Here are a few notes and tips for working with real world data sets.

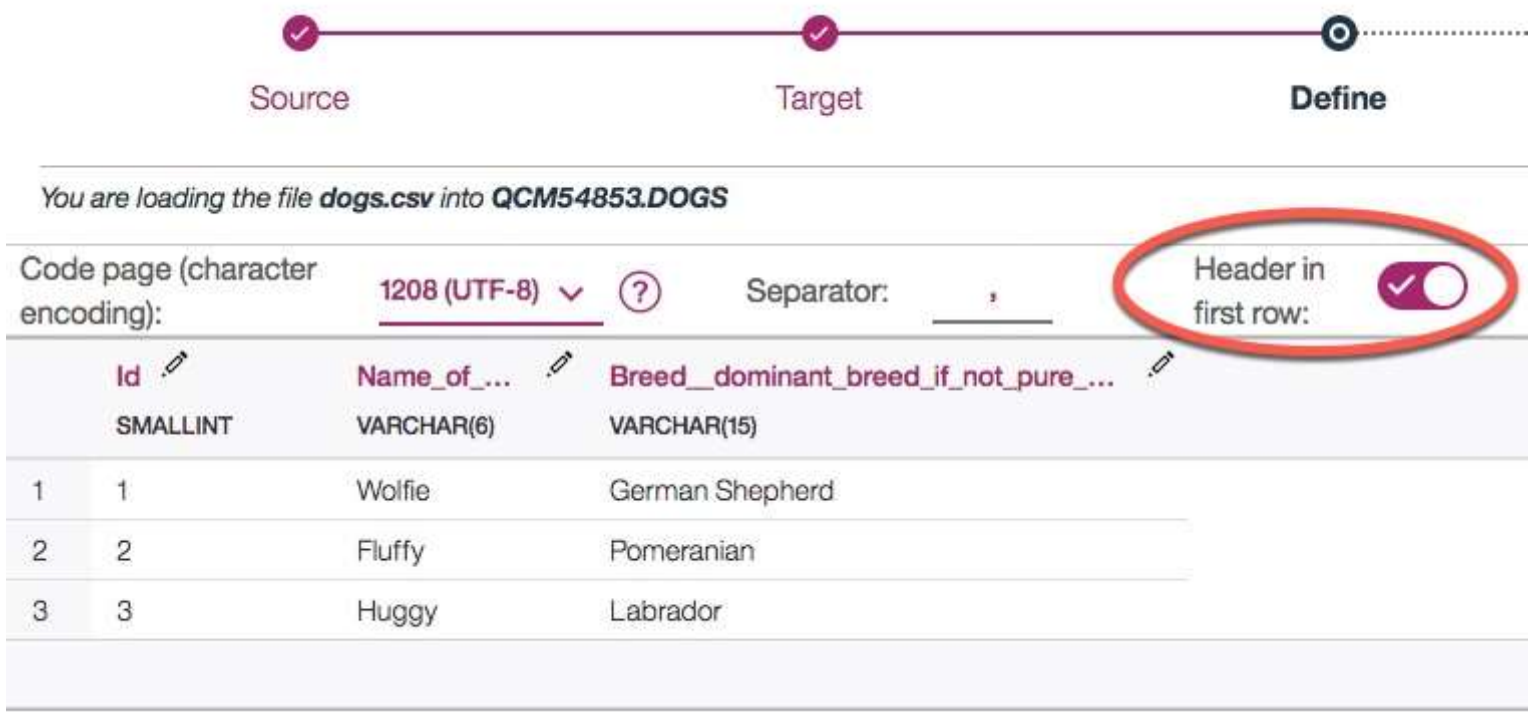
CSV files

In many cases data sets are made available as .CSV files which contain data values separated by commas. Here is an example of a file called "dogs.csv" which contains the following data:

```
Id,Name of Dog,Breed (dominant breed if not pure breed)
1,Wolfie,German Shepherd
2,Fluffy,Pomeranian
3,Huggy,Labrador
```

Column names

Sometimes the first row of the CSV file contains the name of the column. If you are loading the data into the database using the visual LOAD tool in the console, ensure the "Header in first row" is enabled. This will map the first row into column names, and rest of the rows into data rows in the table.



Note: that the default column names may not always be database friendly and if that is the case you may want to edit them before the table is created.

Querying column names with mixed (upper and lower) case

In the example above we loaded the csv file using default column names. This resulted in column names having mixed case. If you try to run the following query you will get an error:

```
1  select id from dogs
```

Error: "ID" is not valid in the context where it is used.. SQLCODE=-206, SQLSTATE=42703, DRIVER=4.22.36

This is because the database parser assumes uppercase names by default. To specify the mixed case column name, use double quotes (i.e. " " but not single quotes ' ') around the column name:

```
1  select "Id" from DOGS
```

Querying columns names with spaces and other characters

If in a csv file the name of the column contained spaces, e.g. "Name of Dog", by default the database may map them to underscores, i.e.: "Name_of_Dog". Other special characters like brackets may also get mapped to underscores. Therefore when you write a query, ensure you use proper case within quotes and substitute special characters to underscores:

```
select "Id", "Name_of_Dog", "Breed__dominant_breed_if_not_pure_breed_" from dogs
```

Note: the trailing underscore after breed_ that is put in place of the closing bracket.

Using quotes in Jupyter notebooks

You may be issuing queries in a notebook by first assigning them to Python variables. In such cases, if your query contains double quotes (e.g. to specify mixed case column name), you could differentiate the quotes by using single quotes ' ' for the Python variable to enclose the SQL query, and double quotes " " for the column names.

```
selectQuery = 'select "Id" from dogs'
```

What if you need to specify single quotes within the query (for example to specify a value in the where clause. In this case you can use backslash \ as the escape character:

```
selectQuery = 'select * from dogs where "Name_of_Dog"=\'Huggy\' '
```

Splitting queries to multiple lines in notebooks

If you have very long queries (such as join queries), it may be useful to split the query into multiple lines for improved readability. In Python notebooks you can use the backslash \ character to indicate continuation to the next row:

```
selectQuery = 'select "Id", "Name_of_Dog", \
  "Breed__dominant_breed_if_not_pure_breed_" from dogs \
   where "Name_of_Dog"=\'Huggy\''
```

Restricting number of rows retrieved

A table may contain thousands or millions of rows but you may only want to just sample some data sample or look at just a few rows to see what kind of data the table contains. You may be tempted to just do "select * from tablename", retrieve the results in a Pandas's dataframe and do a head() on it. But doing so may take a long time for a query to run. Instead you can limit the result set by using "LIMIT" clause. For example use the following query to retrieve just the first 10 rows:

```
select * from census_data LIMIT 10
```

Getting a list of tables in the database

Sometimes your database may contain several tables and you may not remember the correct name. For example you may wonder whether the table is called "DOG", "DOGS" or "FOUR_LEGGED_MAMMALS. Or you may have created several tables with similar names e.g. "DOG1", "DOG_TEST", "DOGTEST1" but want to check which was the last one you created. Database systems typically contain system or catalog tables from where you can query the list of tables and get their attributes. In Db2 its called "SYSCAT.TABLES" (in SQL Server its INFORMATION_SCHEMA.TABLES", and in Oracle: ALL_TABLES or USER_TABLES). To get a list of tables in a Db2 database you can run the following query:

```
select * from syscat.tables
```

The above will return too many tables including system tables so it is better to filter the result set as follows (ensure you replace "QCM54854" with your Db2 username):

```
select TABSCHEMA, TABNAME, CREATE_TIME from syscat.tables where tabschema
  ='QCM54853'
```

```
In [8]: %sql select TABSCHEMA, TABNAME, CREATE_TIME from syscat.tables where tabschema='QCM54853'
```

```
 * ibm_db_sa://qcm54853:***@dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net:50000/BLUDB
Done.
```

Out[8]:

tabschema	tabname	create_time
QCM54853	CENSUS_DATA	2018-05-30 20:47:46.028788
QCM54853	CHICAGO_CRIME_DATA	2018-05-30 20:29:35.452514
QCM54853	CHICAGO_PUBLIC_SCHOOLS	2018-05-30 19:44:20.616858
QCM54853	CHICAGO_SOCIOECONOMIC_DATA	2018-05-04 17:30:38.383376
QCM54853	DEPARTMENTS	2018-05-02 16:55:23.232836
QCM54853	DOGS	2018-06-20 20:38:49.170924
QCM54853	EMPLOYEES	2018-05-02 17:50:47.397152

FYI, in MySQL you can simply do "SHOW TABLES".

Getting a list of columns in a table

If you can't recall the exact name of a column, or want to know its attributes like the data type you can issue a query like the following for Db2:

```
1  select * from syscat.columns where tabname = 'INSTRUCTORS'
```

or:

```
1  select distinct(name), coltype, length from sysibm.syscolumns where tbname =
     'INSTRUCTORS'
```

```
In [12]: %sql select distinct(name), coltype, length \
             from sysibm.syscolumns where tbname = 'CHICAGO_CRIME_DATA'
```

```
 * ibm_db_sa://qcm54853:***@dashdb-txn-sbox-yp-dal09-04.services.dal
Done.
```

Out[12]:

name	coltype	length
Arrest	VARCHAR	5
Beat	SMALLINT	2
Block	VARCHAR	35
Case_Number	VARCHAR	8
Community_Area	DECIMAL	4
Date	VARCHAR	22
Description	VARCHAR	46
District	DECIMAL	4
Domestic	VARCHAR	5
FBI_Code	VARCHAR	3

FYI, In MySQL you can run "SHOW COLUMNS FROM TABNAME".

Mark as completed