

Patrick Murray

Professor Rivas

Database Management

26 February 2026

Part A:

The screenshot shows a database query editor interface. The top section, titled 'Query', contains an SQL query. The bottom section, titled 'Data Output', displays the results of the query in a table format. The table has three columns: 'studentname', 'title', and 'term'. The results show four rows of data.

```
1
2 SELECT s.studentName, c.title, e.term
3 FROM student s
4 JOIN enrollment e ON s.studentID = e.studentID
5 JOIN course c ON e.courseID = c.courseID
6 WHERE e.term = 'Spring2026'
7
```

	studentname character varying	title character varying	term character varying
1	Brian Smith	Database Systems	Spring2026
2	Carla Gomez	Database Systems	Spring2026
3	David Lee	Physics I	Spring2026
4	Emma Brown	Organic Chemistry	Spring2026

Query Query History

```
1 SELECT c.courseID, c.title, COUNT (e.courseID) AS enrollments
2 FROM course c
3 Left join enrollment e
4 ON c.courseID = e.courseID
5 AND e.term = 'Spring2026'
6 GROUP BY c.courseID, c.title
7
```

Data Output Messages Notifications

Showing rows: 1 to 10 Page No: 1 of 1

	courseid [PK] integer	title character varying	enrollments bigint	
1	108	Microeconomics	1	
2	106	World History	1	
3	102	Calculus I	0	
4	107	Engineering Design	0	
5	110	Database Manageme...	1	
6	101	Database Systems	3	
7	103	General Biology	0	
8	104	Physics I	1	
9	105	Organic Chemistry	2	
10	109	Intro to Unity	0	

```
Query  Query History
1  SELECT s.major, COUNT (s.studentID) as majorCount
2  FROM student s
3  GROUP BY s.major
4  ORDER BY majorCount desc
5
6
7
```

Data Output Messages Notifications

Showing rows: 1 to 9 Page No: 1 of 1

	major character varying	majorcount bigint
1	Computer Science	2
2	Chemistry	1
3	Mathematics	1
4	History	1
5	Economics	1
6	Physics	1
7	Engineering	1
8	Data Science	1
9	Biology	1

Part B:

```
Query Query History
1 SELECT s.studentName from student s
2 WHERE s.studentID IN (
3     SELECT e.studentID
4     FROM enrollment e
5     WHERE e.courseID = 110 AND e.term = 'Spring2026'
6 )
7
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	studentname character varying
1	John Shorts

cmpt308_lab4/postgres@PostgreSQL 18

Query Query History

```
1
2 SELECT c.courseID, c.title, COUNT (*) as num_enrollments
3 FROM course c
4 INNER JOIN enrollment e
5 ON c.courseID = e.courseID
6 WHERE e.term = 'Spring2026'
7 GROUP BY c.courseID, c.title
8
```

Data Output Messages Notifications

Showing rows: 1 to 6 Page No: 1 of 1

	courseid [PK] integer	title character varying	num_enrollments bigint
1	101	Database Systems	3
2	104	Physics I	1
3	105	Organic Chemistry	2
4	106	World History	1
5	108	Microeconomics	1
6	110	Database Manageme...	1

cmpt308_lab4/postgres@PostgreSQL 18

Query Query History

```
1
2 SELECT c.courseID, COUNT (*) as num_enrollments
3 FROM course c
4 INNER JOIN enrollment e
5 ON c.courseID = e.courseID
6 WHERE e.term = 'Spring2026'
7 GROUP BY c.courseID
8 HAVING COUNT(*) >= 3
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	courseid [PK] integer	num_enrollments bigint
1	101	3

Part C:

The screenshot shows a SQL IDE interface with a dark theme. The main editor displays a SQL query with line numbers 1 through 8. The query is a nested SELECT statement. Below the editor, there is a toolbar with icons for file operations, a 'SQL' button, and pagination controls. The 'Data Output' tab is active, showing a single row of results with the column 'studentname' and the value 'John Shorts'.

```
1 SELECT s.studentName from student s
2 WHERE s.studentID IN (
3     SELECT e.studentID
4     FROM enrollment e
5     WHERE e.courseID = 110 AND e.term = 'Spring2026'
6 )
7
8
```

Showing rows: 1 to 1 Page No: 1 of 1

	studentname character varying
1	John Shorts

Query Query History

```
1 SELECT c.courseID, c.title
2 FROM course c
3 WHERE EXISTS (
4     SELECT 1
5     FROM enrollment e
6     WHERE c.courseID = e.courseID AND e.term = 'Spring2026'
7
8 )
```

Data Output Messages Notifications

Showing rows: 1 to 6 Page No: 1 of 1

	courseid [PK] integer	title character varying
1	110	Database Manageme...
2	101	Database Systems
3	105	Organic Chemistry
4	108	Microeconomics
5	106	World History
6	104	Physics I

cmpt308_lab4/postgres@PostgreSQL 18

Query Query History

```
1 SELECT studentID FROM enrollment
2 WHERE term = 'Spring2026'
3 GROUP BY studentID
4 HAVING COUNT (*) >= 2
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	studentid integer
1	10

Part D:

Query Query History

```
1 SELECT studentID, studentName
2 FROM student
3 WHERE major = 'Computer Science'
4 UNION
5 SELECT studentID, studentName
6 FROM student
7 WHERE major = 'Data Science'
8 ORDER BY studentID desc
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	studentid integer	studentname character varying
1	10	John Shorts
2	9	Todd Barn
3	1	Alice Johnson

Query Query History

```
1 SELECT studentID, studentName
2 FROM student
3 WHERE major = 'Computer Science'
4 EXCEPT
5 SELECT s.studentID, s.studentName
6 FROM student s
7 JOIN enrollment e
8 ON e.studentID = s.studentID
9 WHERE e.courseID = '110' AND e.term = 'Spring2026'
```

Data Output Messages Notifications

Showing rows: 1 to 2 Page No: 1 of 1

	studentid integer	studentname character varying
1	1	Alice Johnson
2	9	Todd Barn

Part E:

The second tool that I decided to use was ChatGPT since it's my go to AI. I first pasted all of my creation and insertion statements before I followed up with the prompt which was, "This is the basis of my database. I need you to make a query that returns a list of course_id and title for all courses, and also returns the number of distinct students enrolled in each course for term 'Spring2026'. Include courses with zero enrollments. Sort by the count (highest to lowest), then by course_id."

ChatGPT:

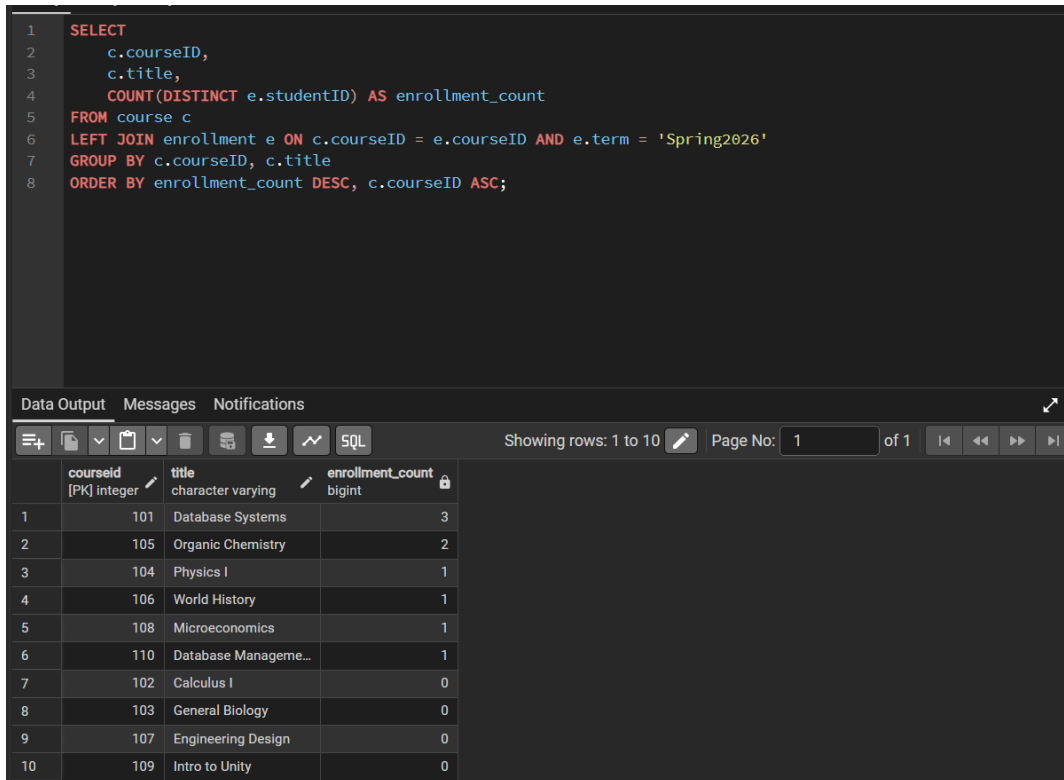
```
SELECT
    c.courseID,
    c.title,
    COUNT(DISTINCT e.studentID) AS student_count
FROM course c
LEFT JOIN enrollment e
    ON c.courseID = e.courseID
    AND e.term = 'Spring2026'
GROUP BY c.courseID, c.title
ORDER BY student_count DESC, c.courseID;
```

Query		Query History	
1	SELECT		
2	c.courseID,		
3	c.title,		
4	COUNT(DISTINCT e.studentID) AS student_count		
5	FROM course c		
6	LEFT JOIN enrollment e		
7	ON c.courseID = e.courseID		
8	AND e.term = 'Spring2026'		
9	GROUP BY c.courseID, c.title		
10	ORDER BY student_count DESC, c.courseID;		

Data Output		Messages	Notifications
Showing rows: 1 to 10	Page No: 1	of 1	
courseid	title	student_count	
[PK] integer	character varying	bigint	
1	101 Database Systems	3	
2	105 Organic Chemistry	2	
3	104 Physics I	1	
4	106 World History	1	
5	108 Microeconomics	1	
6	110 Database Manageme...	1	
7	102 Calculus I	0	
8	103 General Biology	0	
9	107 Engineering Design	0	
10	109 Intro to Unity	0	

Copilot:

```
SELECT
    c.courseID,
    c.title,
    COUNT(DISTINCT e.studentID) AS enrollment_count
FROM course c
LEFT JOIN enrollment e ON c.courseID = e.courseID AND e.term = 'Spring2026'
GROUP BY c.courseID, c.title
ORDER BY enrollment_count DESC, c.courseID ASC;
```



The screenshot shows a SQL IDE interface. The top pane contains the SQL query. The bottom pane shows the 'Data Output' tab with a table of results. The table has three columns: 'courseid' (integer, primary key), 'title' (character varying), and 'enrollment_count' (bigint). The results are sorted by enrollment_count in descending order, then by courseid in ascending order.

	courseid [PK] integer	title character varying	enrollment_count bigint
1	101	Database Systems	3
2	105	Organic Chemistry	2
3	104	Physics I	1
4	106	World History	1
5	108	Microeconomics	1
6	110	Database Manageme...	1
7	102	Calculus I	0
8	103	General Biology	0
9	107	Engineering Design	0
10	109	Intro to Unity	0

Both of these queries worked and gave the exact same output when I ran them. I'd say Copilot was better for this task overall. The explanations it gave for the key components of the query were both shorter and more informative than ChatGPT.

Github repo: [Not-Patrick1/DatabaseLabs: First lab for DB management.](#)