Granting Accessibility to Information in the Highschool Selection Process.

Alec Bulkin

Northeastern University, Boston, MA, USA

Abstract

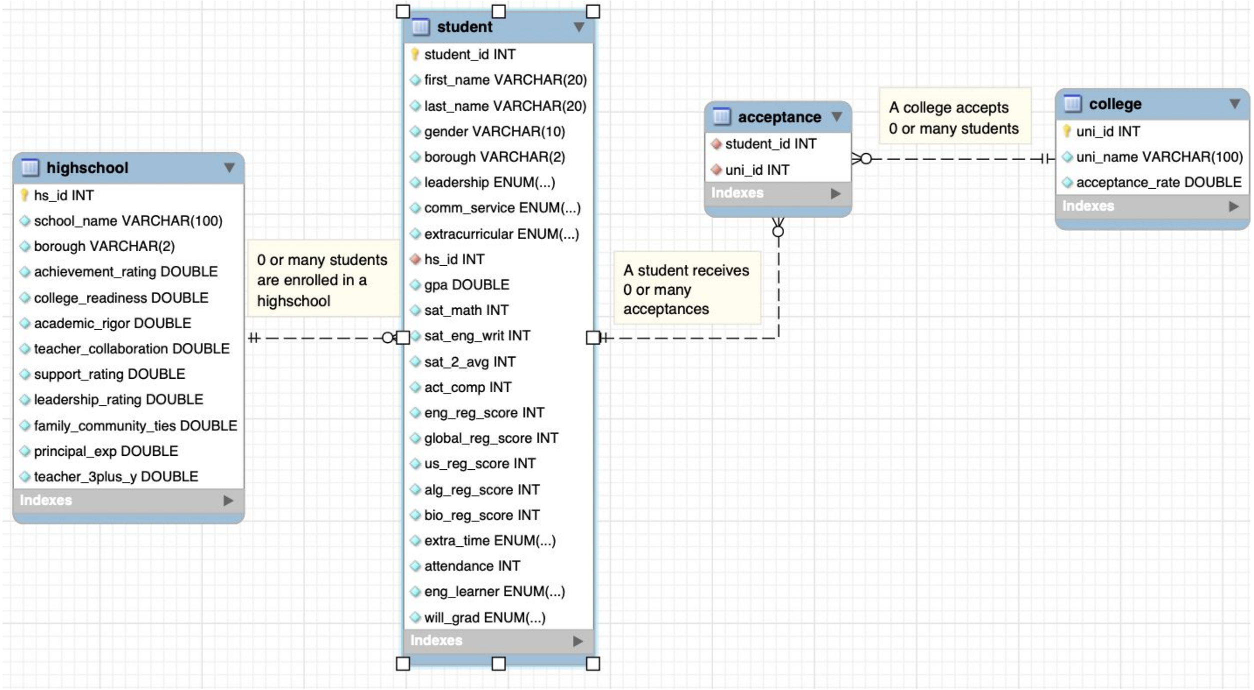
This database is designed to aid parents of rising 8th graders in their search for the perfect high school. It provides them with flexibility and a breadth of accessible data that enables them to effectively filter through the hundreds of public high schools in New York City based on a diverse set of criteria. In this report, the functionality and structure of the highschooldb database will be explored, and examples of its real life application will be provided.

Introduction

New York City is home not only to a rich, diverse culture and history, but to 1.1 million students and their families who participate in the NYC public education system each year. Historically, barely over a couple dozen of the over 500 high schools citywide garner significant attention, mostly as a result of their statistically significant yields to high ranking universities, and above average standardized test scores among their students.

As such, when applying to high school it is often difficult for families to discern where their children would be most likely to fit in and thrive. Thus, the purpose of this database is to help facilitate the high school search process, by enabling families to acquire highly specific information regarding each and every high school in the city, from its ratings to the rate at which its students get accepted to selective colleges.

Database Design



The database consists of four tables, representing highschools, students, acceptances, and colleges.

The highschool table contains the primary key of the highschool, as well as its name and borough, but the primary focus of the data is on the many ratings of the highschools’ inner workings as per the official reviews by the Department of Education of New York City. Each rating is on a scale of 1 to 5, and the specifications for each rating are as follows:

achievement_rating - rates a high school’s student achievement, both academic and extracurricular.

college_readiness - unlike its surrounding column, this represents the proportion of students who are considered well prepared for college.

academic_rigor - rates the degree to which curriculum is sufficiently challenging

teacher_collaboration - rates how well teachers work together

support_rating - rates the extent to which a high school works to provide a supportive environment for students leadership_rating - rates the leadership of a high school’s administration

family_community_ties - rates the relationship a high school has with students’ families as well as with its surrounding communities

The remaining rows represent the number of years of the current principal’s experience at a high school, and the proportion of teachers with 3 or more years of experience. All the ratings and statistics are picked specifically to be useful to families who want to narrow down a search for high schools matching certain minimum standards.

The only major change made to the design of the highschool table was the filling of NULL data with 0s. During the import process, MySQL consistently rejected null values even though the table definition - which can be found in a script attached to this submission - did not have the fields set to not null. While it may seem at first glance that this may be an unorthodox approach to solving the problem, it actually works quite well with regard to maintaining the desired functionality of the database. For example, when a family is searching for a highschool by a table-specific criterion, there is no scenario in which they would filter specifically for a rating or statistic in the highschool table exceeding 0, as this would be redundant based on the range of values. Moreover, when a query would search based on any highschool table field, since the fields will always be greater than 0 the formerly null values will be ignored anyway.

A high school enrolls 0 or many students. For the purposes of the database, all listed students are high school seniors. In the student table, there are at first some descriptions of a student, such as his or her name and what borough he or she is from. However, the bulk of the table consists of boolean values representing a student’s participation (or lack thereof) and role in various extracurricular activities, as well as integers and doubles representing a student’s GPA and test scores on the SAT, ACT, and the New York State exams, also known as Regents. It’s worth noting that some of the aforementioned booleans also represent whether or not a student may have extra test taking time as a result of a learning disability, whether he or she is a learner of the English language, and if he or she will graduate.

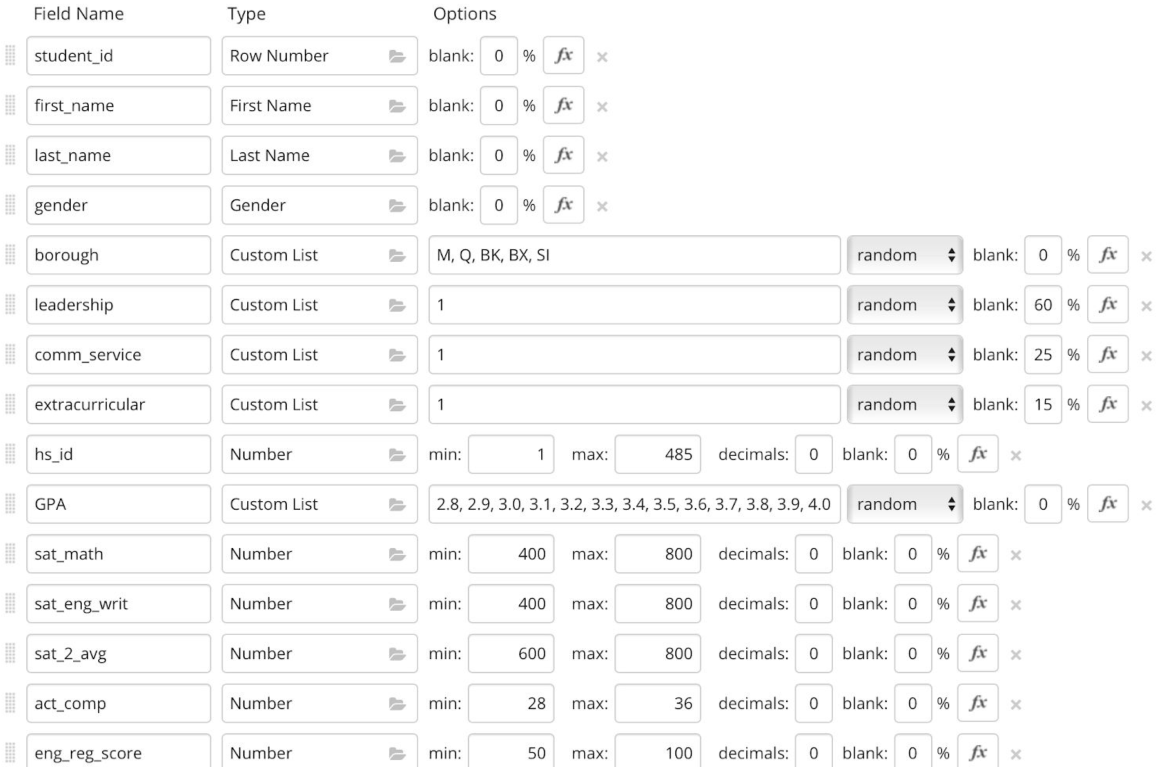
There is also an observed many-to-many relationship between students and colleges, which is indicated in the intermediary acceptance table, which represents offers of admission to a particular university. A student may receive 0 or many offers, while a college may accept 0 or many students. The acceptance table holds only foreign keys from the student and college table. Finally, the college table holds just its primary key, the name of the college, and its acceptance rate, which can be used by families to determine, say, how many students from a particular high school were accepted to universities with a certain minimum acceptance rate. Please note that while acceptance rate isn't perhaps the most accurate representation of how "good" a college is, the database is designed to facilitate a search for highschools, not colleges, and therefore it is unnecessary to provide more detailed descriptive criteria for colleges.

Data Sources and Methods

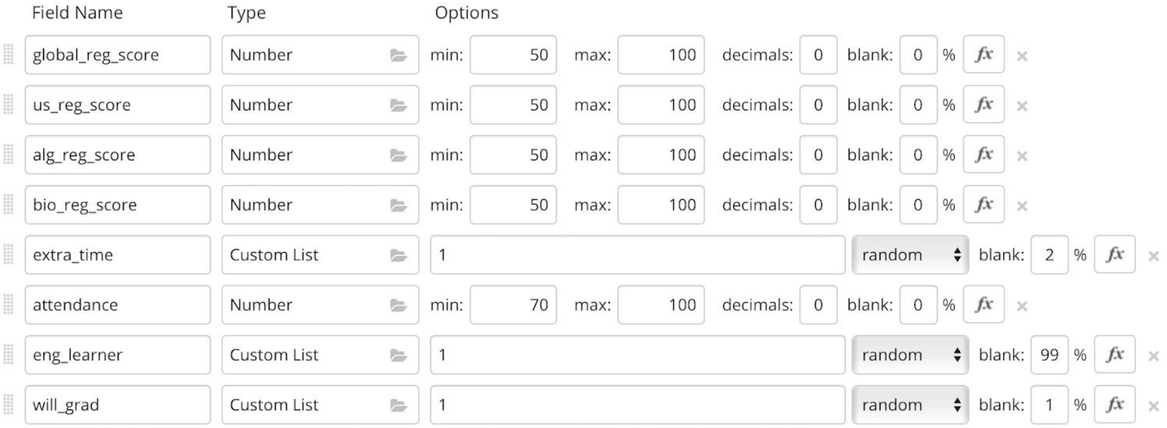
When searching for tables to construct the database, it seemed reasonable to use publicly available data as much as possible, and fill in the "blanks" with randomly generated data. Specifically, the college and highschool tables were both found online, via [1] and [2] respectively. However, in their initial format, the tables contain many columns that aren't altogether necessary for the purposes of the database, so prior to inserting them into the schema the excess columns were all removed. In the case of the highschool table, all percent data was also reformatted into decimal form (to the thousandths place) through the use of the Number Format function in Excel.

The college table proved to be more difficult, as the primary statistic that would be useful for the purposes of the database, the acceptance rate, wasn't present in the original table. Per Professor Rachlin's recommendation, data of 100 top ranking colleges' admission rates [3] was manually inserted into the new acceptance_rate column. This still left some 7050 blank rows, and it seemed reasonable to assume college admission rates were normally distributed. Thus, the remaining blank rows were filled using the Excel formula =NORM.INV(RAND(), 0.5, 0.17) which describes a number derived from a normal distribution with a mean of 0.5 and standard deviation of 0.17. From a statistical standpoint, this means that extremely close to 100% of the data would be contained within the bounds of 0 and 100, and there would be sufficient variation from the mean, .5, to simulate the true distribution of acceptance rates. The column was then searched for occurrences of a proportion that was either negative or exceeded 1, and in rows where this was the case the Excel formula was simply rerun manually, thereby providing a "proper" result. Ultimately, while this is certainly not a perfect estimate of the true statistics, as no public data was found containing all 7150 necessary data points for acceptance_rate, this was a necessary step to make the database capable of effectively demonstrating its functionality.

For the sake of maintaining privacy, no public database containing NYC public high school seniors exists, so the entire table was generated by 15 repeated uses of Mockaroo with a row generation value of 1000, which is the limit for free users. The definition of columns was as follows:



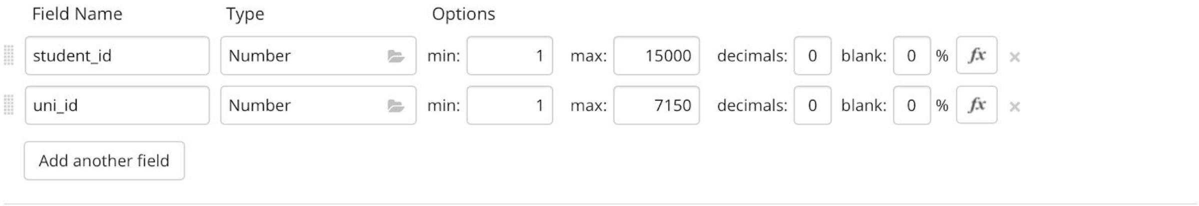
Field Name	Type	Options
student_id	Row Number	blank: 0 % *fx*
first_name	First Name	blank: 0 % *fx*
last_name	Last Name	blank: 0 % *fx*
gender	Gender	blank: 0 % *fx*
borough	Custom List	M, Q, BK, BX, SI · random · blank: 0 % *fx*
leadership	Custom List	1 · random · blank: 60 % *fx*
comm_service	Custom List	1 · random · blank: 25 % *fx*
extracurricular	Custom List	1 · random · blank: 15 % *fx*
hs_id	Number	min: 1 max: 485 decimals: 0 blank: 0 % *fx*
GPA	Custom List	2.8, 2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0 · random · blank: 0 % *fx*
sat_math	Number	min: 400 max: 800 decimals: 0 blank: 0 % *fx*
sat_eng_writ	Number	min: 400 max: 800 decimals: 0 blank: 0 % *fx*
sat_2_avg	Number	min: 600 max: 800 decimals: 0 blank: 0 % *fx*
act_comp	Number	min: 28 max: 36 decimals: 0 blank: 0 % *fx*
eng_reg_score	Number	min: 50 max: 100 decimals: 0 blank: 0 % *fx*



Field Name	Type	Options
global_reg_score	Number	min: 50 max: 100 decimals: 0 blank: 0 % *fx*
us_reg_score	Number	min: 50 max: 100 decimals: 0 blank: 0 % *fx*
alg_reg_score	Number	min: 50 max: 100 decimals: 0 blank: 0 % *fx*
bio_reg_score	Number	min: 50 max: 100 decimals: 0 blank: 0 % *fx*
extra_time	Custom List	1 · random · blank: 2 % *fx*
attendance	Number	min: 70 max: 100 decimals: 0 blank: 0 % *fx*
eng_learner	Custom List	1 · random · blank: 99 % *fx*
will_grad	Custom List	1 · random · blank: 1 % *fx*

The statistics of the student table were generated via simple number generation in an appropriate range. The generation of booleans was slightly more complex, as Mockaroo doesn't have the capability to customize the distribution of true and false values generated by the boolean data type. Thus, columns containing 1s and blank values were generated to simulate an appropriate ratio of true and false values. Ratios were decided based on logical conjecture. For example, it is less likely that a student would have a leadership role and volunteering experience than it is that he or she would be participating in any random extracurricular. Once all 15000 rows of data were generated and appended properly, the columns were modified by copying in the raw, non formula linked values from columns containing the Excel formula =if($COL_LETTER=””,0,1) which replaces all empty spaces from the original column with 0 and while maintaining 1s. This change ensures that the columns can easily be imported into a MySQL schema table as Boolean data types.

Similarly, the acceptance table was also generated through the use of Mockaroo. The generation specifications are as follows:



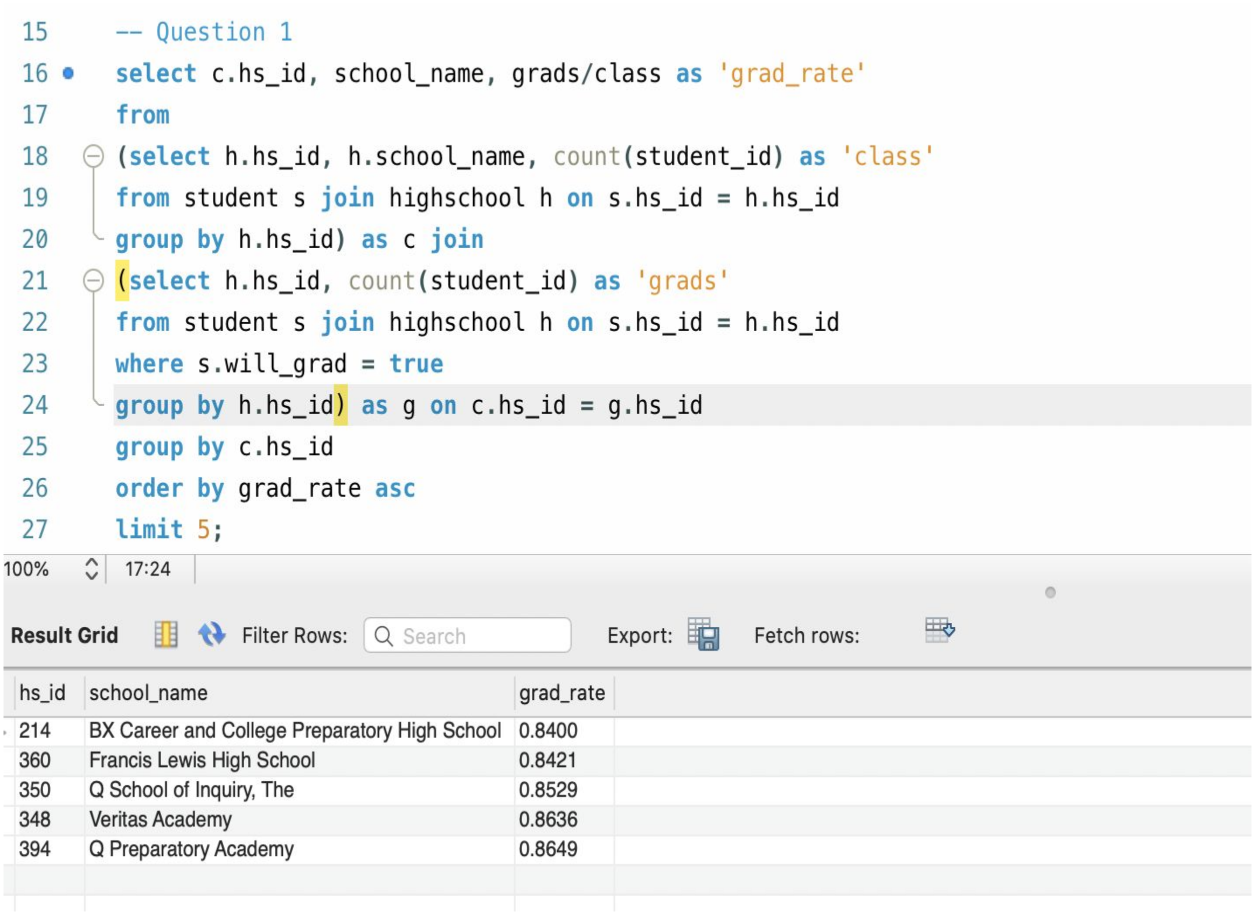
Field Name	Type	Options
student_id	Number	min: 1 max: 15000 decimals: 0 blank: 0 %
uni_id	Number	min: 1 max: 7150 decimals: 0 blank: 0 %

However, while this is less complex than the specification for the student table, it has a noteworthy flaw: Mockaroo can’t account for creating duplicate pairs of foreign keys. In other words, via Mockaroo generation there can be multiple acceptance records of the same student being admitted to the same college, which shouldn’t be the case. Once again, it became necessary to clean the data in Excel. With 39000 rows of data generated, the table was filtered for duplicates, leaving slightly under 39000 rows of distinct pairs of student_id and uni_id.

Finally, with all the tables generated and properly formatted for the purposes of the database, the next step was to import them into the schema, highschooldb, using the MySQL import wizard.

User Cases

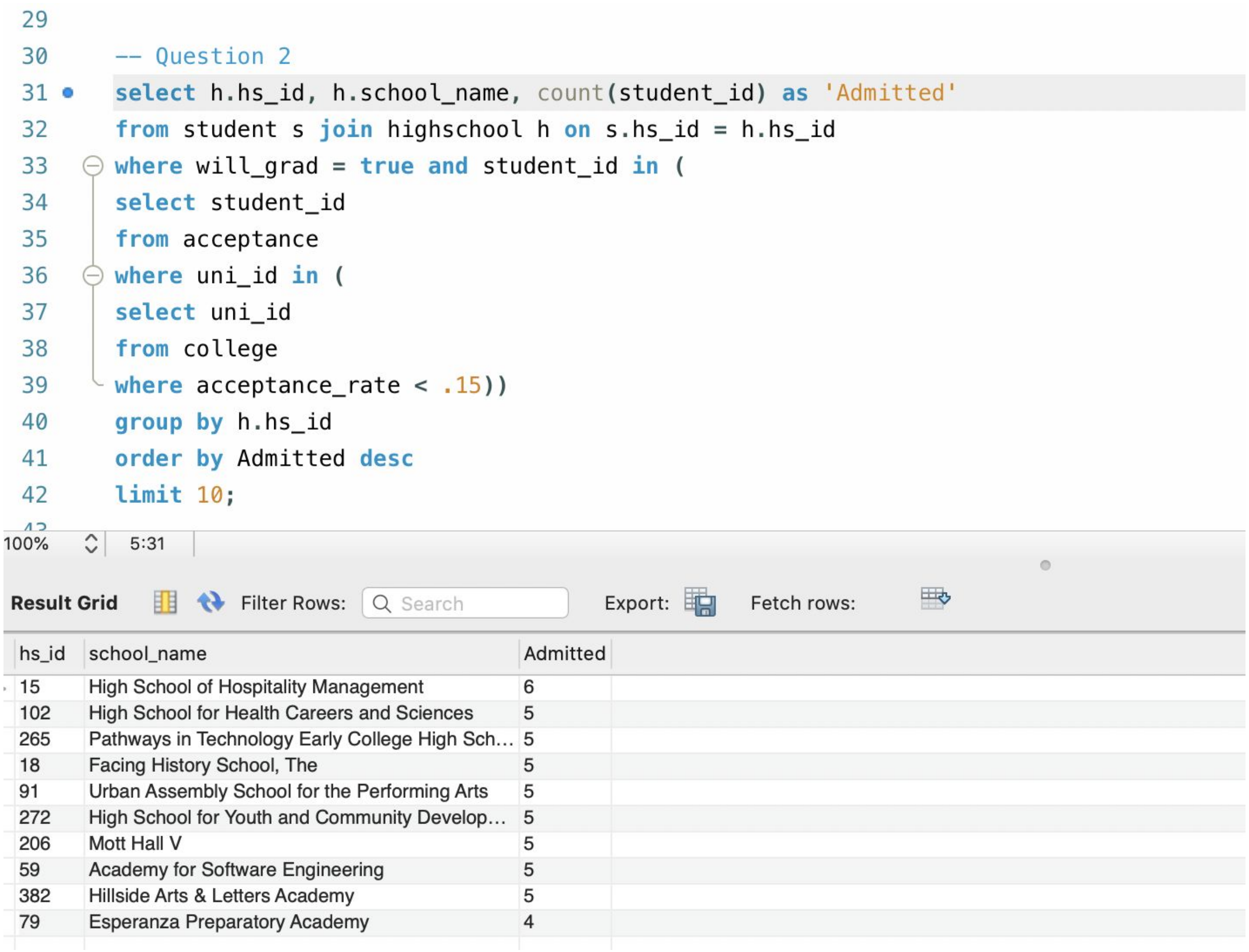
Q1: Say a family wants to know what high schools to steer clear of. Let the question be phrased as follows: which schools have the lowest graduation rate? Return hs_id, school_name, and graduation rate, limit to the five lowest graduation rates.



```
-- Question 1
select c.hs_id, school_name, grads/class as 'grad_rate'
from
(select h.hs_id, h.school_name, count(student_id) as 'class'
from student s join highschool h on s.hs_id = h.hs_id
group by h.hs_id) as c join
(select h.hs_id, count(student_id) as 'grads'
from student s join highschool h on s.hs_id = h.hs_id
where s.will_grad = true
group by h.hs_id) as g on c.hs_id = g.hs_id
group by c.hs_id
order by grad_rate asc
limit 5;
```

hs_id	school_name	grad_rate
214	BX Career and College Preparatory High School	0.8400
360	Francis Lewis High School	0.8421
350	Q School of Inquiry, The	0.8529
348	Veritas Academy	0.8636
394	Q Preparatory Academy	0.8649

Q2: Assume a student and his or her want to find a highschool that sends many of its graduates to selective universities. Let the question be phrased as follows: which highschools have the most admissions to colleges with an acceptance rate of 20% or less? Return the hs_id and school_name of the top 5 highschools ordered by count of graduates to selective colleges.

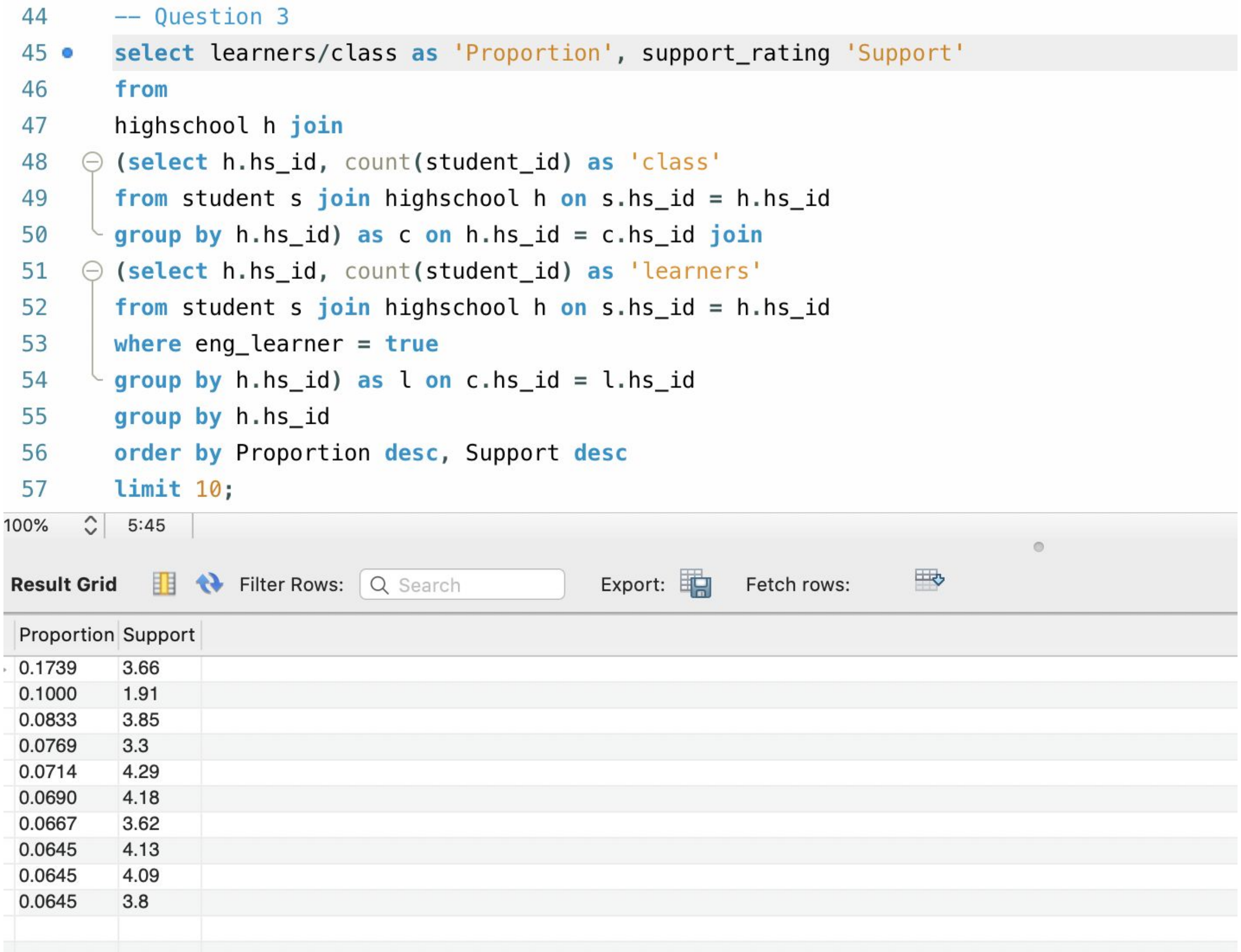


```sql

-- Question 2
select h.hs_id, h.school_name, count(student_id) as 'Admitted'
from student s join highschool h on s.hs_id = h.hs_id
where will_grad = true and student_id in (
select student_id
from acceptance
where uni_id in (
select uni_id
from college
where acceptance_rate < .15))
group by h.hs_id
order by Admitted desc
limit 10;
```

hs_id	school_name	Admitted
15	High School of Hospitality Management	6
102	High School for Health Careers and Sciences	5
265	Pathways in Technology Early College High Sch…	5
18	Facing History School, The	5
91	Urban Assembly School for the Performing Arts	5
272	High School for Youth and Community Develop…	5
206	Mott Hall V	5
59	Academy for Software Engineering	5
382	Hillside Arts & Letters Academy	5
79	Esperanza Preparatory Academy	4

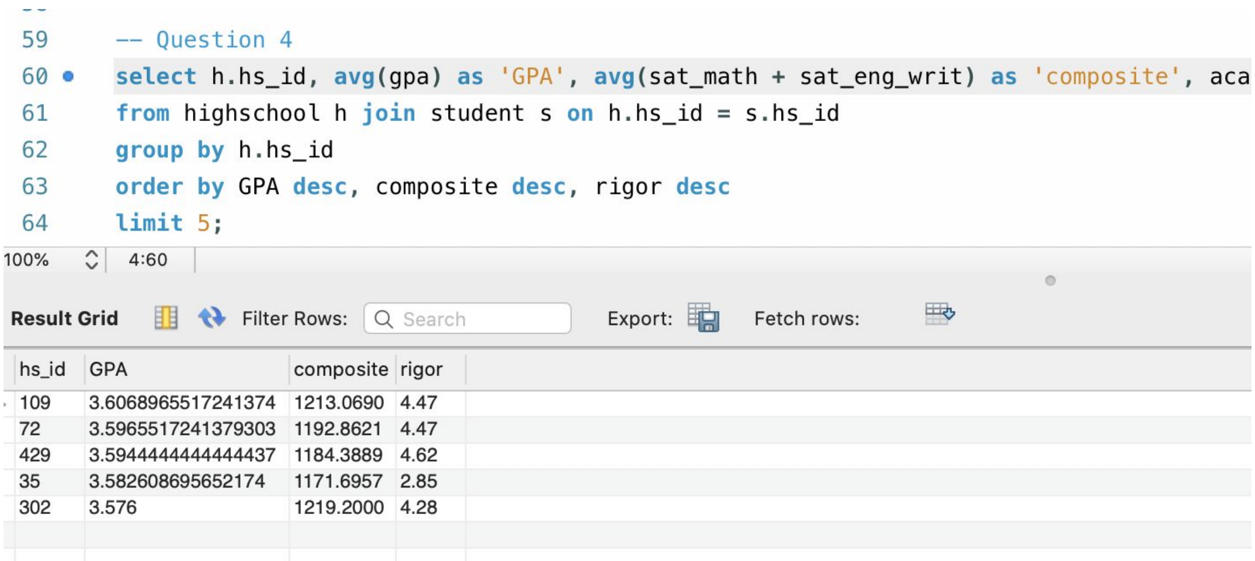
Q3: A family of a student who is an English learner may be interested less in academic rigor and more on ensuring that their student is happy, comfortable, and supported. They'll want to search for highschools that have a high support_rating, and perhaps also a high proportion of other English learners in its student population. Let the question be phrased as follows: which highschools have a high proportion of English learners in their student populations, and also have a support_rating greater or equal to 4.5? Return the hs_id, school_name, proportion of English learners, and support_rating of the top 10 schools, ordered by proportion of English learners and then by support_rating.



```sql
-- Question 3
select learners/class as 'Proportion', support_rating 'Support'
from
highschool h join
(select h.hs_id, count(student_id) as 'class'
from student s join highschool h on s.hs_id = h.hs_id
group by h.hs_id) as c on h.hs_id = c.hs_id join
(select h.hs_id, count(student_id) as 'learners'
from student s join highschool h on s.hs_id = h.hs_id
where eng_learner = true
group by h.hs_id) as l on c.hs_id = l.hs_id
group by h.hs_id
order by Proportion desc, Support desc
limit 10;
```

Proportion	Support
0.1739	3.66
0.1000	1.91
0.0833	3.85
0.0769	3.3
0.0714	4.29
0.0690	4.18
0.0667	3.62
0.0645	4.13
0.0645	4.09
0.0645	3.8

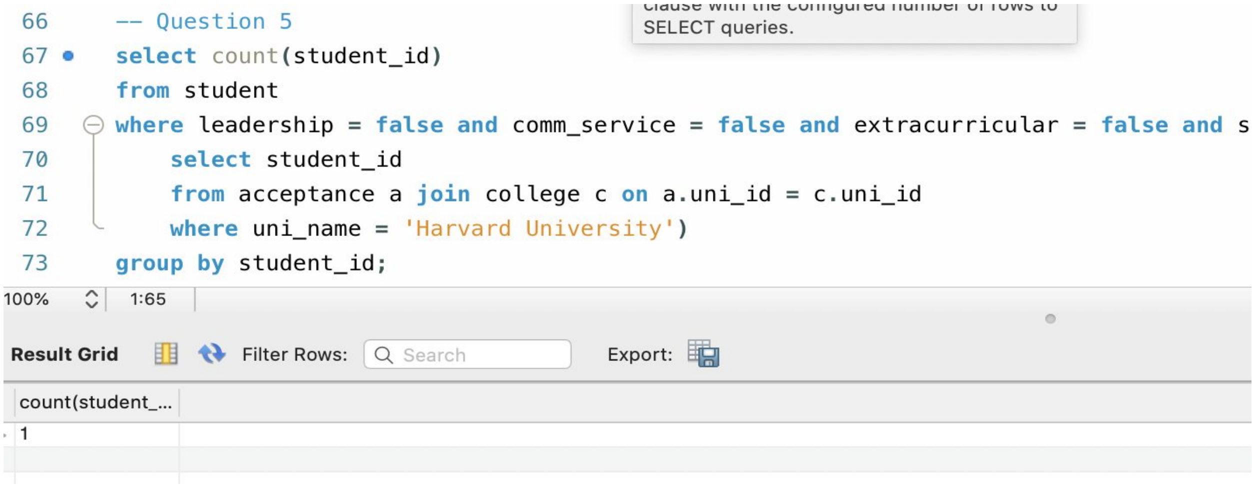
Q4: Another family may be interested specifically in the highschool with the highest academic_rigor rating, average GPA, and average composite SAT score. Let the question be phrased as follows: which highschool has the highest academic_rigor rating, average GPA, and average composite SAT score? Return hs_id, school_name, academic_rigor, average GPA, and average composite SAT score of the top 5 schools, ordered by average GPA, average composite score, and academic rigor rating, in that order.



```
-- Question 4
select h.hs_id, avg(gpa) as 'GPA', avg(sat_math + sat_eng_writ) as 'composite', aca
from highschool h join student s on h.hs_id = s.hs_id
group by h.hs_id
order by GPA desc, composite desc, rigor desc
limit 5;
```

hs_id	GPA	composite	rigor
109	3.6068965517241374	1213.0690	4.47
72	3.5965517241379303	1192.8621	4.47
429	3.5944444444444437	1184.3889	4.62
35	3.582608695652174	1171.6957	2.85
302	3.576	1219.2000	4.28

Q5: Suppose an 8th grade student is incredibly smart but doesn't like participating in any activities and doesn't have extra time on tests. The student's parents want to know how many students like their child got into Harvard University. Let the question be phrased as follows: how many students with no extracurricular participation of any kind and no extra time on tests get into Harvard



```
-- Question 5
select count(student_id)
from student
where leadership = false and comm_service = false and extracurricular = false and s
    select student_id
    from acceptance a join college c on a.uni_id = c.uni_id
    where uni_name = 'Harvard University')
group by student_id;
```

count(student_...
1

Conclusions

In a perfect world, families would have easy access to all the information they need to make educated decisions about their childrens' academic futures. Seeing as how reality doesn't reflect this ideal, this database was constructed with the goal of enhancing accessibility, as well as aiding comparison and analysis of data pertaining to the education quality of highschools. This database defines a multitude of ways to answer a variety of questions families may have about NYC's public high schools, ranging from something as simple as checking average GPA to seeing how various factors of a highschool's inner workings may come together to shape a student's admission process to universities. While limited by a lack of public data regarding student profiles and detailed admission statistics, the highschooldb database nonetheless maintains an advanced degree of flexibility and effectiveness that can be applied and expanded to work from the beginning to the end of a student's academic career.

References

1. “Colleges and Universities.” *HIFLD Open Data*, Homeland Infrastructure Foundation-Level Data (HIFLD), 30 Apr. 2019, hifld-geoplatform.opendata.arcgis.com/datasets/colleges-and-universities.
2. Education, NYC Department of. “2016 - 2017 School Quality Report Results for High Schools: NYC Open Data.” *2016 - 2017 School Quality Report Results for High Schools | NYC Open Data*, NYC OpenData, 2 May 2019, data.cityofnewyork.us/Education/2016-2017-School-Quality-Report-Results-for-High-S/ewhs-k7um.
3. *Top 100 College with Lowest Acceptance Rates (Updated 2020)*, Education Corner, www.educationcorner.com/colleges-with-lowest-acceptance-rates.html.