|  |
| --- |
| ETL Guide: NYC Schools |
| Section 1: Test Score and Demographic Data |

|  |
| --- |
| Version 1.0 | By: James Curtis, Austin Manser |

****

|  |
| --- |
| ETL Pipeline Overview |
| Data Context |
| Background Congratulations on your assignment to the NYC data administration division! This guide will help you perform ETL tasks in support of NYC school data and give you additional context as to why these tasks are important.  The NYC Department of Education collects a myriad of data about all the activity of our students, faculty, and administration. This data supports a wide variety of situational awareness and decision-making, such as informing parents about what happens at our schools, helping our staff monitor the achievement and progress towards improvement goals, and securing federal grant funding. Now that you are helping us keep all this data straight, it’s important to learn how the current process works for our ETL pipelines. Section 1: Performing Extract Procedures We’ll start at the beginning for your new data. Raw CSV and JSON files are available from the NYC OpenData page as part of the city’s effort to be more transparent. For this task, you’ll want to download three files ([SAT data](https://data.cityofnewyork.us/Education/2012-SAT-Results/f9bf-2cp4), [AP data](https://data.cityofnewyork.us/Education/2012-AP-Results/9ct9-prf9/data), and [demographic data](https://data.cityofnewyork.us/Education/2006-2012-School-Demographics-and-Accountability-S/ihfw-zy9j)). For the SAT data and demographic data, download the file as a **CSV**; to do so, click on the “Export” button on the sub-menu bar, top right, and select “CSV”. For the AP data, download it as a **JSON**; to do so, click the “Export” button and right click + download the “JSON” option. The data is supposed to be refreshed annually, but you’ll want to check with the data entry folks for the current release cycle.  After downloading the files, set up your coding environment of choice, and begin importing the data into Pandas. Please see the enclosed sample file for dependencies. You’ll want to import the data into dataframes using the read\_csv module for the two CSV files, and the json module for the JSON file. For our particular JSON, you should specify the “data” area of the JSON when pulling, to avoid a value error upon import. Because of this, you’ll also need to filter and rename some of the columns (if you need help with confirming the correct column names, download the CSV version of this file, as described above for SAT and demographic data. Looking back on historical data, this ordering should be resilient, but it’s worth double-checking. |

|  |
| --- |
| Section 2: Performing Transform Procedures Now for the fun part! We’ll begin the necessary data transformations by filtering the demographic dataframe using double brackets, removing some unnecessary fields. Then, filter the data using a loc indexer to select only the school year needed for this particular dataset (in this example, 2011-2012).  Next, we’re going to insert the school year into the SAT dataframe, which requires gathering and concatenating the two years comprising a school year, based first on the name of the file, and then on user input confirming or correcting the automatic output. This pulls from the name of the SAT file, which is named after the second year of the particular school year’s dataset. If the user confirms the year with a “y” input, the year is applied to the “School Year” column; otherwise, the user is asked to input a school year in format YYYYYYYY (first and second years), which is then applied to the “School Year” column.  Finally, to confirm that there are no duplicates in your DBN (District, Borough, and School Number) columns, such as a fat-finger error, we run an “if else” statement checking that each DBN column, converted to an index, is unique (using “is\_unique”). Section 3: Performing Load Procedures If you’ve made it this far, you’re doing great. Let’s finish strong by using pymysql to create our connection to MySQL via python. Set up your dependencies, check for the existence of pre-existing table names in the sql database, and begin loading the SAT, AP, and demographic data into a MySQL relational database, using the caveat of replacing existing tables if they already exist, and stripping the numerical index from the dataframes. Use SQL queries to create the table shells according to needed columns, using “DBN” as the primary key.  Then, use SQL queries to create useful summary views that can be used in our data visualization programs, like Tableau or Plotly. Some example views used include AP data with and without demographics, SAT data with and without demographic data, AP and SAT data together without demographics, and all data. When you create a view using demographic data, be sure to filter it according to the school year you are presenting. Section 4: Data Caveats to Consider We’re done with the ETL process! The “s” in the test result datasets stands for “suppressed,” and means that 5 or fewer students took a particular test at that school. You could filter out all schools with “s” values across the board, to have a data subset of only schools with greater than 5 students took either an AP or SAT test.  Another caveat: There is missing data in the demographics table for numbers representing each class (Freshman, Sophomore, Junior, Senior). This doesn’t matter much for this particular problem-set, unless you are looking to compare the pool of potential test-takers to the number of students who took a test.  Another small caveat: We don’t necessarily know how many of the test-takers actually attend the school where they took the test. The student count identifies the school they attend at registration for the AP exam, but only if they self-report this information. |