

1. Inspecting the data



Photo by [Jannis Lucas](#) on [Unsplash](#).

Every year, American high school students take SATs, which are standardized tests intended to measure literacy, numeracy, and writing skills. There are three sections - reading, math, and writing, each with a maximum score of 800 points. These tests are extremely important for students and colleges, as they play a pivotal role in the admissions process.

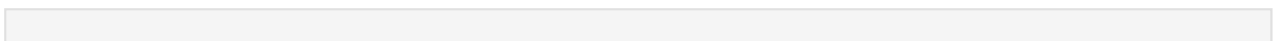
Analyzing the performance of schools is important for a variety of stakeholders, including policy and education professionals, researchers, government, and even parents considering which school their children should attend.

In this notebook, we will take a look at data on SATs across public schools in New York City. Our database contains a single table:

`schools`

column	type	description
<code>school_name</code>	<code>varchar</code>	Name of school
<code>borough</code>	<code>varchar</code>	Borough that the school is located in
<code>building_code</code>	<code>varchar</code>	Code for the building
<code>average_math</code>	<code>int</code>	Average math score for SATs
<code>average_reading</code>	<code>int</code>	Average reading score for SATs
<code>average_writing</code>	<code>int</code>	Average writing score for SATs
<code>percent_tested</code>	<code>numeric</code>	Percentage of students completing SATs

Let's familiarize ourselves with the data by taking a looking at the first few schools!



```
In [2]: %%sql
        postgresql:///schools

        -- Select all columns from the database
        -- Display only the first ten rows

        SELECT *
        FROM schools
        LIMIT 10;
```

10 rows affected.

Out[2]:	school_name	borough	building_code	average_math	average_reading	average_writing	percent_tested
	New Explorations into Science, Technology and Math High School	Manhattan	M022	657	601	601	Non
	Essex Street Academy	Manhattan	M445	395	411	387	78.
	Lower Manhattan Arts Academy	Manhattan	M445	418	428	415	65.
	High School for Dual Language and Asian Studies	Manhattan	M445	613	453	463	95.
	Henry Street School for International Studies	Manhattan	M056	410	406	381	59.
	Bard High School Early College	Manhattan	M097	634	641	639	70.
	Urban Assembly Academy of Government and Law	Manhattan	M445	389	395	381	80.
	Marta Valle High School	Manhattan	M025	438	413	394	35.
	University Neighborhood High School	Manhattan	M446	437	355	352	69.
	New Design High School	Manhattan	M445	381	396	372	73.

```
In [3]: %%nose
        last_output = _
```

```
def test_task1_output_type():
    assert str(type(last_output)) == "<class 'sql.run.ResultSet'", \
        "Please ensure an SQL ResultSet is the output of the code cell."

results = last_output.DataFrame()

def test_task1_results():
    assert results.shape == (10, 7), \
        "The results should have fourteen columns and ten rows."
    assert set(results.columns) == set(['school_name', 'borough', 'building_code', 'ave
    'The results should include all columns from the database, without using an alias.'
    assert last_output.DataFrame().loc[0, 'building_code'] == "M022", \
        "The building code for the first school should be M022."
```

Out[3]: 2/2 tests passed

2. Finding missing values

It looks like the first school in our database had no data in the `percent_tested` column!

Let's identify how many schools have missing data for this column, indicating schools that did not report the percentage of students tested.

To understand whether this missing data problem is widespread in New York, we will also calculate the total number of schools in the database.

In [4]:

```
%%sql

-- Count rows with percent_tested missing and total number of schools
SELECT
    COUNT(*) - COUNT(percent_tested) AS num_tested_missing,
    COUNT(*) AS num_schools
FROM schools;
```

* postgresql:///schools
1 rows affected.

Out[4]:

num_tested_missing	num_schools
20	375

In [5]:

```
%%nose
last_output = _
last_output_df = last_output.DataFrame()

def test_task2_columns():
    assert last_output_df.shape == (1, 2), \
        "Did you correctly select the data? Expected the result to contain one row and two"
    assert set(last_output_df.columns) == set(["num_tested_missing", "num_schools"]), \
        "Did you use the alias `num_tested_missing` and also select the `num_schools` colum

def test_task2_output():
    assert last_output_df.iloc[0, 0] == 20, \
        ""Did you correctly calculate `num_tested_missing`?""
```

```
assert last_output_df.iloc[0, 1] == 375, \
      """Did you correctly calculate the total number of rows in the database?"""
```

Out[5]: 2/2 tests passed

3. Schools by building code

There are 20 schools with missing data for `percent_tested`, which only makes up 5% of all rows in the database.

Now let's turn our attention to how many schools there are. When we displayed the first ten rows of the database, several had the same value in the `building_code` column, suggesting there are multiple schools based in the same location. Let's find out how many unique school locations exist in our database.

```
In [6]: %%sql
-- Count the number of unique building_code values

SELECT COUNT(DISTINCT building_code) num_school_buildings
FROM schools;

* postgresql:///schools
1 rows affected.
```

Out[6]: num_school_buildings

233

```
In [7]: %%nose
last_output = _
last_output_df = last_output.DataFrame()

def test_task3_column_name():
    assert last_output_df.columns.tolist() == ["num_school_buildings"], \
        "Did you use the correct alias for the number of unique school buildings?"

def test_task3_value():
    assert last_output_df.values.tolist() == [[233]], \
        "Did you use the correct method to calculate how many unique school buildings are i
```

Out[7]: 2/2 tests passed

4. Best schools for math

Out of 375 schools, only 233 (62%) have a unique `building_code` !

Now let's start our analysis of school performance. As each school reports individually, we will treat them this way rather than grouping them by `building_code`.

First, let's find all schools with an average math score of at least 80% (out of 800).

In [8]:

```
%%sql

-- Select school and average_math
-- Filter for average_math 640 or higher
-- Display from largest to smallest average_math
```

```
SELECT
    school_name,
    average_math
FROM schools
WHERE average_math >= 640
ORDER BY average_math DESC;
```

```
* postgresql:///schools
10 rows affected.
```

Out[8]:

	school_name	average_math
	Stuyvesant High School	754
	Bronx High School of Science	714
	Staten Island Technical High School	711
	Queens High School for the Sciences at York College	701
	High School for Mathematics, Science, and Engineering at City College	683
	Brooklyn Technical High School	682
	Townsend Harris High School	680
	High School of American Studies at Lehman College	669
	New Explorations into Science, Technology and Math High School	657
	Eleanor Roosevelt High School	641

In [9]:

```
%%nose

last_output = _
last_output_df = last_output.DataFrame()

def test_task4_columns():
    assert set(last_output_df.columns) == set(["school_name", "average_math"]), \
        "Did you select the correct columns?"

def test_task4_filter():
    assert last_output_df["average_math"].min() >= 640, \
        ""Did you correctly filter for "average_math" scores more than or equal to 640?""
    assert last_output_df.shape == (10, 2), \
        ""The output has the wrong number of results, did you correctly filter the "averag

def test_task4_values():
    assert last_output_df.iloc[0,0] == "Stuyvesant High School", \
        ""Did you run the correct query? Expected the first school to be "Stuyvesant High
    assert last_output_df.iloc[0,1] == 754.0, \
        ""Did you correctly sort the values by "average_math" in descending order? Expecte
```

Out[9]: 3/3 tests passed

5. Lowest reading score

Wow, there are only ten public schools in New York City with an average math score of at least 640!

Now let's look at the other end of the spectrum and find the single lowest score for reading. We will only select the score, not the school, to avoid naming and shaming!

```
In [10]: %%sql
-- Find lowest average_reading

SELECT MIN(average_reading) AS lowest_reading
FROM schools;

* postgresql:///schools
1 rows affected.
```

```
Out[10]: lowest_reading
-----
          302
```

```
In [11]: %%nose
last_output = _
last_output_df = last_output.DataFrame()

def test_task5_value():
    assert last_output_df["lowest_reading"].values.tolist() == [302.0], \
        ""Did you select the minimum value for the "average_reading" column?""

def test_task5_alias():
    assert last_output_df.columns.tolist() == ["lowest_reading"], \
        ""Did you use the correct alias? Expected "lowest_reading".""
```

Out[11]: 2/2 tests passed

6. Best writing school

The lowest average score for reading across schools in New York City is less than 40% of the total available points!

Now let's find the school with the highest average writing score.

```
In [12]: %%sql

-- Find the top score for average_writing
-- Group the results by school
-- Sort by max_writing in descending order
-- Reduce output to one school

SELECT
```

```

    school_name,
    MAX(average_writing) AS max_writing
FROM schools
GROUP BY school_name
ORDER BY max_writing DESC
LIMIT 1;

```

```

* postgresql:///schools
1 rows affected.

```

```

Out[12]:
      school_name  max_writing
-----
Stuyvesant High School      693

```

```

In [13]: %%nose
last_output = _
last_output_df = last_output.DataFrame()

def test_task6_columns():
    assert set(last_output_df.columns) == set(["school_name", "max_writing"]), \
        ""Did you select "average_writing" and use an alias?""

def test_task6_shape():
    assert last_output_df.shape[0] == 1, \
        "Did you select the correct number of values? Expected one row."

def test_task6_values():
    assert last_output_df.values.tolist() == [['Stuyvesant High School', 693.0]], \
        ""Did you select the maximum value for "average_writing"? Expected a different val

```

```

Out[13]: 3/3 tests passed

```

7. Top 10 schools

An average writing score of 693 is pretty impressive!

This top writing score was at the same school that got the top math score, Stuyvesant High School. Stuyvesant is widely known as a perennial top school in New York.

What other schools are also excellent across the board? Let's look at scores across reading, writing, and math to find out.

```

In [14]: %%sql

-- Calculate average_sat
-- Group by school_name
-- Sort by average_sat in descending order
-- Display the top ten results

SELECT
    school_name,
    average_math + average_reading + average_writing AS average_sat
FROM schools
GROUP BY school_name
ORDER BY average_sat DESC

```

```
LIMIT 10
```

```
;
```

```
* postgresql:///schools  
10 rows affected.
```

Out[14]:

	school_name	average_sat
	Stuyvesant High School	2144
	Staten Island Technical High School	2041
	Bronx High School of Science	2041
	High School of American Studies at Lehman College	2013
	Townsend Harris High School	1981
	Queens High School for the Sciences at York College	1947
	Bard High School Early College	1914
	Brooklyn Technical High School	1896
	Eleanor Roosevelt High School	1889
	High School for Mathematics, Science, and Engineering at City College	1889

In [15]:

```
%%nose  
last_output = _  
last_output_df = last_output.DataFrame()  
  
def test_task7_columns():  
    assert set(last_output_df.columns) == set(["school_name", "average_sat"]), \  
        """"Did you select the correct columns and use an alias for the sum of the three sat  
  
def test_task7_shape():  
    assert last_output_df.shape[0] == 10, \  
        "Did you limit the number of results to ten?"  
    assert last_output_df.shape[1] == 2, \  
        """"Expected your query to return two columns: "school_name" and "average_sat"."""  
  
def test_task7_values():  
    assert last_output_df.iloc[0].values.tolist() == ['Stuyvesant High School', 2144], \  
        """"Did you correctly define your query? Expected different values for the first sch  
    assert last_output_df["average_sat"].min() == 1889, \  
        """"Did you correctly filter the results? Expected a different lowest score for "ave  
    assert last_output_df["average_sat"].max() == 2144, \  
        """"Did you correctly calculate the "average_sat" column? Expected a different top s
```

Out[15]: 3/3 tests passed

8. Ranking boroughs

There are four schools with average SAT scores of over 2000! Now let's analyze performance by New York City borough.

We will build a query that calculates the number of schools and the average SAT score per borough!


```
In [16]: %%sql
-- Select borough and a count of all schools, aliased as num_schools
-- Calculate the sum of average_math, average_reading, and average_writing, divided by
-- Organize results by borough
-- Display by average_borough_sat in descending order

SELECT
    borough,
    COUNT(*) AS num_schools,
    SUM(average_math + average_reading + average_writing) / COUNT(*) AS average_borough_sat
FROM schools
GROUP BY borough
ORDER BY average_borough_sat DESC;

* postgresql:///schools
5 rows affected.
```

```
Out[16]:
```

borough	num_schools	average_borough_sat
Staten Island	10	1439
Queens	69	1345
Manhattan	89	1340
Brooklyn	109	1230
Bronx	98	1202

```
In [17]: %%nose
last_output = _
last_output_df = last_output.DataFrame()

def test_task8_columns():
    assert set(last_output_df.columns) == set(['borough', 'num_schools', 'average_borough_sat'])
    """Did you select the correct columns and use aliases for the number of schools and average borough SAT score?"""

def test_task8_shape():
    assert last_output_df.shape[0] == 5, \
        "Did you group by the correct column? Expected five rows to be returned: one for each borough."
    assert last_output_df.shape[1] == 3, \
        """Expected your query to return three columns: "borough", "num_schools", and "average_borough_sat"."""

def test_task8_values():
    # Each assert statement checks values per row
    assert last_output_df.iloc[0].values.tolist() == ['Staten Island', 10, 1439], \
        """Did you correctly define your query? Expected different values for Staten Island"""
    assert last_output_df.iloc[1].values.tolist() == ['Queens', 69, 1345], \
        """Did you correctly define your query? Expected different values for Queens."""
    assert last_output_df.iloc[2].values.tolist() == ['Manhattan', 89, 1340], \
        """Did you correctly define your query? Expected different values for Manhattan."""
    assert last_output_df.iloc[3].values.tolist() == ['Brooklyn', 109, 1230], \
        """Did you correctly define your query? Expected different values for Brooklyn."""
    assert last_output_df.iloc[4].values.tolist() == ['Bronx', 98, 1202], \
        """Did you correctly define your query? Expected different values for the Bronx."""
    # Check lowest average_reading score is in the last row
    assert last_output_df.iloc[-1, 0] == 'Bronx', \
        """Did you sort the results by "average_sat" in descending order?"""
```

Out[17]: 3/3 tests passed

9. Brooklyn numbers

It appears that schools in Staten Island, on average, produce higher scores across all three categories. However, there are only 10 schools in Staten Island, compared to an average of 91 schools in the other four boroughs!

For our final query of the database, let's focus on Brooklyn, which has 109 schools. We wish to find the top five schools for math performance.

In [18]:

```
%%sql

-- Select school and average_math
-- Filter for schools in Brooklyn
-- Aggregate on school_name
-- Display results from highest average_math and restrict output to five rows

SELECT
    school_name,
    average_math
FROM schools
WHERE borough = 'Brooklyn'
ORDER BY average_math DESC
LIMIT 5;
```

```
* postgresql:///schools
5 rows affected.
```

Out[18]:

	school_name	average_math
	Brooklyn Technical High School	682
	Brooklyn Latin School	625
	Leon M. Goldstein High School for the Sciences	563
	Millennium Brooklyn High School	553
	Midwood High School	550

In [19]:

```
%%nose

last_output = _
last_output_df = last_output.DataFrame()

def test_task9_columns():
    assert last_output_df.columns.tolist() == ['school_name', 'average_math'], \
        ""Did you select the correct columns? Expected "school_name" and "average_math".""

def test_task9_shape():
    assert last_output_df.shape[0] == 5, \
        "Did you limit the output to 5 rows?"
    assert last_output_df.shape[1] == 2, \
        "Did you select the correct number of columns? Expected two."

def test_task9_school_names():
    assert last_output_df["school_name"].tolist() == ['Brooklyn Technical High School',
```

"Did you correctly filter by borough? Expected a different list of school names."

```
def test_task9_values():
    assert last_output_df["average_math"].max() == 682, \
        ""Did you select the correct values? Expected a maximum value of 682.0 for "averag
    assert last_output_df["average_math"].min() == 550, \
        ""Did you select the correct values? Expected a minimum value of 550.0 for "averag
    assert last_output_df["average_math"].values.tolist() == [682, 625, 563, 553, 550],
        ""Did you sort by "average_math" in descending order? Expected different values.""
```

Out[19]: 4/4 tests passed