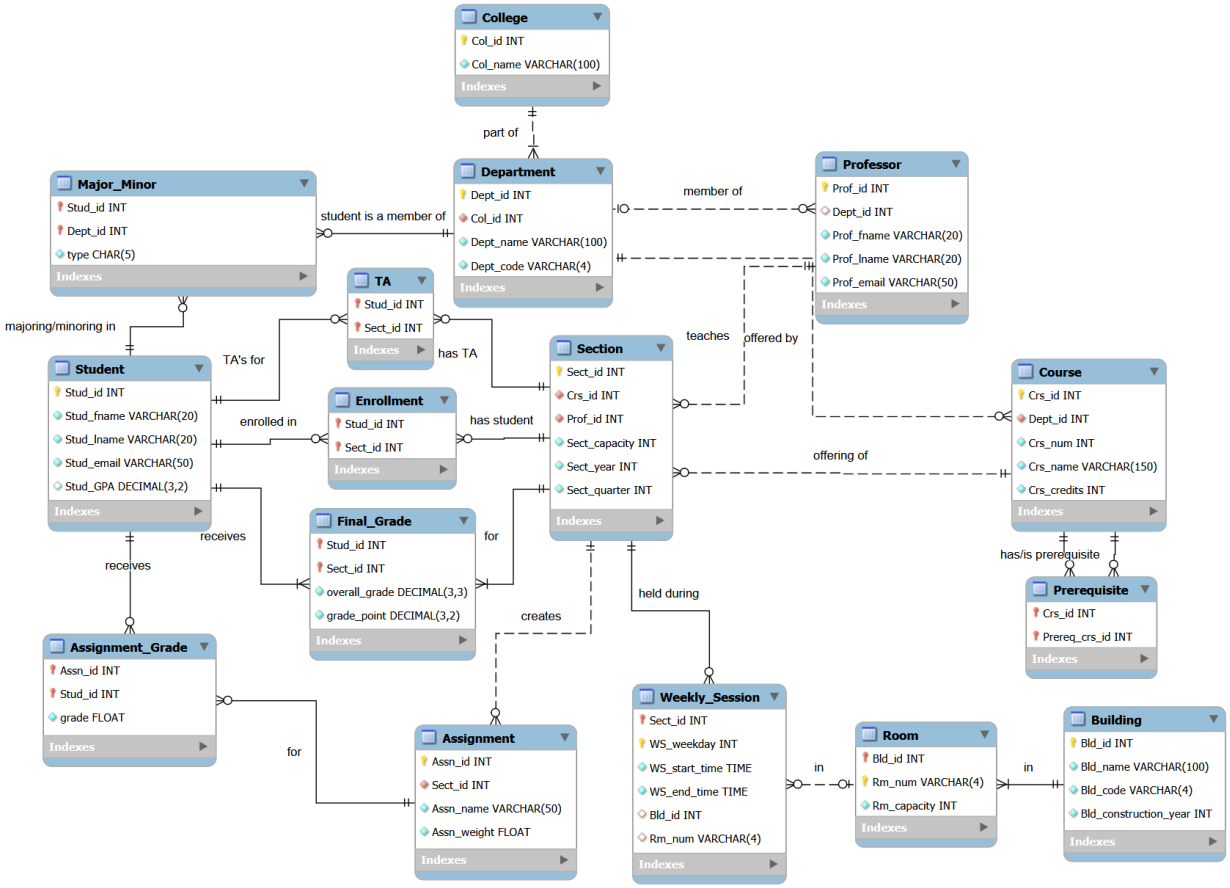
CPSC 5071 Project Milestone 2 (Group 5)

Internal Model (Updated)



The model is, for the part, the same as in Milestone 1, with the majority of changes being to column types (in particular, VARCHAR lengths). For example, we changed Rm_num to be a VARCHAR because our data includes “room numbers” with letters in them. Notably, we dropped the Assn_due_date column for more convenient data generation and added the overall_grade column in the Final_Grade table.

Stored Procedures

GetTranscript

Description: The GetTranscript procedures takes an id (unsigned integer) as input and returns the transcript of the student with that id (or returns an error message if no such student exists). A student may want to use this procedure when planning for registration to see all the course they've taken.

Input:

- id: an unsigned integer

Output:

- If there is a student with the provided id, returns a table containing the course grades of the student. The returned table in sorted in the order the student took the courses and has columns first_name, last_name, year, quarter, department, course_number, course_name, credits, overall_grade, and grade_point.
- If there is no student with the provided id, returns a table with the only column being Message and the only row being "Error: No student found."

Code:

```
DELIMITER //
CREATE PROCEDURE GetTranscript(IN id INT UNSIGNED)
BEGIN
    DECLARE Stud_count INT UNSIGNED;
    SELECT COUNT(*) INTO Stud_count FROM Student WHERE Stud_id = id;

    -- if Stud_count > 0, then such a student exists
    IF Stud_count > 0 THEN
        SELECT St.Stud_fname first_name,
               St.Stud_lname last_name,
               Se.Sect_year year, Se.Sect_quarter quarter,
               D.Dept_code department, C.Crs_num course_number, C.Crs_name course_name, C.Crs_credits credits,
               FG.overall_grade, FG.grade_point
        FROM Student St,Final_Grade FG,Section Se,Course C,Department D
        WHERE St.Stud_id = id AND
              St.Stud_id = FG.Stud_id AND
              FG.Sect_id = Se.Sect_id AND
              Se.Crs_id = C.Crs_id AND
              D.Dept_id = C.Dept_id
        ORDER BY Se.Sect_year, MOD(Se.Sect_quarter+2,4);

    -- otherwise, if Stud_count = 0, then no such student exists
    ELSE
        SELECT 'Error: No student found.' AS Message;
    END IF;
END //
DELIMITER ;
```

Example Usage and Output:

```
CALL GetTranscript(3);
```

first_name	last_name	year	quarter	department	course_number	course_name	credits	overall_grade	grade_point
Rudiger	Butler-Bowdon	2023	1	ECON	200	Introduction to Microeconomics	5	0.828	3.02
Rudiger	Butler-Bowdon	2023	1	ECON	216	Economics of Sports	5	0.877	3.42
Rudiger	Butler-Bowdon	2023	1	ECON	235	Introduction to Environmental Economics	5	0.995	4.00
Rudiger	Butler-Bowdon	2023	1	ECON	496	Honors Seminar	5	0.949	3.99
Rudiger	Butler-Bowdon	2024	2	PHYS	106	Facilitated Group Inquiry III	2	1.000	4.00
Rudiger	Butler-Bowdon	2024	2	ECON	282	Using Econometrics: A Practical Approach	5	1.000	4.00
Rudiger	Butler-Bowdon	2024	2	ECON	345	Global Health Economics	5	1.000	4.00
Rudiger	Butler-Bowdon	2024	2	LING	359	Language and Ethnicity	5	0.911	3.69
Rudiger	Butler-Bowdon	2024	3	ECON	201	Introduction to Macroeconomics	5	0.987	4.00
Rudiger	Butler-Bowdon	2024	3	ECON	230	Economics of Fisheries and Oceans	5	1.000	4.00
Rudiger	Butler-Bowdon	2024	3	RE	250	Introduction to Real Estate	3	1.000	4.00
Rudiger	Butler-Bowdon	2024	3	ECON	346	Using Economics to Solve Today's Health…	4	0.992	4.00
Rudiger	Butler-Bowdon	2024	4	PHIL	291	Ethics in Science	5	1.000	4.00
Rudiger	Butler-Bowdon	2024	4	RE	415	Real Estate Law	3	0.892	3.54
Rudiger	Butler-Bowdon	2024	4	ECON	494	Economy of Japan	5	1.000	4.00
Rudiger	Butler-Bowdon	2024	4	ECON	497	Honors Directed Study	5	0.953	4.00
Rudiger	Butler-Bowdon	2024	1	RE	413	Real Estate Finance and Investment	4	0.903	3.62
Rudiger	Butler-Bowdon	2024	1	RE	418	Best Practices in Sustainable Real Estate	3	1.000	4.00
Rudiger	Butler-Bowdon	2024	1	RE	462	Residential Real Estate Data Analytics	5	1.000	4.00
Rudiger	Butler-Bowdon	2024	1	RE	464	Affordable Housing	4	0.989	4.00
Rudiger	Butler-Bowdon	2025	2	RE	397	Introduction to Real Estate Data Modeling	5	1.000	4.00
Rudiger	Butler-Bowdon	2025	2	RE	416	Real Estate Economics and Market Analysis	4	0.949	3.99
Rudiger	Butler-Bowdon	2025	2	PHIL	466	Philosophy of the Social Sciences	5	1.000	4.00
Rudiger	Butler-Bowdon	2025	2	MATH	499	Undergraduate Research	8	0.996	4.00
Rudiger	Butler-Bowdon	2025	3	MATH	100	Algebra	5	1.000	4.00
Rudiger	Butler-Bowdon	2025	3	RE	361	Property Transactions	3	1.000	4.00
Rudiger	Butler-Bowdon	2025	3	RE	411	Real Estate Valuation and Appraisal	3	1.000	4.00
Rudiger	Butler-Bowdon	2025	3	RE	497	Real Estate Data Modeling	4	0.985	4.00
Rudiger	Butler-Bowdon	2025	4	RE	363	Real Estate Development Process	3	0.941	3.93
Rudiger	Butler-Bowdon	2025	4	RE	408	Financial Modeling for Real Estate I	3	0.989	4.00
Rudiger	Butler-Bowdon	2025	4	RE	431	Real Estate Portfolio Management	4	0.978	4.00
Rudiger	Butler-Bowdon	2025	4	RE	459	Risk and Reward in Sustainable Develop…	3	0.997	4.00

GetProfessorSchedule

Description: The GetProfessorSchedule procedures takes an id, year, and section (all unsigned integers) as input and returns a table containing a schedule for all the sections the professor with the provided id will teach that quarter (or returns an error message if no such professor exists). A professor may want to use this procedure when, for example, planning their commute.

Input:

- id: an unsigned integer
- year_: an unsigned integer
- quarter_: an unsigned integer

Output:

- If there is a professor with the provided id, returns a table containing information about the sections they will teach during the specified quarter (Note: if passing in an invalid year/quarter will just result in an empty result set). The returned table is sorted by weekday and then start

time and has columns first_name, last_name, department, course_num, course_name, building, room_num, weekday, start_time, and end_time.

- If there is no professor with the provided id, returns a table with the only column being Message and the only row being “Error: No professor found.”

Code:

```
DELIMITER //
CREATE PROCEDURE GetProfessorSchedule(IN id INT UNSIGNED, IN year_ INT UNSIGNED, IN quarter_ INT UNSIGNED)
BEGIN
    DECLARE Prof_count INT UNSIGNED;
    SELECT COUNT(*) INTO Prof_count FROM Professor WHERE Prof_id = id;

    -- if Prof_count > 0, then such a professor exists
    IF Prof_count > 0 THEN
        SELECT P.Prof_fname first_name, P.Prof_lname last_name,
               D.Dept_code department, C.Crs_num course_num, C.Crs_name course_name,
               B.Bld_name building, WS.Rm_num room_num,
               WS_weekday weekday, WS_start_time start_time, WS_end_time end_time
        FROM Professor P, Section S, Course C, Department D,Weekly_Session WS, Building B
        WHERE P.Prof_id = id AND S.Sect_year = year_ AND S.Sect_quarter = quarter_ AND
              P.Prof_id = S.Prof_id AND
              C.Crs_id = S.Crs_id AND
              D.Dept_id = C.Dept_id AND
              S.Sect_id = WS.Sect_id AND
              B.Bld_id = WS.Bld_id
        ORDER BY WS_weekday,WS_start_time;

    -- otherwise, if Prof_count = 0, then no such professor exists
    ELSE
        SELECT 'Error: No professor found.' AS Message;
    END IF;
END //
DELIMITER ;
```

Example Usage and Output:

```
CALL GetProfessorSchedule(5,2023,1);
```

first_name	last_name	department	course_num	course_name	building	room_num	weekday	start_time	end_time
Idaline	Matushevitz	CSE	160	Data Programming	Mary Gates Hall	389	1	10:00:00	10:50:00
Idaline	Matushevitz	CSE	122	Introduction to Computer Programming II	Mary Gates Hall	389	2	09:00:00	09:50:00
Idaline	Matushevitz	CSE	122	Introduction to Computer Programming II	Mary Gates Hall	389	3	09:00:00	09:50:00
Idaline	Matushevitz	CSE	160	Data Programming	Mary Gates Hall	389	3	10:00:00	10:50:00
Idaline	Matushevitz	CSE	122	Introduction to Computer Programming II	Mary Gates Hall	389	5	09:00:00	09:50:00
Idaline	Matushevitz	CSE	160	Data Programming	Mary Gates Hall	389	5	10:00:00	10:50:00

Queries

How many sections has each professor taught in the 2023-2024 school year?

Relevance: University administrators may want to see which professors are the most active (in terms of teaching) as well as larger trends across departments (e.g., PHIL professors are teaching a lot of courses, which suggests that the demand/accessibility of PHIL courses is higher).

Code:

```
SELECT P.Prof_id id, P.Prof_fname first_name, P.Prof_lname last_name,
       D.Dept_code department,
       COUNT(*) sections_taught
FROM Professor P
LEFT JOIN Section S ON S.Prof_id = P.Prof_id
LEFT JOIN Department D ON P.Dept_id = D.Dept_id
WHERE (S.Sect_year = 2023 AND S.Sect_quarter = 1) OR
      (S.Sect_year = 2024 AND S.Sect_quarter > 1)
GROUP BY P.Prof_id
ORDER BY department DESC, sections_taught DESC;
```

Output: The output contains ~200 rows, so only the first and last few rows are shown.

id	first_name	last_name	department	sections_taught
167	Lucio	Gouldthorpe	RE	6
26	Dasya	Ownsworth	RE	6
185	Ly	Murney	RE	6
188	Eduino	Halston	RE	5
150	Sayre	Tennet	RE	5
141	Darby	Russell	RE	4
50	Nola	Carlon	RE	4
62	Jasun	Tapenden	RE	4
75	Blaire	Haibel	RE	4
92	Jamima	Charette	RE	4
79	Corabella	Greenmon	RE	4
171	Cornall	Aldersea	RE	3
73	Thaddus	Caldaro	RE	3
133	Bailey	Wickson	RE	3
39	Lanie	Johnsey	RE	3
77	Leshia	Bastick	RE	2
61	Tony	Beekmann	RE	2
116	Lambert	Geffen	RE	2
181	Katheryn	Rigglesford	RE	2
88	Sterne	Bispham	RE	2
24	Anabella	Couthard	RE	2
67	Vince	Portam	RE	2
184	Marquita	Hindsberg	RE	2

109	Carine	Evenett	CSE	6
128	Fannie	Szymanzyk	CSE	5
38	Angie	Learned	CSE	5
174	Shanan	Pallant	CSE	5
135	Mitchael	Campkin	CSE	3
104	Beth	Harradine	CSE	3
87	Sheeree	Kastel	CSE	2
194	Glenna	Asey	CSE	2
58	Evy	Letherbury	ARCH	24
179	Dorice	Lermit	ARCH	24
198	Annie	Eble	ARCH	23
7	Ikey	Parrott	ARCH	23
94	Hendrika	Biset	ARCH	22
191	Edin	Phifer	ARCH	22
30	Cass	Hargrove	ARCH	21
144	Salem	Cordelette	ARCH	21
161	Merci	Folomkin	ARCH	20
95	Anna	O'Hartagan	ARCH	20
46	Cindra	Rackley	ARCH	20
172	Delphine	Bithell	ARCH	20
166	Emyle	Streat	ARCH	20
15	Paige	Arstall	ARCH	16

What are the 5 courses with the lowest passing rate?

Relevance: Students may want to see which courses are particularly hard when planning registration so that they don't overload their schedule. Administrators may want to see if students perform systematically worse in certain classes (which might suggest a lack of resources of insufficient preparation in prerequisite courses). (We limit the results to the 5 lowest so that we don't have the same problem as with the previous query.)

Code:

```
SELECT D.Dept_code department, C.Crs_num course_num, C.Crs_name course_name,
       COUNT(DISTINCT FG.Sect_id) num_sections,
       COUNT(FG.Stud_id) num_students,
       -- use 1.0 as passing cut off
       ROUND(SUM(CASE WHEN FG.grade_point >= 1.0 THEN 1 ELSE 0 END)/COUNT(*),2) passing_rate
FROM COURSE C, Section S, Department D, Final_Grade FG
WHERE C.Crs_id = S.Crs_id AND C.Dept_id = D.Dept_id AND S.Sect_id = FG.Sect_id
GROUP BY C.Crs_id
HAVING COUNT(FG.Stud_id) >= 20  -- only courses with more than 20 sample size
ORDER BY passing_rate ASC
LIMIT 5;
```

Output:

department	course_num	course_name	num_sections	num_students	passing_rate
ARCH	301	Introduction to Architectural Design II	5	48	0.83
ARCH	481	3D Modeling and Rendering	12	194	0.92
ECON	497	Honors Directed Study	10	135	0.92
MATH	207	Introduction to Differential Equations	6	186	0.92
ARCH	201	Architectural Design and Representation II	7	207	0.93

Who are the 5 hardest professors?

Relevance: When choosing which sections to take, students might want to know how difficult a professor is (i.e., how well students perform in their sections). Professors might want to see how their statistics compare to other professors to see if they should change up their teaching/grading/assignments/etc.

Code:

```
SELECT P.Prof_id id, P.Prof_fname first_name, P.Prof_lname last_name,
       COUNT(DISTINCT FG.Sect_id) num_sections,
       COUNT(FG.Stud_id) num_students,
       ROUND(AVG(FG.overall_grade),3) average_grade,
       ROUND(AVG(FG.grade_point),2) average_grade_point
FROM Professor P, Section S, Final_Grade FG
WHERE P.Prof_id = S.Prof_id AND S.Sect_id = FG.Sect_id
GROUP BY P.Prof_id
HAVING COUNT(FG.Stud_id) >= 20  -- only courses with at least 20 sample size
ORDER BY average_grade ASC
LIMIT 5;
```

Output:

id	first_name	last_name	num_sections	num_students	average_grade	average_grade_point
97	Lishe	Christian	10	151	0.723	2.19
116	Lambert	Geffen	5	123	0.743	2.33
39	Lanie	Johnsey	8	180	0.772	2.55
154	Willis	Marriner	4	190	0.773	2.55
96	Cleavland	Millam	8	150	0.793	2.71

Which buildings host the most sections (top 10)?

Relevance: Administrators may want to see, for example, which buildings see the most students and, therefore, might most appreciate renovations. On the other hand, it is perhaps the busiest buildings that are the *worst* to renovate if those renovations make many classrooms unable to be used (which would make scheduling harder). They may also want to see which buildings host the most sections because perhaps those are the buildings most likely to be able to fit one more, or vice versa (for section scheduling).

Code:

```
SELECT B.Bld_name building, COUNT(DISTINCT WS.Sect_id) num_sections
FROM Building B, Weekly_Session WS
WHERE B.Bld_id = WS.Bld_id
GROUP BY building
ORDER BY num_sections DESC
LIMIT 10;
```

Output:

building	num_sections
Mary Gates Hall	512
Smith Hall	467
Loew Hall	452
Savery Hall	408
Thomson Hall	328
Mechanical Engineering Building	306
Denny Hall	241
Lee Paul Sieg Building	218
Johnson Hall	198
Kane Hall	182

What are the most popular majors and minors?

Relevance: Administrators may want to which are the biggest departments in order to determine things like resource allocation, future planning, or even university marketing. For example, if they see that the Business department is much larger than every other department, then they might decide to prioritize it in, for example, a new building.

Code:

```
SELECT D.Dept_name, mm.type,
       COUNT(mm.stud_id) AS num_students
FROM Department D, Major_Minor MM
WHERE MM.Dept_id = D.Dept_id
GROUP BY D.dept_name, MM.type
ORDER BY num_students DESC;
```

Output:

Dept_name	type	num_students
Architecture	MAJOR	350
Department of Physics	MAJOR	345
Department of Linguistics	MAJOR	333
Department of Philosophy	MAJOR	332
Paul G. Allen School of Computer Science & Engi...	MAJOR	332
Mechanical Engineering	MAJOR	325
Department of Mathematics	MAJOR	312
Real Estate	MAJOR	300
Department of Economics	MAJOR	295
Paul G. Allen School of Computer Science & Engi...	MINOR	71
Department of Philosophy	MINOR	68
Real Estate	MINOR	68
Department of Economics	MINOR	66
Department of Physics	MINOR	62
Department of Linguistics	MINOR	59
Mechanical Engineering	MINOR	52
Architecture	MINOR	47
Department of Mathematics	MINOR	47

What is the relationship between student grades and TA frequency?

Relevance: Administrators may want to study the relationship between student grades and TA frequency when it comes to things like accessibility. For example, perhaps they want to hire more TA's and the current GPA requirement is too strict. Then they would want to make the hiring process (and student thoughts about who is able to become a TA) more open to students with lower a lower GPA.

Code:

```
WITH TA_counts AS (
    SELECT
        Stud_id,
        COUNT(DISTINCT Sect_id) as times_TAd
    FROM TA
    GROUP BY Stud_id
)
SELECT
    CASE
        WHEN TC.times_TAd > 5 THEN 'Frequent TA (>5 sections)'
        WHEN tc.times_TAd IS NOT NULL THEN 'Occasional TA (1-5 sections)'
        ELSE 'Never TA'
    END TA_frequency,
    COUNT(DISTINCT s.Stud_id) student_count,
    ROUND(AVG(S.Stud_GPA),2) average_GPA,
    ROUND(MIN(s.Stud_GPA),2) minimum_GPA,
    ROUND(MAX(s.Stud_GPA),2) maximum_GPA
FROM Student S
LEFT JOIN TA_counts TC ON S.Stud_id = TC.Stud_id
WHERE S.Stud_GPA IS NOT NULL
GROUP BY TA_frequency
ORDER BY average_GPA DESC;
```

Output:

TA_frequency	student_count	average_GPA	minimum_GPA	maximum_GPA
Frequent TA (>5 sections)	521	3.734	2.24	4.00
Occasional TA (1-5 sections)	2402	3.263	1.08	4.00
Never TA	77	1.801	0.90	3.01

What is the average student GPA by major?

Relevance: Administrators may want to look at average student GPA by major to, for example, assess equitable access to resources (e.g., perhaps one department receives way lower grades because they don't have a reliable internet connection in their buildings).

Code:

```
SELECT d.Dept_name department, AVG(fg.grade_point) AS avg_gpa
FROM Department d
    JOIN Major_Minor mm ON mm.Dept_id = d.Dept_id
    JOIN Student s ON s.Stud_id = mm.Stud_id
    JOIN Final_Grade fg ON fg.Stud_id = s.Stud_id
WHERE mm.type = 'MAJOR'
GROUP BY d.Dept_name
ORDER BY avg_gpa DESC;
```

Output:

department	avg_gpa
Department of Physics	3.398314
Mechanical Engineering	3.318252
Department of Philosophy	3.309590
Department of Economics	3.304278
Department of Mathematics	3.296853
Department of Linguistics	3.290834
Real Estate	3.191725
Paul G. Allen School of Computer Science & Engi…	3.189877
Architecture	3.176690

How does the number of majors/minors relate to their GPA?

Relevance: Administrators can understand the effect of taking multiple majors/minors. They might find, for example, that students who take more than one major receive worse grades, perhaps due to the heavy course load or stress. They would pass this information off to counselors who would then use it to inform how they advise students.

Code:

```
WITH Student_programs AS (
    SELECT
        Stud_id,
        COUNT(DISTINCT Dept_id) total_programs
    FROM Major_Minor
    GROUP BY Stud_id
)
SELECT
    SP.total_programs,
    ROUND(AVG(S.Stud_GPA), 3) as avg_gpa
FROM Student S
JOIN Student_programs SP ON S.Stud_id = SP.Stud_id
WHERE S.Stud_GPA IS NOT NULL
GROUP BY SP.total_programs
ORDER BY total_programs;
```

Output:

total_programs	avg_gpa
1	3.263
2	3.293
3	3.295

How does student performance change across the different quarters?

Relevance: Administrators might find a correlation between the quarters and grades, which might prompt action. For example, seasonal depression may lead to students performing worse in Winter quarter; if administrators see this, they might choose to increase visibility of things like depression call centers/counseling.

Code:

```
SELECT Sect_quarter quarter,
       ROUND(AVG(fg.grade_point),3) as avg_grade_point,
       ROUND(AVG(fg.overall_grade*100), 2) as avg_percentage
FROM Section S, Final_Grade FG
WHERE S.Sect_id = FG.Sect_id
GROUP BY Sect_quarter
ORDER BY Sect_quarter;
```

Output:

quarter	avg_grade_point	avg_percentage
1	3.056	84.35
2	3.385	89.09
3	3.401	89.40
4	3.376	88.94

Is there a relationship between alphabetic order of a student's name and their grades?

Relevance: If assignments are graded in order of last names, then TA's and professors may be more tired/irritable by the time they get to the students with last names later in the alphabet. Administrators/professors would be interested in discovering such a bias so that they can counteract it (e.g., by shuffling the grading order).

Code:

```
SELECT
    UPPER(LEFT(S.Stud_lname, 1)) as last_name_initial,
    COUNT(DISTINCT s.Stud_id) as num_Students,
    ROUND(AVG(s.Stud_GPA), 3) as avg_gpa
FROM Student S
WHERE S.Stud_GPA IS NOT NULL
GROUP BY last_name_initial
ORDER BY avg_gpa DESC;
```

Output:

last_name_initial	num_Students	avg_gpa
J	58	3.544
Q	4	3.510
U	6	3.437
R	116	3.409
N	40	3.400
T	124	3.379
H	146	3.351
C	251	3.342
O	70	3.336
S	280	3.330
K	118	3.317
I	38	3.298
M	254	3.292
W	128	3.290
A	99	3.289
G	209	3.289
F	105	3.276
B	315	3.272
E	67	3.268
D	192	3.267
L	150	3.266
Z	8	3.253
P	166	3.239
V	44	3.201
Y	12	3.001