# Queries with OUTPUT Final Submission

Sujal Prajapati - 202301478 Kalp Shah - 202301481 Vraj Parikh - 202301440

# **SQL Queries**:

#### **Elective Management Queries:**

1. Insert a new elective course into the database

```
INSERT INTO Course (CourseID, CourseName, Branch, Credits, Lecture, Tutorial, Labs, Material)

VALUES ('CSE01', 'Data Science', 'CSE', 4, 3, TRUE, TRUE, 'Material content here');

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 107 msec.
```

2. Update the details of an existing elective course

```
571 VUPDATE Course
572 SET CourseName = 'Advanced Data Science',
573 Credits = 5,
Material = 'Updated material content'
575 WHERE CourseID = 'CSE01';
576

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 227 msec.
```

3. Delete an elective course from the database

```
577 V

578
WHERE CourseID = 'CSE01';

Data Output Messages Notifications

DELETE 1

Query returned successfully in 166 msec.
```

#### 4. Retrieve a list of all elective courses



### 5. Retrieve details of a specific elective course by course ID



## 6. Find all elective courses offered by a specific department (Branch)



#### 7. Find all elective courses with a specific credit count



#### **Student Enrollment Queries:**

1. Enroll a student in an elective course.

```
INSERT INTO Enrollment (StudentID, CourseID, GradeReceived)
VALUES ('202101001', 'AA02', NULL);

766
767
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 61 msec.

2. Remove a student from an elective course.

```
766
767 DELETE FROM Enrollment
768 WHERE StudentID = '202101001' AND CourseID = 'GT01';
```

Data Output Messages Notifications

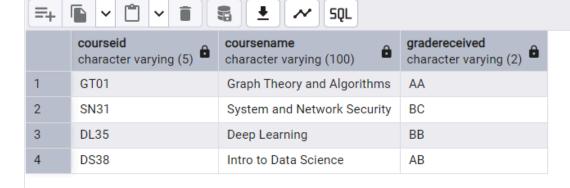
DELETE 0

Query returned successfully in 349 msec.

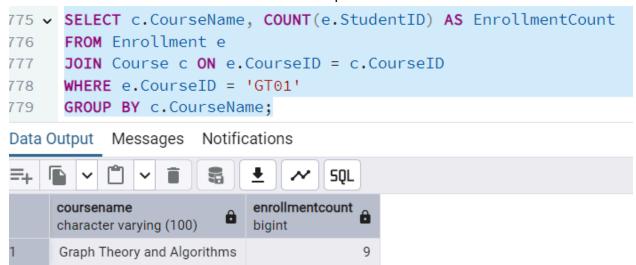
3. Retrieve all courses a specific student has registered for.

```
770 > SELECT c.CourseID, c.CourseName, e.GradeReceived
771    FROM Enrollment e
772    JOIN Course c ON e.CourseID = c.CourseID
773    WHERE e.StudentID = '202101002';
774
```

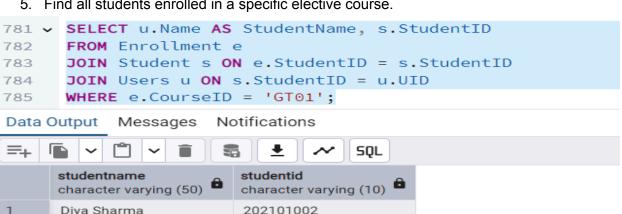
Data Output Messages Notifications



4. Find the number of students enrolled in a specific elective course.

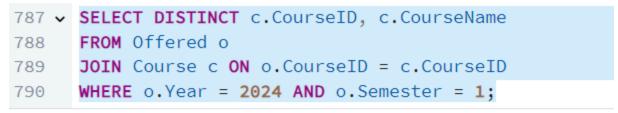


5. Find all students enrolled in a specific elective course.

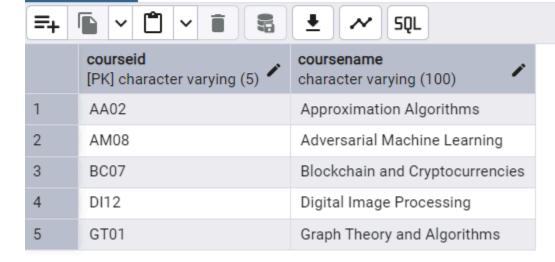


	studentname character varying (50)	studentid character varying (10)
1	Diya Sharma	202101002
2	Ananya Reddy	202104001
3	Krish Malhotra	202104002
4	Ishaan Kumar	202111001
5	Aanya Verma	202201001
6	Aditi Nair	202211001
7	Advait Trivedi	202301001
8	Kabir Srinivasan	202311001
9	Pari Gupta	202401001

6. Retrieve all elective courses available for a particular semester.

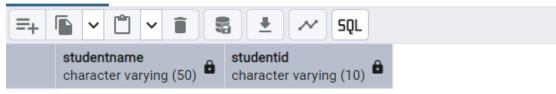


# Data Output Messages Notifications



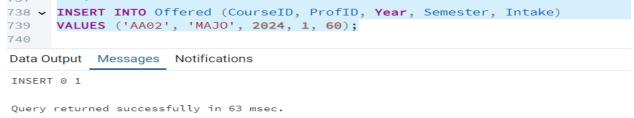
7. Retrieve students who have not enrolled in any elective course.

# Data Output Messages Notifications

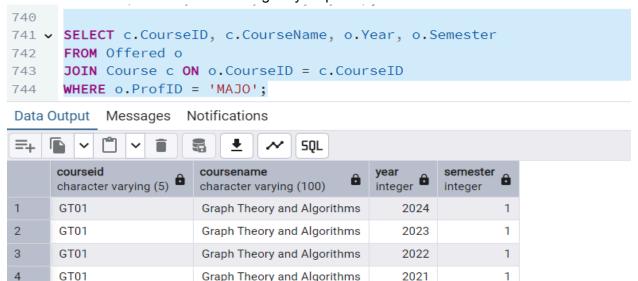


#### **Faculty Related Queries:**

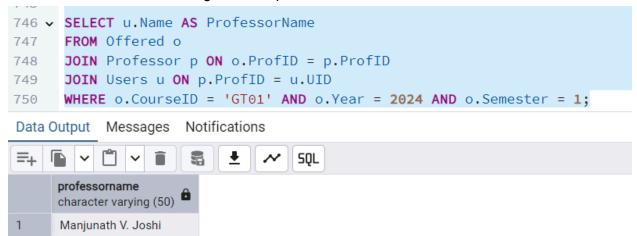
1. Assign an instructor to a specific elective course.



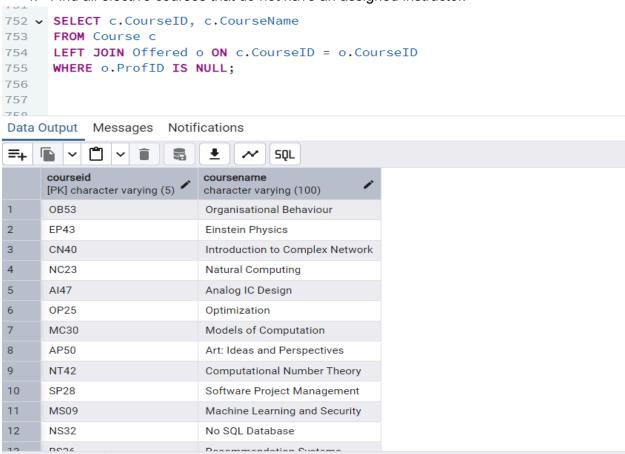
2. Retrieve all elective courses taught by a specific instructor.



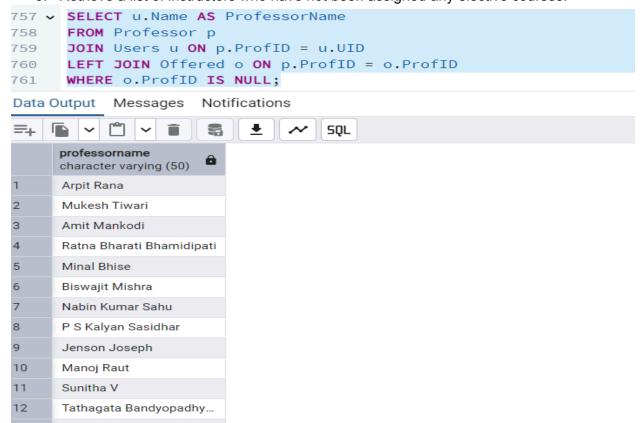
3. Find the instructor assigned to a specific elective course.



4. Find all elective courses that do not have an assigned instructor.

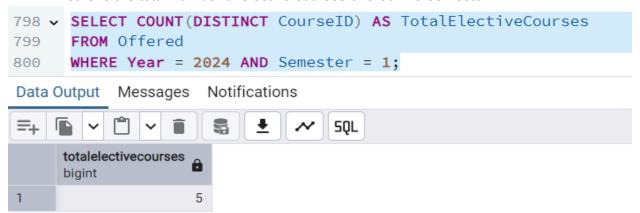


Retrieve a list of instructors who have not been assigned any elective courses.

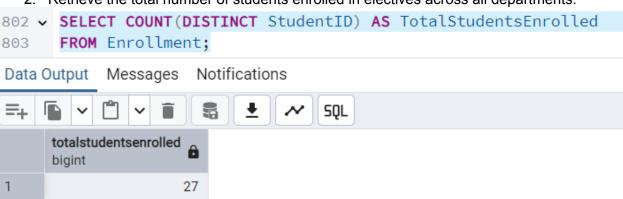


#### **Course Related Queries:**

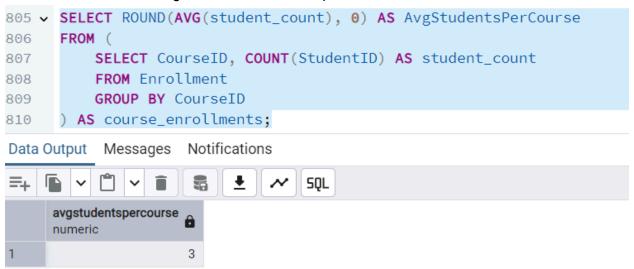
1. Retrieve the total number of elective courses offered in a semester.



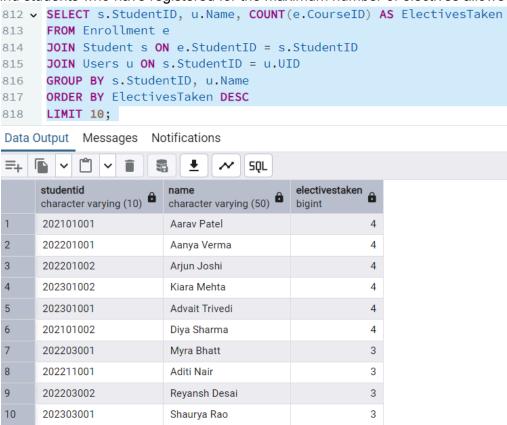
2. Retrieve the total number of students enrolled in electives across all departments.



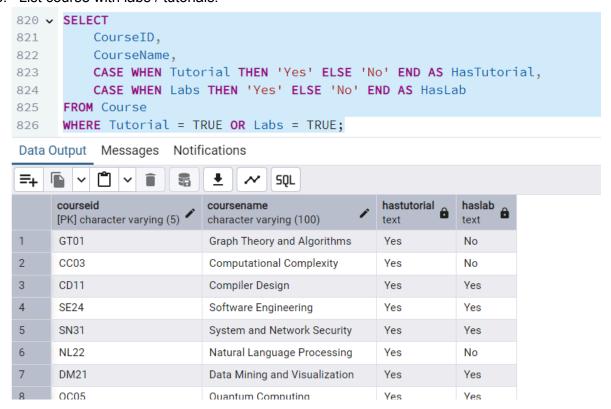
3. Retrieve the average number of students per elective course.



4. Find students who have registered for the maximum number of electives allowed.



5. List course with labs / tutorials.



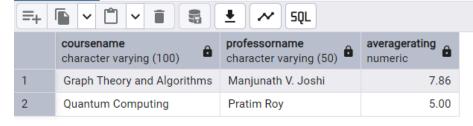
#### **Rating Related Queries:**

Query Query History

1. Top rated course with professor list in a particular semester and year

```
554 V SELECT
555
          c.CourseName,
556
          p.Name AS ProfessorName,
557
          ROUND(AVG(r.Overall_Rating)::numeric, 2) AS AverageRating
558
      FROM Rating r
      JOIN Course c ON r.CourseID = c.CourseID
559
      JOIN Professor pr ON r.ProfID = pr.ProfID
560
561
      JOIN Users p ON pr.ProfID = p.UID
      JOIN Offered o ON r.CourseID = o.CourseID AND r.ProfID = o.ProfID
562
      WHERE o.Year = 2023 AND o.Semester = 1
563
      GROUP BY c.CourseName, p.Name
564
565
      ORDER BY AverageRating DESC
566
      LIMIT 2;
```

## Data Output Messages Notifications



#### 2. Most Difficult Course List

```
592 SELECT
593
          c.CourseName,
          ROUND(AVG(r.Course_Difficulty)::numeric, 2) AS AvgDifficulty
594
      FROM Rating r
595
      JOIN Course c ON r.CourseID = c.CourseID
596
      GROUP BY c.CourseName
597
      ORDER BY AvgDifficulty DESC
598
599
      LIMIT 10;
600
```

#### Data Output Messages Notifications

=+		<u>♣</u>
	coursename character varying (100)	avgdifficulty numeric
1	Quantum Computing	9.00
2	Introduction to Cryptography	8.00
3	Wireless System Design	8.00
4	Deep Learning	8.00
5	Operating Systems	7.00
6	Modern Algebra	6.67

#### Most Useful Courses

```
601 v SELECT c.CourseName, COUNT(p.CourseID) AS NumOfDependentCourses
       FROM Prerequisite p
       JOIN Course c ON p.Preq_CourseID = c.CourseID
603
604
       GROUP BY c.CourseName
       ORDER BY NumOfDependentCourses DESC
605
606
       LIMIT 5;
Data Output Messages Notifications
=+
                                        SQL
      coursename
                              numofdependentcourses
      character varying (100)
      Graph Theory and Algorithms
                                                   4
2
      Introduction to AI
                                                   3
                                                   3
3
      Modern Algebra
      Digital Signal Processing
                                                   2
4
5
      Intro to Data Science
                                                   1
```

### 4. Course offering average grade above 8

```
SELECT
    c.CourseName,
    ROUND (AVG (
        CASE
            WHEN e.GradeReceived = 'AA' THEN 10
            WHEN e.GradeReceived = 'AB' THEN 9
            WHEN e.GradeReceived = 'BB' THEN 8
            WHEN e.GradeReceived = 'BC' THEN 7
            WHEN e.GradeReceived = 'CC' THEN 6
            WHEN e.GradeReceived = 'CD' THEN 5
            WHEN e.GradeReceived = 'DD' THEN 4
            WHEN e.GradeReceived = 'DE' THEN 3
            WHEN e.GradeReceived = 'F' THEN 0
            ELSE NULL
        END
    )::numeric, 2) AS AvgGrade
FROM Enrollment e
JOIN Course c ON e.CourseID = c.CourseID
GROUP BY c.CourseName
HAVING AVG(
    CASE
        WHEN e.GradeReceived = 'AA' THEN 10
        WHEN e.GradeReceived = 'AB' THEN 9
        WHEN e.GradeReceived = 'BB' THEN 8
        WHEN e.GradeReceived = 'BC' THEN 7
        WHEN e.GradeReceived = 'CC' THEN 6
        WHEN e.GradeReceived = 'CD' THEN 5
        WUEN a CradeDeceived - IDDI TUEN 4
```

```
WHEN e.GradeReceived = 'DD' THEN 4

WHEN e.GradeReceived = 'DE' THEN 3

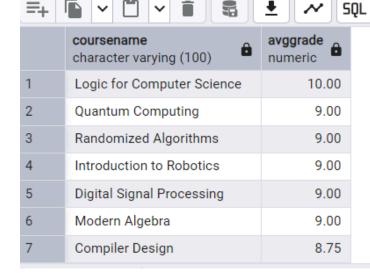
WHEN e.GradeReceived = 'F' THEN 0

ELSE NULL

END

ORDER BY AvgGrade DESC;
```

## Data Output Messages Notifications



#### 5. All ratings given by a particular student

```
645 	✓ SELECT c.CourseName, u.Name AS ProfessorName,
             r.Course_Difficulty, r.Tutorial_Lab_Difficulty,
646
647
             r.Project_Difficulty, r.Overall_Rating
648
      FROM Rating r
      JOIN Course c ON r.CourseID = c.CourseID
649
      JOIN Professor p ON r.ProfID = p.ProfID
650
      JOIN Users u ON p.ProfID = u.UID
651
652
      WHERE r.StudentID = '202101001'
      ORDER BY r.Overall_Rating DESC;
653
654
655
656
657
658
659
Data Output Messages Notifications
```

=+ 🕩 🗸 🖺 🕶 SQL Showing rows: 1 to 3						
	coursename character varying (100)	professorname character varying (50)	course_difficulty integer	tutorial_lab_difficulty integer	project_difficulty integer	overall_rating integer
1	Graph Theory and Algorithms	Manjunath V. Joshi	6	5	7	8
2	Modern Algebra	Gopinath Panda	8	6	[null]	7
3	Introduction to AI	Prasenjit Majumder	7	6	8	7

## 6. All ratings of a particular course with its average rating

```
657 SELECT
658
         c.CourseName,
659
         u.Name AS ProfessorName,
       r.Course_Difficulty,
r.Tutorial_Lab_Difficulty,
660
661
      r.Project_Difficulty,
r.Overall_Rating
662
663
664 FROM Rating r
JOIN Course c ON r.CourseID = c.CourseID
JOIN Professor p ON r.ProfID = p.ProfID
JOIN Users u ON p.ProfID = u.UID
668 WHERE r.CourseID = 'GT01'
669
670 UNION ALL
671
672
     -- Average rating for course GT01 (rounded to 2 decimal places)
673
     SELECT
         'Average for GT01' AS CourseName,
674
675
          NULL AS ProfessorName,
          NULL AS Course_Difficulty,
676
677
          NULL AS Tutorial_Lab_Difficulty,
678
          NULL AS Project_Difficulty,
679
         ROUND(AVG(Overall_Rating)::numeric, 2) AS Overall_Rating
680
     FROM Rating
681
      WHERE CourseID = 'GT01'
682
683
      ORDER BY ProfessorName NULLS LAST;
```

	coursename character varying	professorname character varying	course_difficulty integer	tutorial_lab_difficulty integer	project_difficulty integer	overall_rating numeric
1	Graph Theory and Algorithms	Manjunath V. Joshi	4	3	5	10
2	Graph Theory and Algorithms	Manjunath V. Joshi	7	6	8	7
3	Graph Theory and Algorithms	Manjunath V. Joshi	6	5	7	8
4	Graph Theory and Algorithms	Manjunath V. Joshi	8	7	9	6
5	Graph Theory and Algorithms	Manjunath V. Joshi	7	6	8	7
6	Graph Theory and Algorithms	Manjunath V. Joshi	5	4	6	9
7	Graph Theory and Algorithms	Manjunath V. Joshi	6	5	7	8
8	Average for GT01	[null]	[null]	[null]	[null]	7.86

## 7. Three most loved professors

P M Jat

Prasenjit Majumder

Manjunath V. Joshi

2

3

4

```
684 V SELECT
           u.Name AS ProfessorName,
685
686
           ROUND(AVG(r.Overall_Rating)::numeric, 2) AS AvgRating
687
      FROM Rating r
      JOIN Professor p ON r.ProfID = p.ProfID
688
689
      JOIN Users u ON p.ProfID = u.UID
      GROUP BY u.Name
690
691
      ORDER BY AvgRating DESC
692
      LIMIT 3;
Data Output Messages Notifications
=+
                                     SQL
     professorname
                         avgrating
     character varying (50)
                         numeric
```

# 8. Number of students with a particular grade in a specified course

9.00

8.33

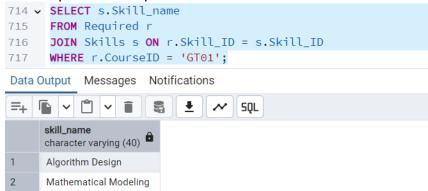
7.86

```
Query Query History
694 SELECT
695
          GradeReceived,
696
          COUNT(*) AS NumberOfStudents
      FROM Enrollment
697
698
      WHERE CourseID = 'GT01' AND GradeReceived IS NOT NULL
699
      GROUP BY GradeReceived
      ORDER BY
700
701
          CASE
702
               WHEN GradeReceived = 'AA' THEN 1
              WHEN GradeReceived = 'AB' THEN 2
703
              WHEN GradeReceived = 'BB' THEN 3
704
              WHEN GradeReceived = 'BC' THEN 4
705
              WHEN GradeReceived = 'CC' THEN 5
706
              WHEN GradeReceived = 'CD' THEN 6
707
              WHEN GradeReceived = 'DD' THEN 7
708
              WHEN GradeReceived = 'DE' THEN 8
709
              WHEN GradeReceived = 'F' THEN 9
710
711
               ELSE 10
712
          END;
713
Data Output Messages
                     Notifications
                           #
                                    SQL
gradereceived
                       numberofstudents
     character varying (2)
                       bigint
1
                                     5
2
     AB
                                     2
3
     BB
                                     1
```

1

#### **SKILLS and PreRequisite Related Queries:**

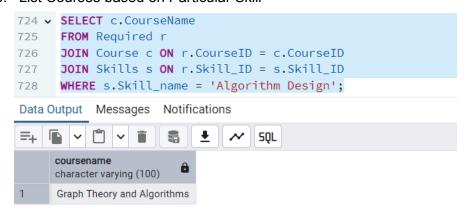
1. Skills acquired from a particular course



2. Prerequisite required for a particular course



3. List Courses based on Particular Skill



## **Specialization Related Queries:**

1. Professors with most specialization:

