
`name`	String	Restaurant name
`boro`	String	Borough in which restaurant is located
`building`	String	Building number
`street`	String	Street name
`zipcode`	Integer	Zipcode
`phone`	Big integer	Phone number
`cuisine`	String	Description of cuisine

Additionally, there is a primary key that is auto-incremented.

The types of each field are fairly obvious given their roles in the database (the `name` of the restaurant is a string, whereas its `zipcode` is a number). Because phone numbers are quite long, the `phone` field is a Big Integer. Note that the `building` field is a string instead of a number, because sometimes building numbers contain a dash.

The string fields have maximum character lengths (typically 100) to accommodate longer fields present in the input data.

The `camis` field is used as a unique index into the table. This helps reduce the average query runtime, critical when loading large amounts of data into the table, which requires frequent lookups to see if a particular restaurant already exists before inserting it.

Because the full NYC Open Data also contain information about inspections (where one restaurant can be inspected several times, for example), the fully-loaded `Restaurant` table should be much smaller than the input data.

InspectionResults

The `InspectionResults` table contains the following fields. These fields uniquely define an inspection.

Name	Type	Description
`restaurant`	Foreign key	ID mapping uniquely to entry in `Restaurant` table
`inspection_type`	String	Type of inspection
`inspection_date`	Date	Date of inspection
`grade`	String	Grade (e.g., A, B)
`score`	Integer	Inspection score (the lower, the better)
`grade_date`	Date	Date grade was issued

Again, these map nicely to the input data, with obvious data types. The `restaurant` field is a foreign key that maps uniquely to a restaurant in the `Restaurant` table. One restaurant can have one or more inspections.

Note that certain inspection data in NYC Open Data like violation code and description are not captured in this table. This was intentional to reduce schema complexity and data loading times, as these data are not especially crucial to the application.

If these additional data were to be included, instead of simply adding them as additional fields into the `InspectionResults` table, it is possible to create two new tables that link to the current one using foreign keys. In that case, `Violations` and `Grades` can be separate tables. The first would contain inspection information like violation code, action, and description. The second would separate out the `grade` and `score` information already contained in `InspectionResults`. This more hierarchical approach gives nice structure to the data, but may be more difficult to understand or use.