Final Project: Phase 1 (Submission 2)

Oliver Hancock - ohancock@bu.edu

BUID: U86095543

Fall 2025

MET AD 599 – Python & SQL for Business Analytics

Prof. Sree Valath Bhuan Das

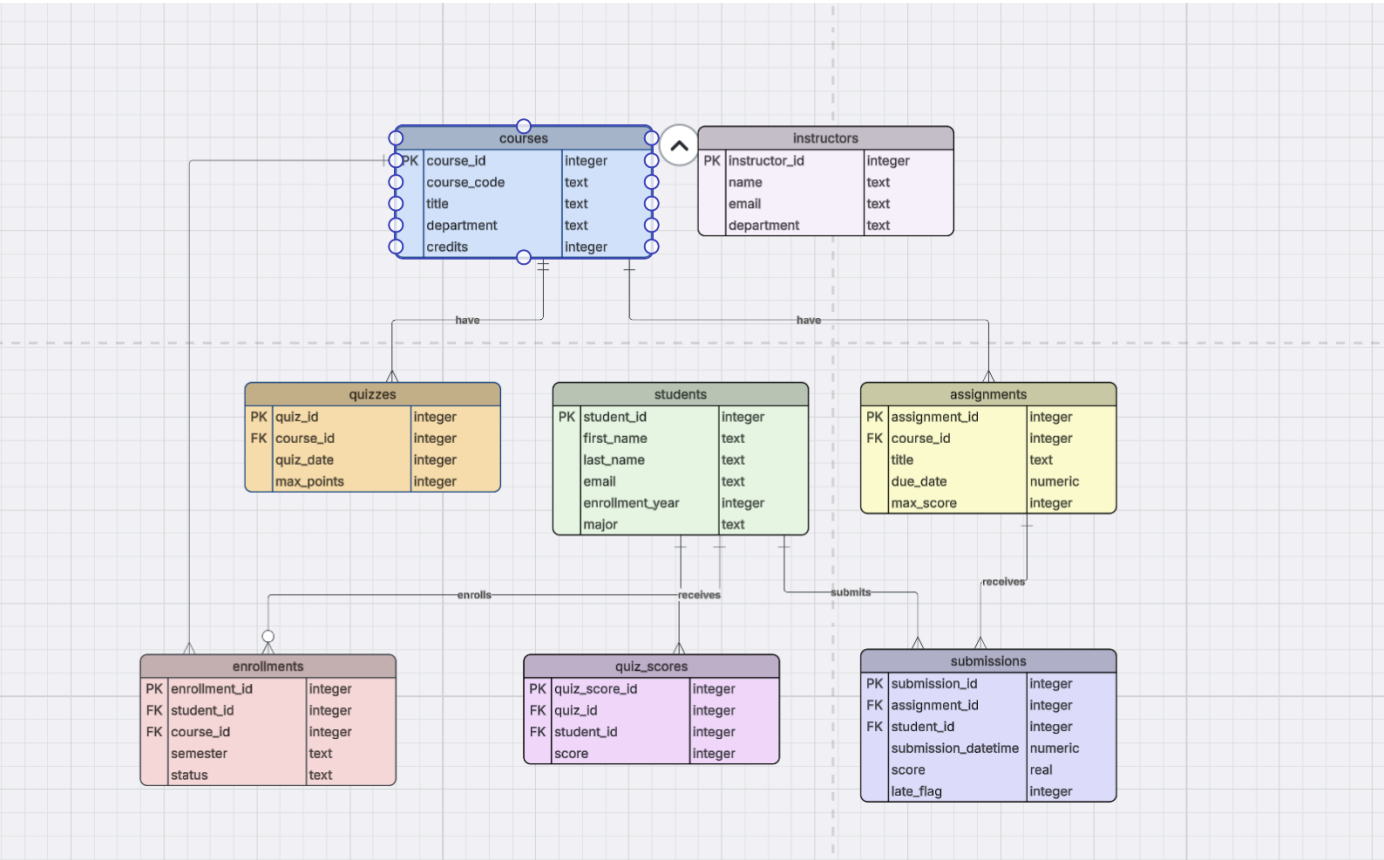
Executive Summary: Phase 1

Phase 1 of this project focuses on becoming fully grounded in the LMS.db file before jumping into heavier analytics. I start by reverse-engineering the ERD, mapping out every table, key, and relationship, and putting together a clean data dictionary and domain overview. This helps me understand the business context and how everything fits together behind the scenes. With that foundation in place, I move into the SQL work: one original analytical question, five standard queries, and two window-function queries. These let me show how the tables connect, how the data aggregates, and what kinds of insights you can pull straight from the database.

By the end of Phase 1, I've built the structure and the early insights needed to support what comes next: machine learning, visualizations, and the final reporting in Phase 2.

Task 1: ERD and Data Understanding

Reverse-Engineered ERD



Business Understanding Summary

One-page description of the domain

The lms.db file represents a university-level Learning Management System with all the core pieces you'd expect: students, instructors, courses, assignments, quizzes, submissions, and grades. It captures the full academic workflow—from students enrolling in courses to completing assessments and having their performance recorded.

Each course includes multiple assessments, and every submission is logged with a score and timestamp. This structure supports accurate tracking of student progress and gives a clear picture of how learning activities unfold across the term. From an analytics standpoint, this type of data is extremely useful. It allows you to look at performance trends, spot engagement issues, identify students who may be falling behind, and evaluate whether assessments are actually aligned with learning goals. Overall, the domain reflects the core data framework needed for academic recordkeeping and informed decision-making in higher education.

Key Business Processes Represented in the Data

The database reflects the core functions of a typical higher-education Learning Management System (LMS). The key business processes supported by the schema include:

<u>Student Information Management</u>

- Creating and maintaining student profiles.
- Tracking academic details such as enrollment year and declared major.

<u>Instructor & Course Administration</u>

- Storing instructor information

Course Enrollment Workflow

- Students selecting and enrolling in courses.
- Maintaining accurate enrollment records each term.

Coursework Delivery and Assessment

- Creating and organizing assignments and quizzes.
- Collecting assignment submissions from students
- Recording quiz and assignment scores.

Academic Performance Tracking

- Associating grades and scores with individual students.
- Monitoring overall academic progress across assessments.

Data Dictionary

The following data dictionary provides a structured overview of all tables, columns, data types, and relationships within the LMS database:

Student:

Column	Data Type	Definition
student_id	INTEGER (PK)	Unique identifier for each student
first_name	TEXT	Student's first name
last_name	TEXT	Student's last name
email	TEXT UNIQUE	Student's institutional email
enrollment_year	INTEGER	Year the student entered the institution
major	TEXT	Student's academic major

Relationship Hints

- Referenced by **submissions**, **quiz_scores**, and **enrollments**
- One student → Many enrollments, submissions, quiz scores

Instructors:

Column	Data Type	Definition
instructor_id	INTEGER (PK)	Unique instructor identifier
first_name	TEXT	Instructor's first name
last_name	TEXT	Instructor's last name
email	TEXT UNIQUE	Instructor email
department	TEXT	Academic department

Relationship Hints

- One instructor > Many courses

Courses:

Column	**Data Type**	**Definition**
course_id	INTEGER (PK)	Unique course identifier
course_name	TEXT	Course title
instructor_id	INTEGER (FK)	Instructor who teaches the course
department	TEXT	Department offering the course
credits	INTEGER	Course credit value

Relationship Hints

- Linked to instructors through instructor_id
- One course > Many assignments, quizzes, enrollments, submissions (indirect through assignments)

Enrollments:

Column	Data Type	Definition
enrollment_id	INTEGER (PK)	Enrollment record ID
student_id	INTEGER (FK)	Student taking the course
course_id	INTEGER (FK)	Course being taken
enrollment_date	DATE	Date student enrolled

Relationship Hints

- Many-to-many resolution between students and courses

Assignments:

Column	Data Type	Definition
assignment_id	INTEGER (PK)	Unique assignment identifier
course_id	INTEGER (FK)	Course for which assignment is created
title	TEXT	Assignment name
description	TEXT	Assignment instructions
due_date	DATE	Date assignment is due

Relationship Hints

- One course > Many assignments
- Assignments > submissions (one-to-many)

Submissions:

Column	Data Type	Definition
submission_id	INTEGER (PK)	Unique submission identifier
assignment_id	INTEGER (FK)	Linked assignment
student_id	INTEGER (FK)	Student submitting

submission_date	DATE	Date submitted
grade	REAL	Numeric grade

Relationship Hints

- One assignment > Many submissions
- One student > Many submissions

Quizzes:

Column	Data Type	Definition
quiz_id	INTEGER (PK)	Unique quiz identifier
course_id	INTEGER (FK)	Associated course
title	TEXT	Quiz title
total_points	REAL	Maximum points

Relationship Hints

- quizzes > quiz_scores (one-to-many)

Quiz_scores:

Column	Data Type	Definition
quiz_score_id	INTEGER (PK)	Score record ID
quiz_id	INTEGER (FK)	Quiz taken
student_id	INTEGER (FK)	Student who took quiz
score	REAL	Score earned

Relationship Hints

- One quiz > Many quiz_scores
- One student > Many quiz_scores

Domain-Specific Terminology:

Term	Meaning in LMS Context
LMS	Learning Management System used for coursework and student activity tracking
Enrollment	A student's registration in a course
Assignment	A graded task students must submit
Submission	A student's completed assignment
Quiz	Timed or graded assessment linked to a course
Score / Grade	Academic performance metrics
Instructor	Faculty responsible for delivering a course
Credits	Academic value representing workload

Propose Business Questions

Based on the data located in the LMS database, a range of analytical business questions can be explored to better understand student performance, departmental activity, and course engagement patterns. These questions leverage the relationships between students, courses, assessments, and enrollment data to uncover trends, identify gaps, and support data-driven decision-making. Below are several business-focused analytical questions derived directly from the data:

1. For each course, what is the total number of enrolled students, and what are the top 5 highest enrolling courses?
2. Which courses have the highest and lowest grade averages?
3. By course, which students have the highest average grade across all their submissions in that course?
4. How many assignments are created for each course?

5. What is the average quiz score by course?
6. For each student, what proportion of assigned coursework (assignments) is missing (no submission), and which students have the highest missing rate?
7. How do average grades across assignments and quizzes differ when grouped by student major?
8. By department, how many quizzes and assignments are associated with their courses, and which departments rely more on quizzes as opposed assignments?
9. By department, what is the total number of enrolled students across all its courses, and how does this compare between departments?
10. By course, what percentage of enrolled students submitted work on or before the due_date, and which assignments have the lowest on-time submission rates

---

Task 2: Analytical SQL Queries

Analytical Question that can be solved by SQL Queries - Own Question

Business Question: What are the average quiz scores for each course, and how do these course-level averages compare to the overall quiz performance within each department?

SQL Query: located in the file, 'TASK2-sub2.sql'

Output screenshot:

department	course_id	course_title	avg_course_quiz_score	avg_department_quiz_score
Business	41	Streamlined mobile neural-net	33.2136991229981	25.92
Business	14	Reverse-engineered incremental webs:	30	25.92
Business	5	Innovative zero-defect concept	29.9075144508671	25.92
Business	24	Programmable eco-centric attitude	27.6558384547849	25.92
Business	27	Optimized zero tolerance model	23.2929782082324	25.92
Business	43	Centralized holistic Local Area Netv	23.2788798133022	25.92
Business	2	Robust even-keeled benchmark	20	25.92
Business	22	Devolved 24hour product	20	25.92
CS	13	Organic asymmetric capacity	33.3614457831325	29.91
CS	15	Profound impactful groupware	33.2913919903654	29.91
CS	4	Stand-alone national attitude	33.093591419776	29.91
CS	34	Devolved didactic benchmark	32.3661064921852	29.91
CS	38	Polarized optimal core	30.1243339253996	29.91
CS	45	Streamlined real-time array	30.0803374541548	29.91
CS	33	Seamless 5thgeneration core	30	29.91

Interpretation of findings:

The results reveal clear differences in quiz performance both across and within departments. In Business, quiz averages show substantial variation, with some courses performing well above the departmental mean and others falling significantly below it. This wide spread suggests uneven assessment difficulty or differing levels of student engagement. In contrast, the CS department shows many more results closer to the average. This consistency indicates more uniform assessment practices or steadier student performances. Comparing individual course averages to their department's highlights where performance aligns with expectations and where deviation occurs. High averages may reflect strong comprehension or accessible assessments, while lower averages may signal areas where students need

additional support or where course content should be reviewed. Overall, the pattern helps identify courses that may benefit from closer attention or targeted instructional adjustments.

Explanation of joins, filters, window logic:

The query uses joins to combine information from three different tables (quiz_scores, quizzes, and courses), so that each quiz score can be connected to the quiz it came from and the course and department it belongs to. These joins are essential because the data is split across multiple different tables, and combining them allows course-level averages to be calculated accurately. The query also uses window logic through a window function that computes the average quiz score for each department. Unlike a regular GROUP BY, the window function calculates the department average while still keeping each course's row visible. This makes it possible to directly compare each course's average quiz score to the overall performance of its department in the same output. There are no filters in this query, meaning all available quiz data is included. This provides a full view of quiz performance across all courses and departments without restricting the analysis to any specific subset of the data.

Why the insight matters to the business:

This insight helps the business quickly see which courses are performing as expected and which may need extra attention. Courses that fall below their department's average may indicate areas where students struggle or where teaching or assessments could be improved. Courses that perform above average can point to effective practices worth sharing. By spotting these patterns early, academic leaders can make better decisions about where to focus support, how to improve course quality, and how to create a more consistent learning experience for students.

---

5 Standard Analytical Queries

Query 1:

Business Question: Which courses appear to be most difficult based on average assignment scores?

SQL Query: located in the file, 'TASK2-sub2.sql'

Output screenshot:

course_id	course_code	course_title	department	avg_assignment_score
28	Physics294	Synergistic well-modul	Physics	49.9747251951841
46	CS480	Triple-buffered radica	CS	49.9752670161643
37	Physics489	Adaptive motivating pa	Physics	56.2151195746873
21	Economics358	Sharable empowering fu	Economics	57.0223833669047
3	Physics430	Triple-buffered 6thgen	Physics	58.7193934783412
5	Business175	Innovative zero-defect	Business	58.8757272451995
29	Math279	Synchronized attitude-	Math	58.9186513852013
40	Math367	Secured value-added su	Math	58.9596260564984

Interpretation of findings:

The query calculates the average assignment score for each course and orders them from lowest to highest, allowing us to see which courses students appear to struggle with the most. Courses that rise to the top of this list (lowest average scores) can be considered more difficult based on student performance. These lower averages may indicate assignments that are challenging, stricter grading, or course content that students find harder to engage with. Across departments, the results show where students face the most difficulty and which courses may require further instructional attention..

Explanation of joins, filters, window logic:

The query links courses to their assignments and then brings in student submissions to calculate an overall average score per course. The left join ensures all assignments are considered, while the HAVING clause removes courses without submission data. Ordering by

average score in ascending order allows the analysis to quickly identify the courses with the lowest student performance.

Why the insight matters to the business:

Identifying difficult courses is essential for improving student outcomes and maintaining quality. Low average scores can signal where additional academic support, clearer assessment design, or revised course structure may be needed. This insight helps institutions proactively address challenges before they impact retention or overall student success.

Query 2:

Business Question: How does assignment lateness impact student grades?

SQL Query: located in the file, 'TASK2-sub2.sql'

Output screenshot:

submission_timeliness	num_submissions	avg_score
Late	6721	62.6419079366377
On time or early	11245	66.4909697982665

Interpretation of findings:

This query separates on-time submissions from late ones and compares their average scores. If late submissions show significantly lower average performance, it suggests that students who miss deadlines tend to earn lower grades. If the averages are close, lateness may not strongly influence grading in this dataset. This comparison provides a direct indicator of how timeliness aligns with student outcomes.

Explanation of joins, filters, window logic:

The query joins submissions to assignments so each submission can be evaluated against the correct assignment due date. It uses SQLite's JULIANDAY() function to compare

timestamps, categorizing each submission using a CASE statement as either “On time or early” or “Late.” We then aggregate by this category to count submissions and compute their average score. The WHERE clause removes ungraded submissions, ensuring the averages reflect actual grading outcomes.

Why the insight matters to the business:

Understanding how lateness affects grades helps instructors and administrators refine deadline policies and student support strategies. If late submissions consistently receive lower scores, it may indicate the need for interventions such as earlier reminders, clearer guidance, or more targeted academic support. This type of insight can also strengthen early alert systems by identifying behavioral patterns, such as frequent lateness, that often correlate with academic risk.

Query 3:

Business Question: Which instructors have the highest average student performance?

SQL Query: N/A

Output screenshot: N/A

Interpretation of findings:

The current LMS schema does not support this analysis because there is no relationship linking instructors to the courses they teach, making it impossible to attribute student performance data to specific instructors. To enable this type of instructional performance analysis, the schema would need to be extended to include a direct relationship e.g. instructor_id field in the courses table (or a separate course to instructor mapping table), so that student outcomes can be accurately attributed to individual instructors.

Explanation of joins, filters, window logic: N/A

Why the insight matters to the business: N/A

Query 4

Business Question: How has student enrollment changed across semesters?

SQL Query: located in the file, 'TASK2-sub2.sql'

Output screenshot:

4 rows • 211ms

semester	total_enrollments
2024SP	1258
2024FA	1215
2025SP	1217
2025FA	1234

Interpretation of findings:

This query shows how overall enrollment shifts from one semester to the next. Increases may reflect strong course demand, effective program outreach, or changes in academic planning, while declines can signal reduced student interest, scheduling constraints, or broader academic-cycle effects. The semester-by-semester totals provide a clear view of enrollment trends and highlight notable changes over time. Overall, enrollments remain in the low to mid 1200s. The decrease from 1258 to 1215 likely reflects a significant number of students graduating in the spring semester (if graduation data was available, we could verify this or not).

Explanation of joins, filters, window logic:

This analysis relies solely on the enrollments table, grouping records by semester to produce aggregate counts. Because each row in enrollments represents a student registered in a course, counting rows reflects total course-level enrollments. Sorting by semester places the results in chronological order, enabling easy comparison across terms.

Why the insight matters to the business:

Tracking enrollment changes across semesters is important for understanding program growth, planning resources, and identifying patterns in student engagement. Recognizing terms with unusually high or low enrollment allows administrators to adjust course offerings, staffing, and support resources as needed. Ongoing monitoring of these trends helps ensure that decisions about the learning environment are grounded in data.

Query 5

Business Question: Which students are academically at risk based on consistently low scores?

SQL Query: located in the file, 'TASK2-sub2.sql'

Output screenshot:

student_id	student_name	avg_score	overall_avg	risk_status
169	Jonathan Harris	58.8197781127143	65.0510530236919	At Risk
100	Joseph Caldwell	59.2178854932289	65.0510530236919	At Risk
127	Thomas Liu	59.6821272880513	65.0510530236919	At Risk
43	John Levine	59.9955258111249	65.0510530236919	At Risk
97	Ryan Martinez	60.3696685806957	65.0510530236919	At Risk
208	John Barber	60.4886578096103	65.0510530236919	At Risk
349	Ashley Vargas	60.5497248762759	65.0510530236919	At Risk
219	Erica Holland	60.6617741419715	65.0510530236919	At Risk

student_id	student_name	avg_score	overall_avg	risk_status
221	Dr. Emily Thomas	69.1294366829705	65.0510530236919	Not At Risk;
66	Tony Mcbride	69.134449579882	65.0510530236919	Not At Risk;
285	Amanda Bailey	69.2855793734779	65.0510530236919	Not At Risk;
199	Christopher Davis	69.6488670509824	65.0510530236919	Not At Risk;
2	Johnny Gilbert	69.6708620580314	65.0510530236919	Not At Risk;
166	Jennifer Schultz	70.2118930534955	65.0510530236919	Not At Risk;
25	Troy Hernandez	70.6720269889399	65.0510530236919	Not At Risk;

Interpretation of findings:

This analysis identifies students as academically ‘At Risk’ or ‘Not At Risk’ based on how their average score compares to the overall class average. Students whose averages fall below this benchmark are considered ‘At Risk’, indicating that they are performing consistently lower than their peers across graded work. Because this approach is relative, it operates much like grading on a curve, where a student’s status is determined by comparison to the cohort rather than a fixed standard. The resulting list highlights which students are falling behind the broader performance level reflected in the LMS data.

Explanation of joins, filters, window logic:

The query joins the students table with submissions so that each student can be linked to their graded work. It then computes two key aggregates: each student’s average score and the overall average across all submissions, obtained via a small table joined into the query. Using a boolean comparison (AVG(s.score) < overall_avg), the query flags whether each student’s average is below this benchmark and transforms that result into a textual label, “At Risk” or “Not At Risk,” without relying on CASE or WHEN. Grouping by student ensures one summary row per learner, with their ID, name, average score, and risk label.

Why the insight matters to the business:

Even with a curve-based definition, comparing students to the overall performance level provides a useful signal for identifying those who may need additional support. Students flagged as *At Risk* are not only those with isolated low scores, but those whose overall average is consistently below the group, suggesting persistent challenges with the course content. This insight can inform targeted outreach e.g. tutoring referrals, check-ins from advisors, and helps instructors quickly see which students may be most vulnerable to falling further behind as the term progresses, ultimately failing the class.

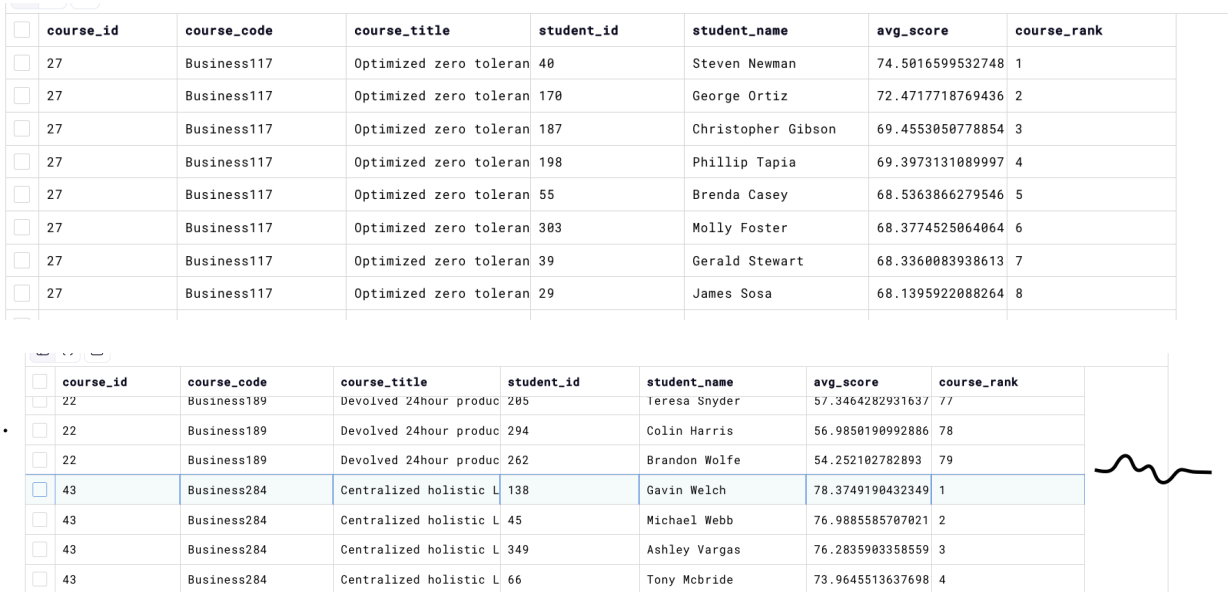
2 Window Function Queries

Query 1

Business question: What is each student's rank within each course based on average score?

SQL Query: located in the file, 'TASK2-sub2.sql'

Screenshot of output:



course_id	course_code	course_title	student_id	student_name	avg_score	course_rank
27	Business117	Optimized zero toleran	40	Steven Newman	74.5016599532748	1
27	Business117	Optimized zero toleran	170	George Ortiz	72.4717718769436	2
27	Business117	Optimized zero toleran	187	Christopher Gibson	69.4553050778854	3
27	Business117	Optimized zero toleran	198	Phillip Tapia	69.3973131089997	4
27	Business117	Optimized zero toleran	55	Brenda Casey	68.5363866279546	5
27	Business117	Optimized zero toleran	303	Molly Foster	68.3774525064064	6
27	Business117	Optimized zero toleran	39	Gerald Stewart	68.3360083938613	7
27	Business117	Optimized zero toleran	29	James Sosa	68.1395922088264	8

course_id	course_code	course_title	student_id	student_name	avg_score	course_rank
22	Business189	Devolved 24hour produc	205	Teresa Snyder	57.3464282931637	77
22	Business189	Devolved 24hour produc	294	Colin Harris	56.9850190992886	78
22	Business189	Devolved 24hour produc	262	Brandon Wolfe	54.252102782893	79
43	Business284	Centralized holistic L	138	Gavin Welch	78.3749190432349	1
43	Business284	Centralized holistic L	45	Michael Webb	76.9885585707021	2
43	Business284	Centralized holistic L	349	Ashley Vargas	76.2835903358559	3
43	Business284	Centralized holistic L	66	Tony Mcbride	73.9645513637698	4

- etc.

Interpretation of findings:

This query ranks students within each course according to their average assignment score. A rank of **1** indicates the highest-performing student in that course, while larger rank values indicate relatively lower performance. Because the ranking uses averages, it reflects sustained performance rather than individual high or low outliers. This allows instructors or administrators to identify top performers, middle performers, and those who may require additional support within a specific course.

Explanation of joins, filters, window logic:

The query joins students to their submissions, submissions to assignments, and assignments to courses, ensuring that each score is correctly associated with both the student and the course where it was earned. Scores are filtered to include only non-null values. The GROUP BY produces each student's average score per course. The RANK() OVER window function partitions the data by course, then orders students within each course from highest to lowest average score. This window logic allows ranking without collapsing results into a single aggregated dataset.

Why the insight matters to the business:

Ranking students within each course provides instructors with a clear picture of performance distribution and relative standing. This can help identify students excelling in the material, those maintaining average performance, and those who may be struggling. Such insights support targeted interventions, grading curve considerations, pedagogical adjustments, and early alerts for academic risk. By using window functions, the analysis preserves detail while allowing powerful comparisons within course-level contexts.

Query 2

Business question: How does each student's cumulative GPA-style average evolve semester by semester?

SQL Query: located in the file, 'TASK2-sub2.sql'

Screenshot of output:

student_id	student_name	semester	sem_avg	sem_attempts	cumulative_gpa
1	William Hill	2024SP	57.24364436193	9	57.24
1	William Hill	2024FA	67.85290362458	23	64.87
1	William Hill	2025SP	63.84707067292	7	64.69
1	William Hill	2025FA	63.96603550971	28	64.38
2	Johnny Gilbert	2024SP	65.38570039950	14	65.39
2	Johnny Gilbert	2024FA	67.11091464055	10	66.1
2	Johnny Gilbert	2025SP	75.04331326889	11	68.91
2	Johnny Gilbert	2025FA	72.47416654987	13	69.88

Interpretation of findings:

This query shows, for each student and each semester, their semester average (sem_avg) and a cumulative GPA-style average (cumulative_gpa) up to and including that term. Early rows for a student reflect performance in their first semester; later rows show how that cumulative average moves as new semesters are added. A rising cumulative GPA suggests steady improvement over time, while a flat or declining pattern can indicate that more recent semesters are not offsetting earlier performance. Looking at these trends helps you see not just where students are now, but how they got there.

Explanation of joins, filters, window logic:

The inner query aggregates scores at the student–semester level: it joins students to submissions to assignments to courses to enrollments so that each graded submission is tied to both the correct student and semester. It then computes the average score and number of graded items for each student in each term, and assigns a numeric sem_order so semesters can be ordered consistently. The outer query uses window functions to build a cumulative GPA-style measure: it keeps a running sum of quality points (sem_avg * sem_attempts) and a running sum of attempts (sem_attempts) partitioned by student and ordered by semester, then divides them to get the cumulative average at each step.

Why the Insight Matters:

Tracking cumulative performance semester by semester provides a much richer view of student success than a single snapshot. It allows you to identify students who are improving over time, those whose performance is stagnating, and those whose GPA-style average is steadily declining and may need intervention. At the program level, these trajectories can also reveal whether students tend to struggle in particular stages of the curriculum (e.g., first term vs. later terms), helping inform advising strategies, course sequencing, and academic support services.