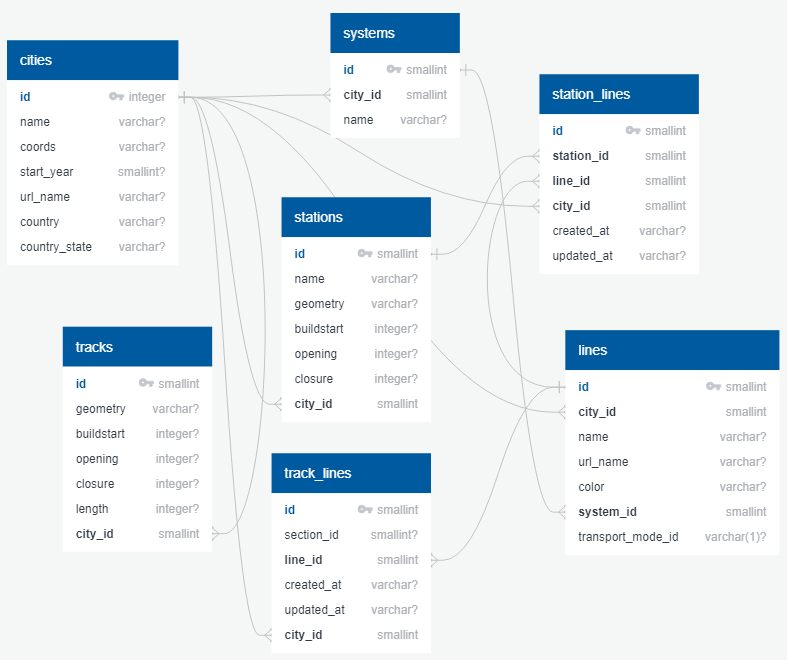
**Project to Extract, Transform, and Load data from Transit Systems of the World**

**Carlos Moreno, Ramesh Kalagnanam, Kent Bryant - UNC Boot Camp Data, August 17, 2019**

A dataset of transit systems of world cities was obtained at <https://www.citylines.co/data>. The data is stored in 7 .csv files, apparently derived from .geojson files for individual cities, which are also available on the site. The 7 files are:

* cities.csv
* systems.csv
* lines.csv
* stations.csv
* station\_lines.csv
* tracks.csv
* track\_lines,.csv

The csv files were examined, and an entity relationship diagram (ERD) was created using QuickDatabaseDiagrams, and SQL code was generated from the ERD to enable quick creation of a database in Postgres, which enabled further analysis of the raw data using Postgres Query Editor.

**Data Extraction:**

Using the Postgres Query Editor, it was found that the cities table contained far more cities (334), than for which there was corresponding data on any of the other tables; i.e. for the station\_lines table:

“SELECT COUNT(DISTINCT city\_id) FROM station\_lines” - (returned 81 rows)

A Jupyter notebook was created and the original .csv files were read into dataframes. The same steps of establishing the limited number of cities (81) was done in pandas, i.e.:

list\_cities = list(station\_lines\_df["city\_id"].unique())

**Data Transformation:**

The station\_lines table in fact had the fewest cities represented among all of the tables, so it was decided to limit the entire dataset to records for those 81 cities. In pandas for example

station\_lines\_df1 = station\_lines\_df[station\_lines\_df["city\_id"].isin(list\_cities)]

This was the first major step of data cleaning. Further cleaning of the data is as follows:

cities dataframe:

* created ‘lon’ and ‘lat’ fields from the original ‘coords’ field and removed ‘coords’; i.e. new\_city\_df = city\_data\_df["coords"].str.split("(", n = 1, expand = True), city\_data\_df["Lat"] = new2\_city\_df["Lat"]
* removed ‘country\_state’ field

lines dataframe: dropped the transport\_mode\_id field because it was mostly blank and its purpose unknown

station\_lines dataframe:

* extracted the longitude and latitude values from the ‘geometry’ field to create new Lon and Lat fields, then dropped geometry field
* ‘created\_at’ and ‘update\_at fields’ were kept, though we were not sure of the purpose of those fields

tracks dataframe and stations dataframe:

* parsed ‘geometry’ field into new Lon and Lat fields for a single geo point, then dropped geometry field, but would do differently next time, since we later ascertained that geometry field in this table contained much more data than for just one geo point
* ‘buildstart’ and ‘opening’ fields: where there was not a valid year, original data had a mishmash of alternate values to indicate; any rows contain values other than the range of 1750 through 2100 were changed to ‘9999’

systems dataframe: – many ‘name’ values were blank, those were changed to “no name”

track\_lines dataframe: – no changes; the fields ‘created\_at’ and ‘update\_at’ were considered for deletion, but because we were not sure whether these values indicated the dates the track lines were created, or when this data was created.

In both the stations and tracks dataframes, inconsistent invalid/blank/null values in the ‘closure’ field were uniformly set to ‘9999’

Created 2 new dataframes from merges:

1. station\_lines and stations were joined into station\_lines\_merge – decided it was of benefit to have fuller data in one table to facilitate later analysis, but original tables still kept
2. track\_lines and tracks were joined into track\_lines\_merge – same rationale

**Data Loading**

Using an SQL Alchemy database engine connection, a Postgres database was created named ‘transit\_systems’:

conn = psycopg2.connect(user='<user\_name>', host='',password='<password>')

conn.set\_isolation\_level(ISOLATION\_LEVEL\_AUTOCOMMIT)

cur = conn.cursor()

cur.execute("CREATE DATABASE transit\_systems")

The 9 new dataframes were then pushed to the database as tables, i.e.:

station\_lines\_df3.to\_sql("station\_lines", engine)

**Problems Encountered**

When first recreating the original data in a Postgres database, code from StackOverflow was used that caused the creation of a database called ‘postgres’ in Kent’s installation. This database could not be dropped and caused all future tables to be created in this ‘postgres’ database despite attempts to create the tables in other newly created databases.

**Future Considerations**

We could not be sure that despite limiting our data to the 81 cities, that all of the remaining data would link together with integrity, so that is a project for the future.