ETL: Extract, Transform, Load

NYC Department of Education

NYC OpenData

About the Data

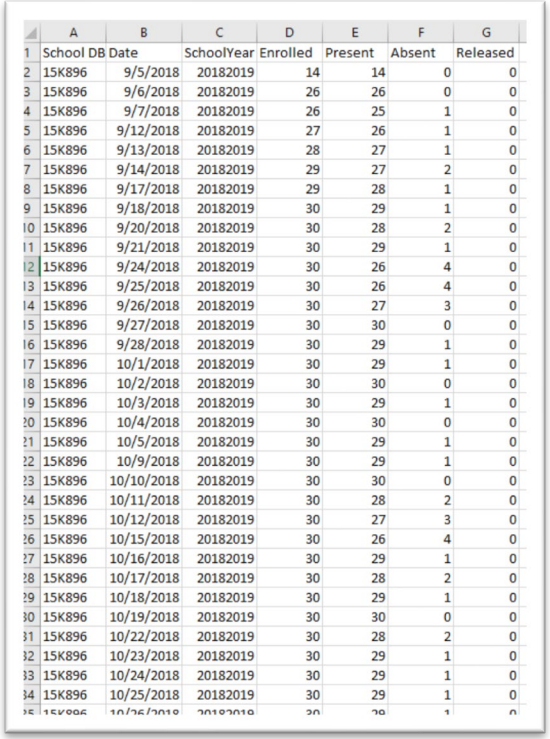
For our Week 13 ETL assignment we chose to extract, transform and load year-over-year NYC public school attendance data. The data, provided by NYC's Department of Education, was sourced through the NYC Open Data API. The raw dataset includes School DBN, Date and School Year; as well as the daily number of Enrolled, Present, Absent and Released students for the 2018-19 and 2019-20 school years.

The second dataset provided school names and grade levels ('K-8,' 'K-12,' etc.) corresponding to the DBN codes. This gave us a bit more context around the numbers we were working with.

Extract

While the original dataset is in JSON format, calling the API directly limits the data returned to just 1,000 rows.

Instead we downloaded the entire dataset in CSV format from the NYC Open Data website.



	A	B	C	D	E	F	G
1	School DB	Date	SchoolYear	Enrolled	Present	Absent	Released
2	15K896	9/5/2018	20182019	14	14	0	0
3	15K896	9/6/2018	20182019	26	26	0	0
4	15K896	9/7/2018	20182019	26	25	1	0
5	15K896	9/12/2018	20182019	27	26	1	0
6	15K896	9/13/2018	20182019	28	27	1	0
7	15K896	9/14/2018	20182019	29	27	2	0
8	15K896	9/17/2018	20182019	29	28	1	0
9	15K896	9/18/2018	20182019	30	29	1	0
10	15K896	9/20/2018	20182019	30	28	2	0
11	15K896	9/21/2018	20182019	30	29	1	0
12	15K896	9/24/2018	20182019	30	26	4	0
13	15K896	9/25/2018	20182019	30	26	4	0
14	15K896	9/26/2018	20182019	30	27	3	0
15	15K896	9/27/2018	20182019	30	30	0	0
16	15K896	9/28/2018	20182019	30	29	1	0
17	15K896	10/1/2018	20182019	30	29	1	0
18	15K896	10/2/2018	20182019	30	30	0	0
19	15K896	10/3/2018	20182019	30	29	1	0
20	15K896	10/4/2018	20182019	30	30	0	0
21	15K896	10/5/2018	20182019	30	29	1	0
22	15K896	10/9/2018	20182019	30	29	1	0
23	15K896	10/10/2018	20182019	30	30	0	0
24	15K896	10/11/2018	20182019	30	28	2	0
25	15K896	10/12/2018	20182019	30	27	3	0
26	15K896	10/15/2018	20182019	30	26	4	0
27	15K896	10/16/2018	20182019	30	29	1	0
28	15K896	10/17/2018	20182019	30	28	2	0
29	15K896	10/18/2018	20182019	30	29	1	0
30	15K896	10/19/2018	20182019	30	30	0	0
31	15K896	10/22/2018	20182019	30	28	2	0
32	15K896	10/23/2018	20182019	30	29	1	0
33	15K896	10/24/2018	20182019	30	29	1	0
34	15K896	10/25/2018	20182019	30	29	1	0

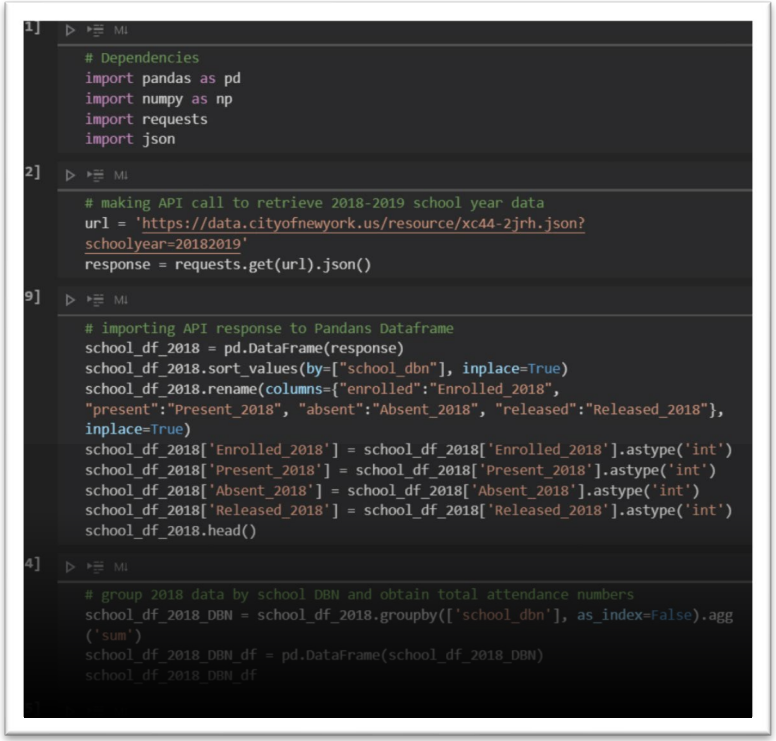

```
[
  {
    "school_dbn": "15K896",
    "date": "09/05/2018",
    "schoolyear": "20182019",
    "enrolled": "14",
    "present": "14",
    "absent": "0",
    "released": "0"
  },
  {
    "school_dbn": "15K896",
    "date": "09/06/2018",
    "schoolyear": "20182019",
    "enrolled": "26",
    "present": "26",
    "absent": "0",
    "released": "0"
  },
  {
    "school_dbn": "15K896",
    "date": "09/07/2018",
    "schoolyear": "20182019",
    "enrolled": "26",
    "present": "25",
    "absent": "1",
    "released": "0"
```

Transform

With the dataset fully accessible (450K+ rows) we can clean, group and merge the data for easier loading and analysis.

Specific transformations:

- Converting numeric data types from text to integers *(i.e. Enrolled, Present, Absent and Released)*.
- Creating separate pd.DataFrames by school year for comparison
- Grouping attendance data by school
- Merging both school year datasets on the school_dbn value.
- Merging the new consolidated table with the ‘school_name’ and ‘school_level’ dataset.



```
[1]
# Dependencies
import pandas as pd
import numpy as np
import requests
import json

[2]
# making API call to retrieve 2018-2019 school year data
url = 'https://data.cityofnewyork.us/resource/xc44-2jrh.json?
schoolyear=20182019'
response = requests.get(url).json()

[9]
# importing API response to Pandans Dataframe
school_df_2018 = pd.DataFrame(response)
school_df_2018.sort_values(by=["school_dbn"], inplace=True)
school_df_2018.rename(columns={"enrolled":"Enrolled_2018",
"present":"Present_2018", "absent":"Absent_2018", "released":"Released_2018"},
inplace=True)
school_df_2018['Enrolled_2018'] = school_df_2018['Enrolled_2018'].astype('int')
school_df_2018['Present_2018'] = school_df_2018['Present_2018'].astype('int')
school_df_2018['Absent_2018'] = school_df_2018['Absent_2018'].astype('int')
school_df_2018['Released_2018'] = school_df_2018['Released_2018'].astype('int')
school_df_2018.head()

[4]
# group 2018 data by school DBN and obtain total attendance numbers
school_df_2018_DBN = school_df_2018.groupby(['school_dbn'], as_index=False).agg
('sum')
school_df_2018_DBN_df = pd.DataFrame(school_df_2018_DBN)
school_df_2018_DBN_df
```

Load

Given how clean and clearly structed the dataset is, we moved the final DataFrame to a PostgreSQL relational database. This also allows for more mathematical and statistical calculations.

Load sequence:

- Creating a new ‘NYC_School_DB’ database in PGAdmin and a ‘schools’ table to mirror our final pandas Dataframe
- Establishing the database connection inside of our Juptyter Notebook and loading in the data
- Confirming the data was loaded with the “SELECT * FROM schools” command

2	01M019	P.S. 019 Asher Levy	Elementary	44994	40744	3979	271	27491	25248	2242	1
3	01M020	P.S. 020 Anna Silver	Elementary	85674	78870	6804	0	53695	48929	4762	4
4	01M034	P.S. 034 Franklin D. Roo...	K-8	54900	48554	6309	37	33896	29536	4360	0
5	01M063	The STAR Academy - P...	Elementary	40873	37111	3731	31	26659	24134	2494	31

Load

1.

NYC_School_DB/postgres@PostgreSQL 12

Query Editor | Query History

```
1  CREATE TABLE schools (
2      school_dbn VARCHAR PRIMARY KEY,
3      school_name VARCHAR,
4      school_level VARCHAR,
5      enrolled_2018 INTEGER,
6      present_2018 INTEGER,
7      absent_2018 INTEGER,
8      released_2018 INTEGER,
9      enrolled_2019 INTEGER,
10     present_2019 INTEGER,
11     absent_2019 INTEGER,
12     released_2019 INTEGER
13 );
14
15 select * FROM schools;
```

2.

Connect to local database

```
[9]
# setup database connection
db_connection_string = "postgres:postgres@localhost:5432/NYC_School_DB"
engine = create_engine(f'postgresql:///{db_connection_string}')

[10]
# check for tables
engine.table_names()

['schools']

[12]
# use pandas to load data to schools table
complete_school_df.to_sql(name='schools', con=engine, if_exists='append', index=False)

# confirming the data was added to the database
pd.read_sql_query('select * from schools', con=engine).head(5)
```

	school_dbn	school_name	school_level	enrolled_2018	present_2018	absent_2018	released_2018	enrolled_2019	present_2019	absent_2019	released_2019
0	01M015	P.S. 015 Roberto Clemente	Elementary	30908	28609	2278	21	21113	19599	1500	14
1	01M019	P.S. 019 Asher Levy	Elementary	44994	40744	3979	271	27491	25248	2242	1
2	01M020	P.S. 020 Anna Silver	Elementary	85674	78870	6804	0	53695	48929	4762	4
3	01M034	P.S. 034 Franklin D. Roosevelt	K-8	54900	48554	6309	37	33896	29536	4360	0
4	01M063	The STAR Academy - P.S.63	Elementary	40873	37111	3731	31	26659	24134	2494	31

3.

	school_dbn [PK] character varying	school_name character varying	school_level character varying	enrolled_2018 integer	present_2018 integer	absent_2018 integer	released_2018 integer	enrolled_2019 integer	present_2019 integer	absent_2019 integer	released_2019 integer
1	01M015	P.S. 015 Roberto Cleme...	Elementary	30908	28609	2278	21	21113	19599	1500	1
2	01M019	P.S. 019 Asher Levy	Elementary	44994	40744	3979	271	27491	25248	2242	
3	01M020	P.S. 020 Anna Silver	Elementary	85674	78870	6804	0	53695	48929	4762	
4	01M034	P.S. 034 Franklin D. Roo...	K-8	54900	48554	6309	37	33896	29536	4360	
5	01M063	The STAR Academy - P...	Elementary	40873	37111	3731	31	26659	24134	2494	3

Further Analysis

Now that the original data has been extracted, transformed (cleaned & formatted) and loaded into a SQL database, we can easily conduct further analyses such as:

- Attendance by school and/or by school year
- Enrollment by school / school year
- Release rate by school / school year

Additional consideration:

- NYC Open Data has an additional dataset that provides school DBN codes / names as well as city borough, neighborhood, school description, course curriculum, extra curriculars and contact info.
- The information above, and other relevant city datasets, can be layered on for more contextual analysis

Questions?