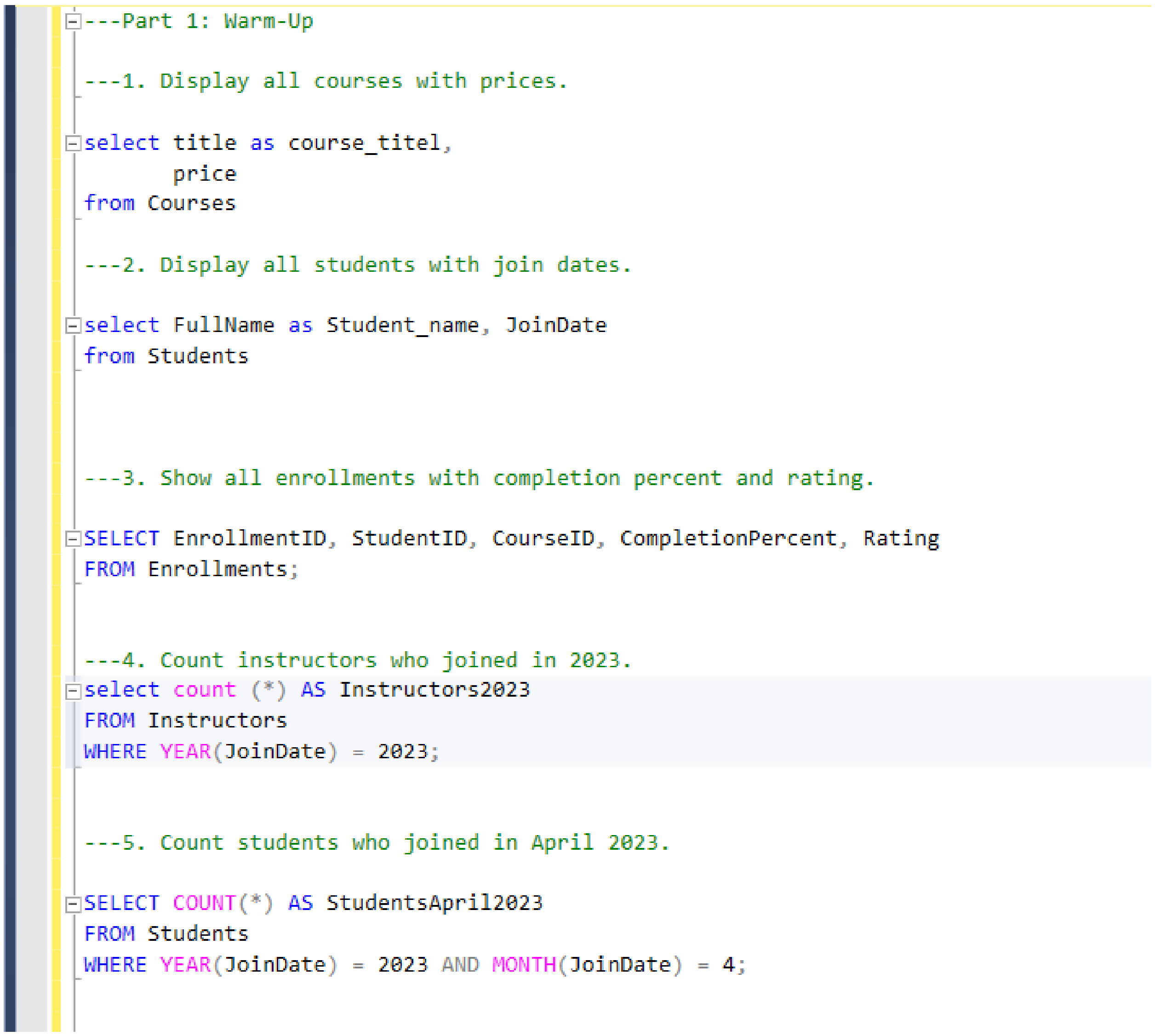
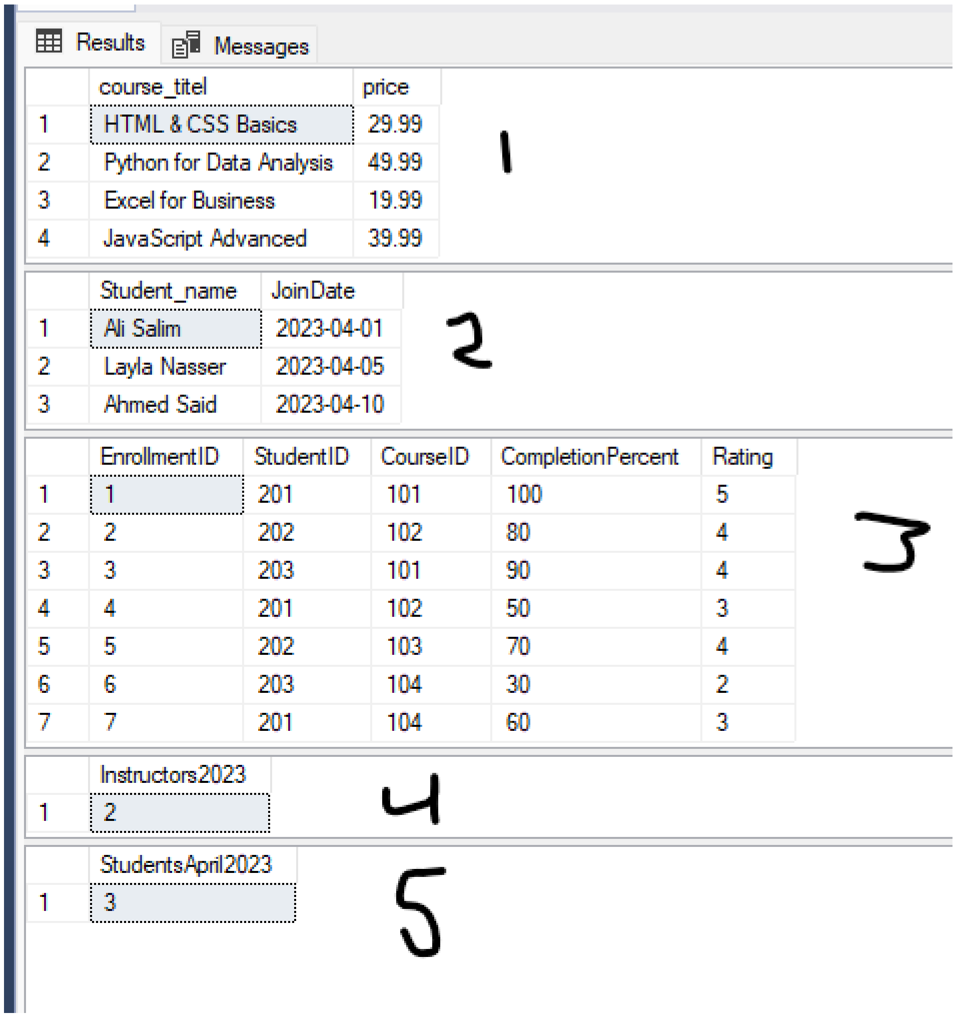
Aggregation Functions:



```sql
---Part 1: Warm-Up

---1. Display all courses with prices.

select title as course_titel,
       price
from Courses

---2. Display all students with join dates.

select FullName as Student_name, JoinDate
from Students



---3. Show all enrollments with completion percent and rating.

SELECT EnrollmentID, StudentID, CourseID, CompletionPercent, Rating
FROM Enrollments;


---4. Count instructors who joined in 2023.
select count (*) AS Instructors2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;


---5. Count students who joined in April 2023.

SELECT COUNT(*) AS StudentsApril2023
FROM Students
WHERE YEAR(JoinDate) = 2023 AND MONTH(JoinDate) = 4;
```



Results Messages

	course_titel	price
1	HTML & CSS Basics	29.99
2	Python for Data Analysis	49.99
3	Excel for Business	19.99
4	JavaScript Advanced	39.99

1

	Student_name	JoinDate
1	Ali Salim	2023-04-01
2	Layla Nasser	2023-04-05
3	Ahmed Said	2023-04-10

2

	EnrollmentID	StudentID	CourseID	CompletionPercent	Rating
1	1	201	101	100	5
2	2	202	102	80	4
3	3	203	101	90	4
4	4	201	102	50	3
5	5	202	103	70	4
6	6	203	104	30	2
7	7	201	104	60	3

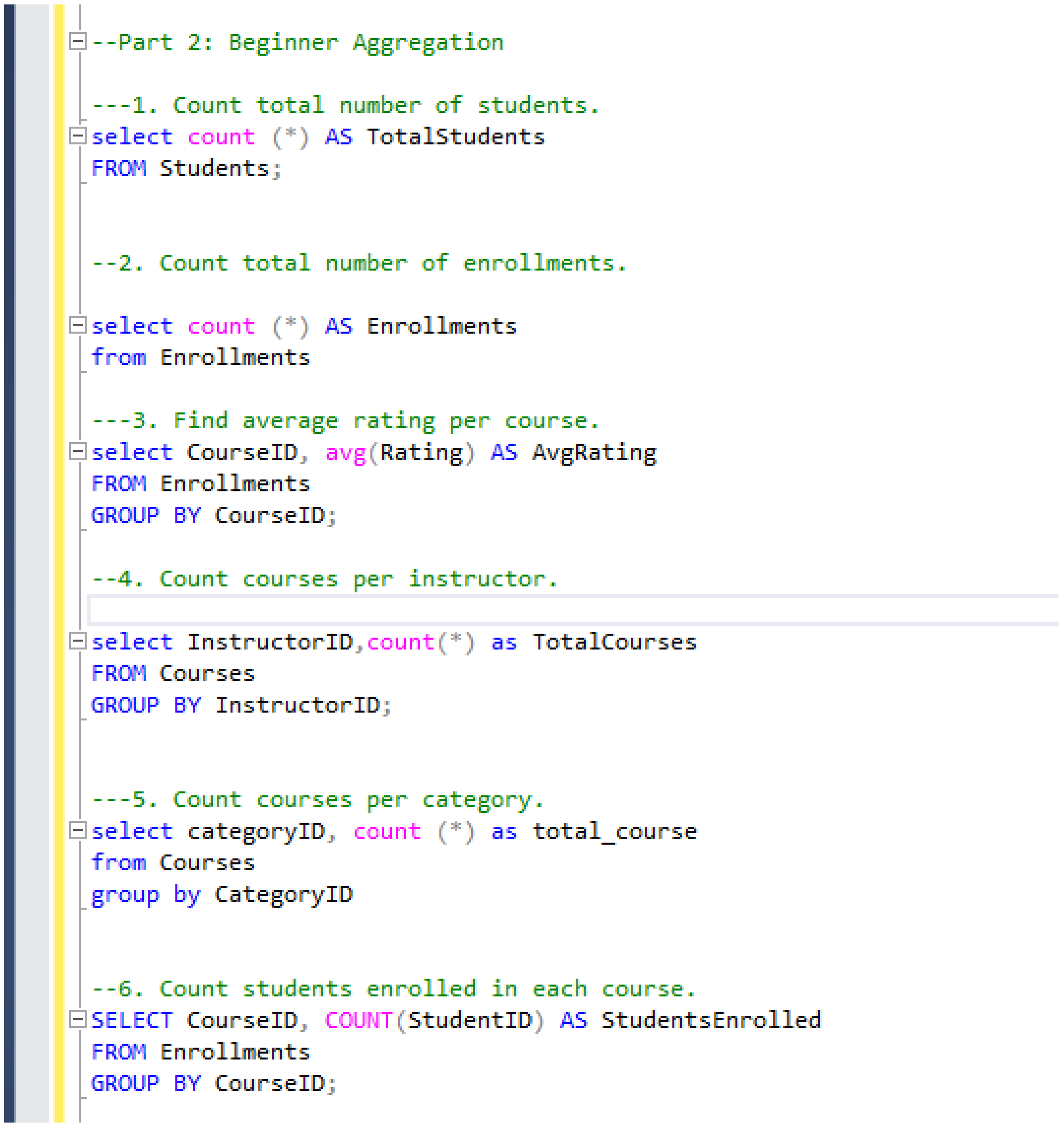
3

	Instructors2023
1	2

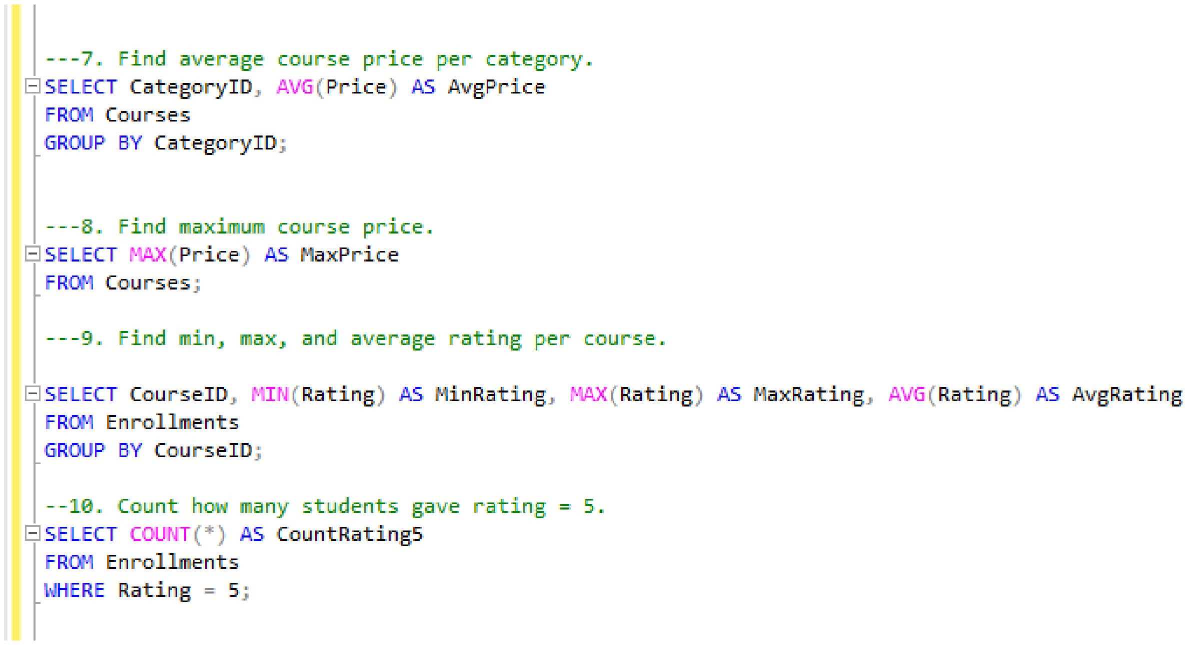
4

	StudentsApril2023
1	3

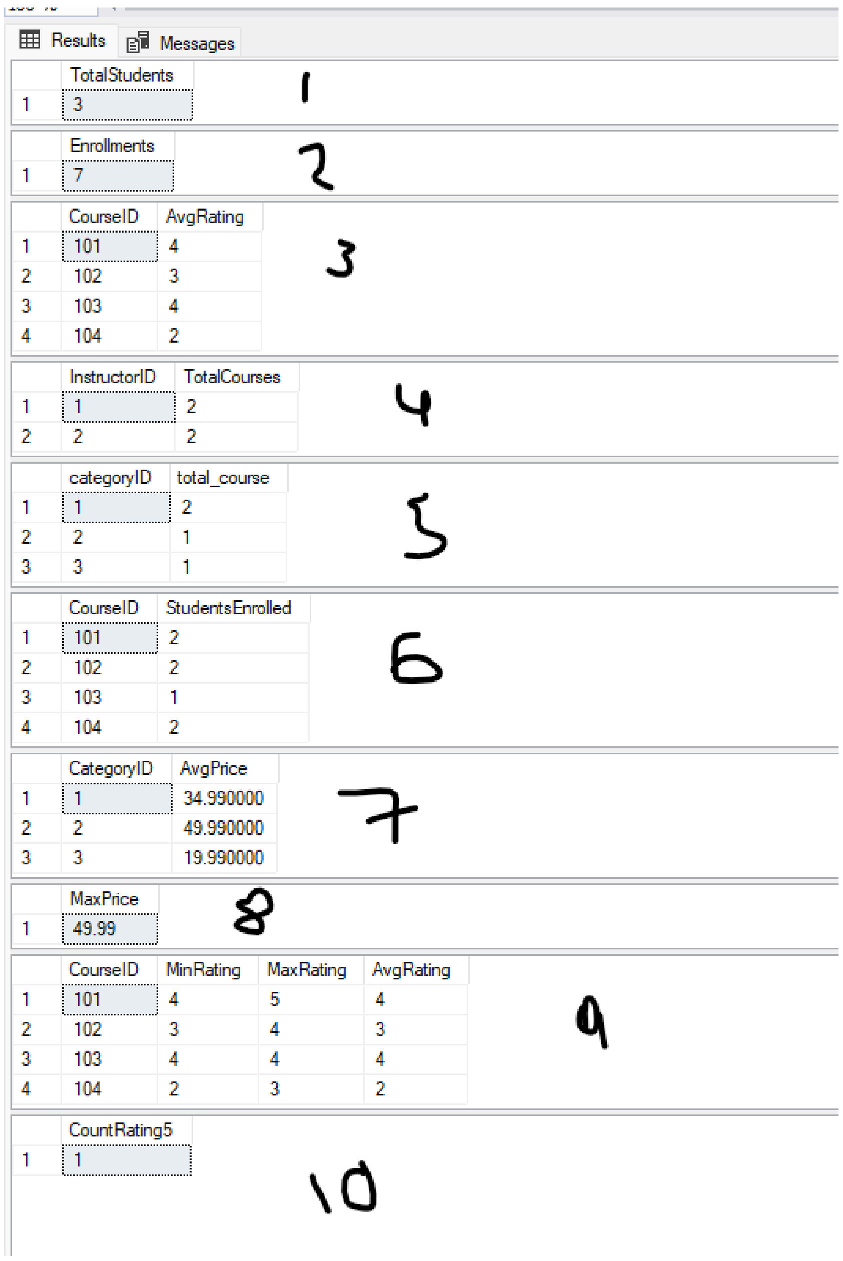
5



```
--Part 2: Beginner Aggregation

---1. Count total number of students.
select count (*) AS TotalStudents
FROM Students;


--2. Count total number of enrollments.

select count (*) AS Enrollments
from Enrollments

---3. Find average rating per course.
select CourseID, avg(Rating) AS AvgRating
FROM Enrollments
GROUP BY CourseID;

--4. Count courses per instructor.

select InstructorID,count(*) as TotalCourses
FROM Courses
GROUP BY InstructorID;


---5. Count courses per category.
select categoryID, count (*) as total_course
from Courses
group by CategoryID


--6. Count students enrolled in each course.
SELECT CourseID, COUNT(StudentID) AS StudentsEnrolled
FROM Enrollments
GROUP BY CourseID;
```



```sql
---7. Find average course price per category.
SELECT CategoryID, AVG(Price) AS AvgPrice
FROM Courses
GROUP BY CategoryID;


---8. Find maximum course price.
SELECT MAX(Price) AS MaxPrice
FROM Courses;

---9. Find min, max, and average rating per course.

SELECT CourseID, MIN(Rating) AS MinRating, MAX(Rating) AS MaxRating, AVG(Rating) AS AvgRating
FROM Enrollments
GROUP BY CourseID;

--10. Count how many students gave rating = 5.
SELECT COUNT(*) AS CountRating5
FROM Enrollments
WHERE Rating = 5;
```



Results Messages

	TotalStudents
1	3

1

	Enrollments
1	7

2

	CourseID	AvgRating
1	101	4
2	102	3
3	103	4
4	104	2

3

	InstructorID	TotalCourses
1	1	2
2	2	2

4

	categoryID	total_course
1	1	2
2	2	1
3	3	1

5

	CourseID	StudentsEnrolled
1	101	2
2	102	2
3	103	1
4	104	2

6

	CategoryID	AvgPrice
1	1	34.990000
2	2	49.990000
3	3	19.990000

7

	MaxPrice
1	49.99

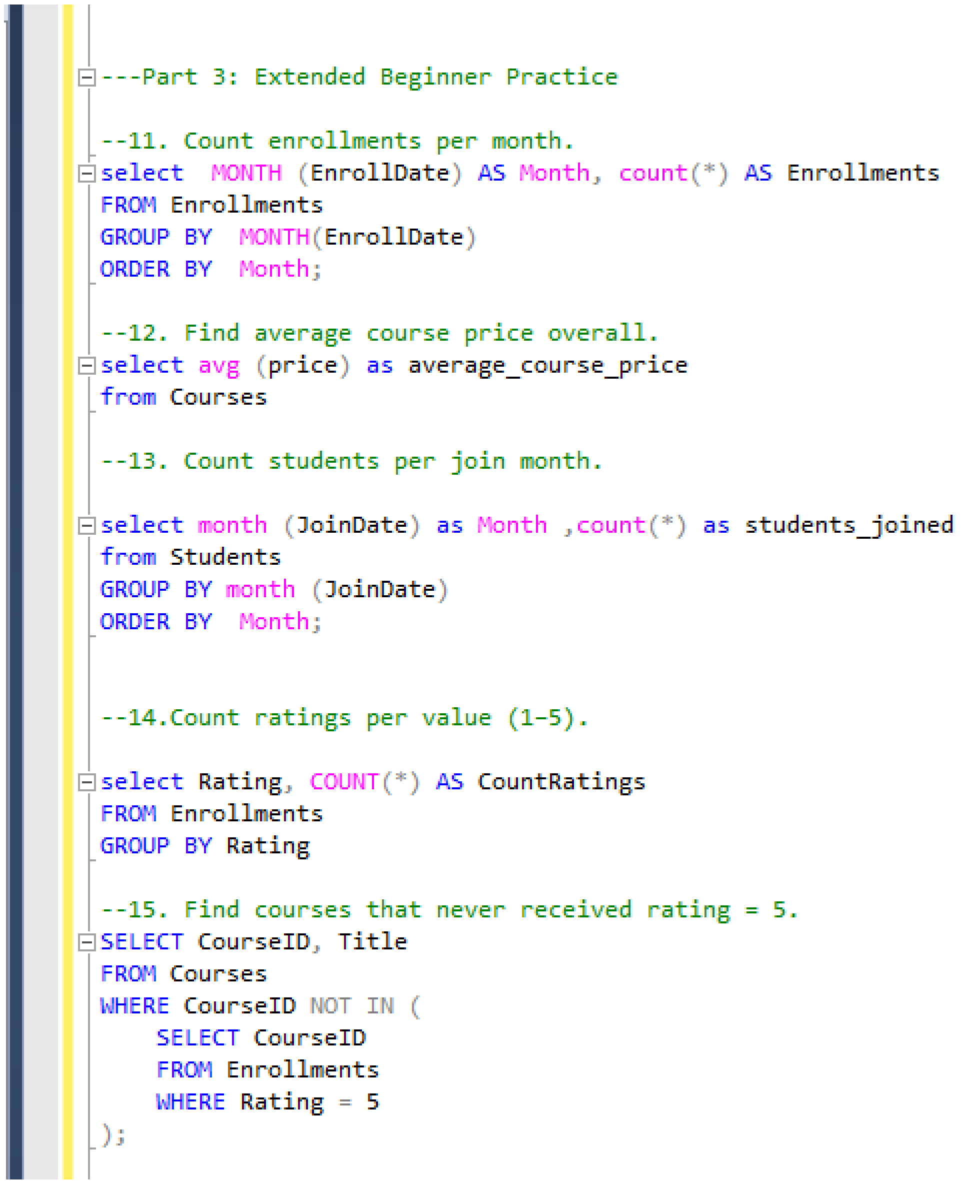
8

	CourseID	MinRating	MaxRating	AvgRating
1	101	4	5	4
2	102	3	4	3
3	103	4	4	4
4	104	2	3	2

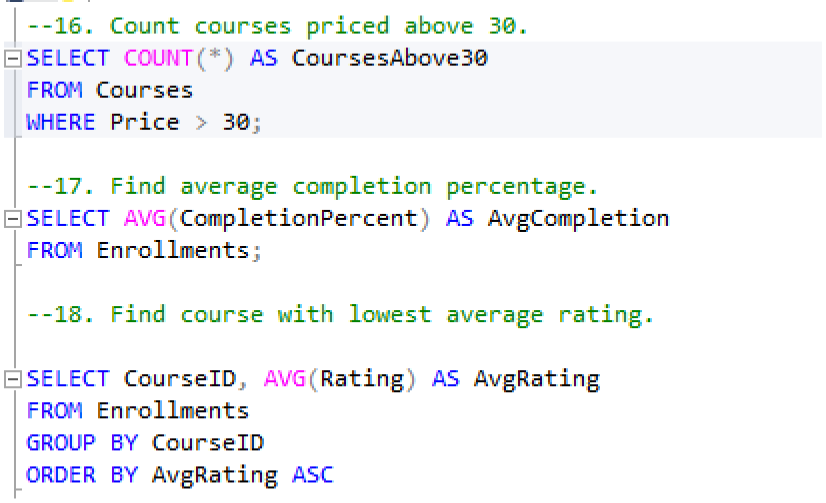
9

	CountRating5
1	1

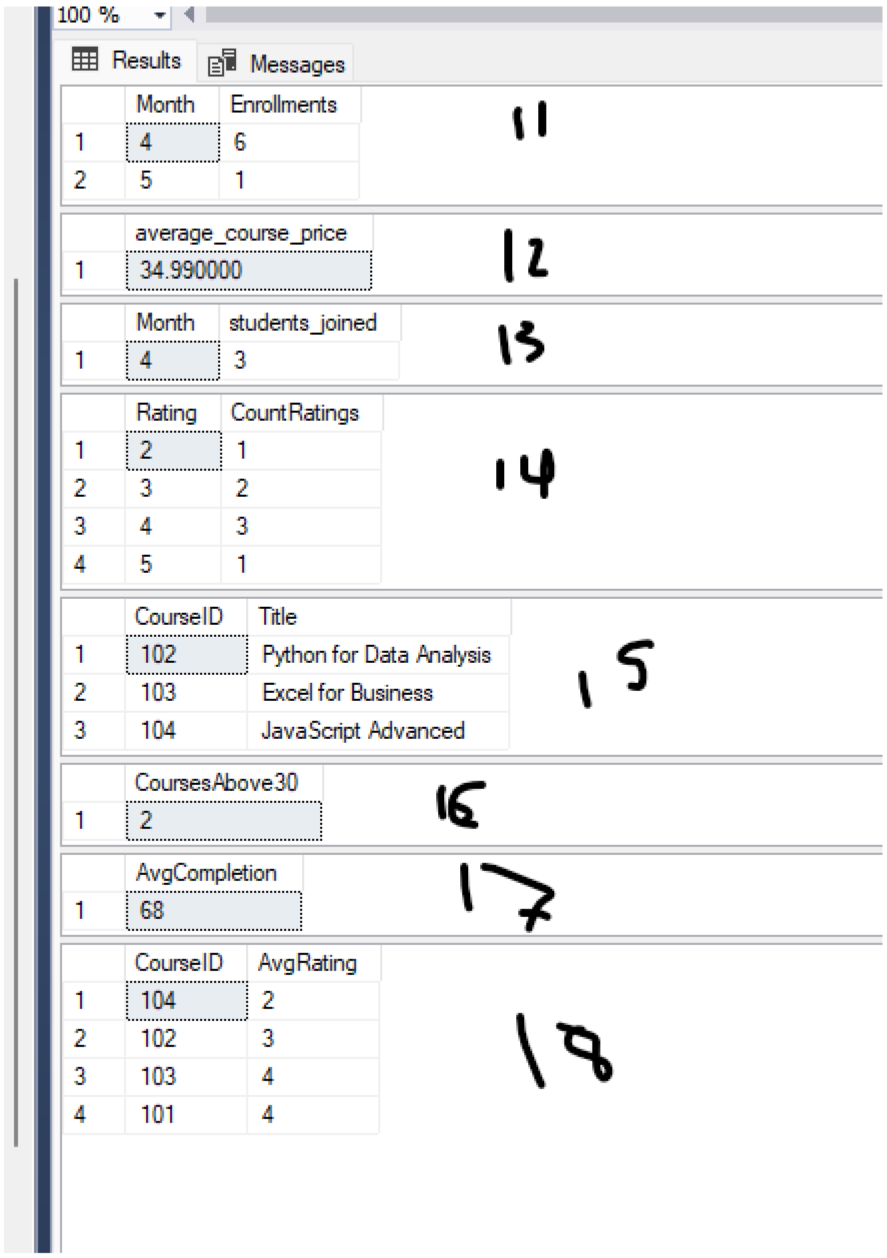
10



```sql
---Part 3: Extended Beginner Practice

--11. Count enrollments per month.
select  MONTH (EnrollDate) AS Month, count(*) AS Enrollments
FROM Enrollments
GROUP BY  MONTH(EnrollDate)
ORDER BY  Month;

--12. Find average course price overall.
select avg (price) as average_course_price
from Courses

--13. Count students per join month.

select month (JoinDate) as Month ,count(*) as students_joined
from Students
GROUP BY month (JoinDate)
ORDER BY  Month;


--14.Count ratings per value (1-5).

select Rating, COUNT(*) AS CountRatings
FROM Enrollments
GROUP BY Rating

--15. Find courses that never received rating = 5.
SELECT CourseID, Title
FROM Courses
WHERE CourseID NOT IN (
    SELECT CourseID
    FROM Enrollments
    WHERE Rating = 5
);
```



```sql
--16. Count courses priced above 30.
SELECT COUNT(*) AS CoursesAbove30
FROM Courses
WHERE Price > 30;

--17. Find average completion percentage.
SELECT AVG(CompletionPercent) AS AvgCompletion
FROM Enrollments;

--18. Find course with lowest average rating.

SELECT CourseID, AVG(Rating) AS AvgRating
FROM Enrollments
GROUP BY CourseID
ORDER BY AvgRating ASC
```



100 %

Results | Messages

	Month	Enrollments
1	4	6
2	5	1

11

	average_course_price
1	34.990000

12

	Month	students_joined
1	4	3

13

	Rating	CountRatings
1	2	1
2	3	2
3	4	3
4	5	1

14

	CourseID	Title
1	102	Python for Data Analysis
2	103	Excel for Business
3	104	JavaScript Advanced

15

	CoursesAbove30
1	2

16

	AvgCompletion
1	68

17

	CourseID	AvgRating
1	104	2
2	102	3
3	103	4
4	101	4

18

Reflection:

Answer briefly:

• What was the easiest?

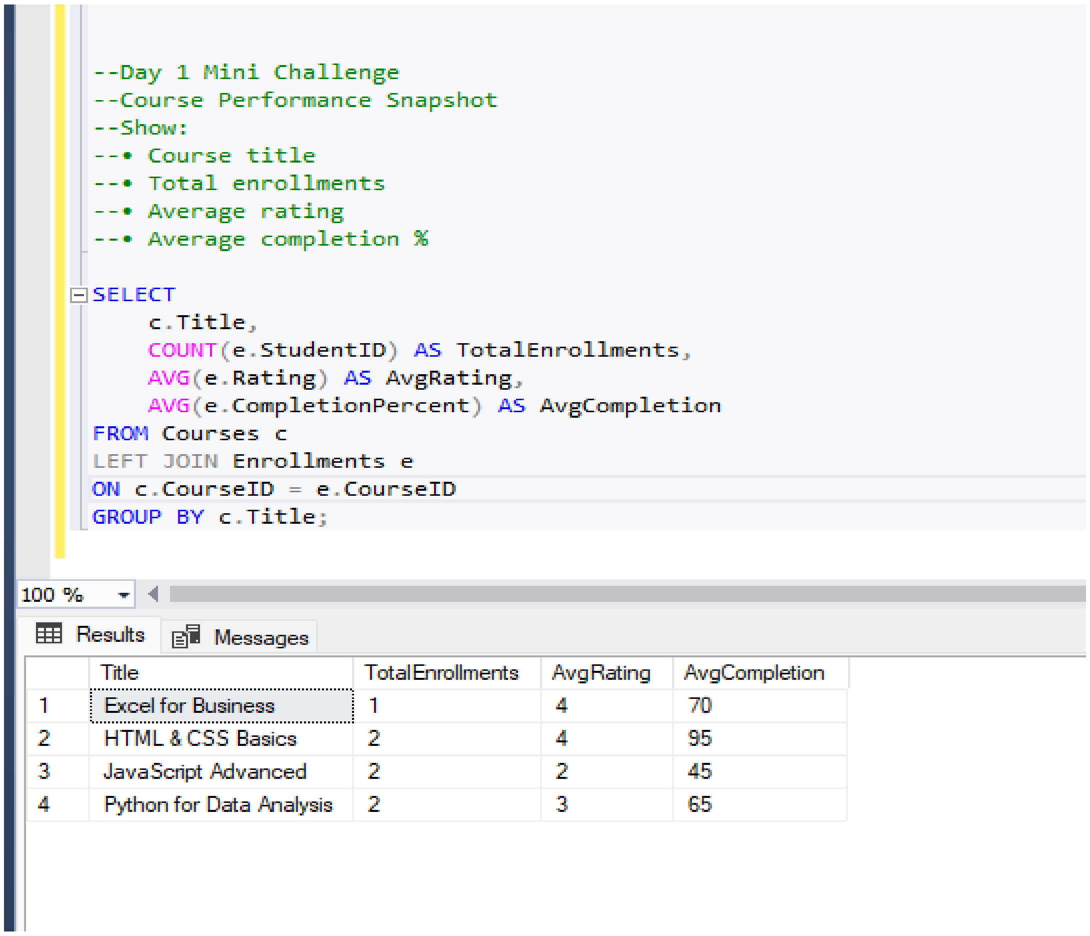
basic aggregation function:(count,avg,max,min).

• What was the hardest?

"Group by" is little bit hard, it needs more work to understand it well.

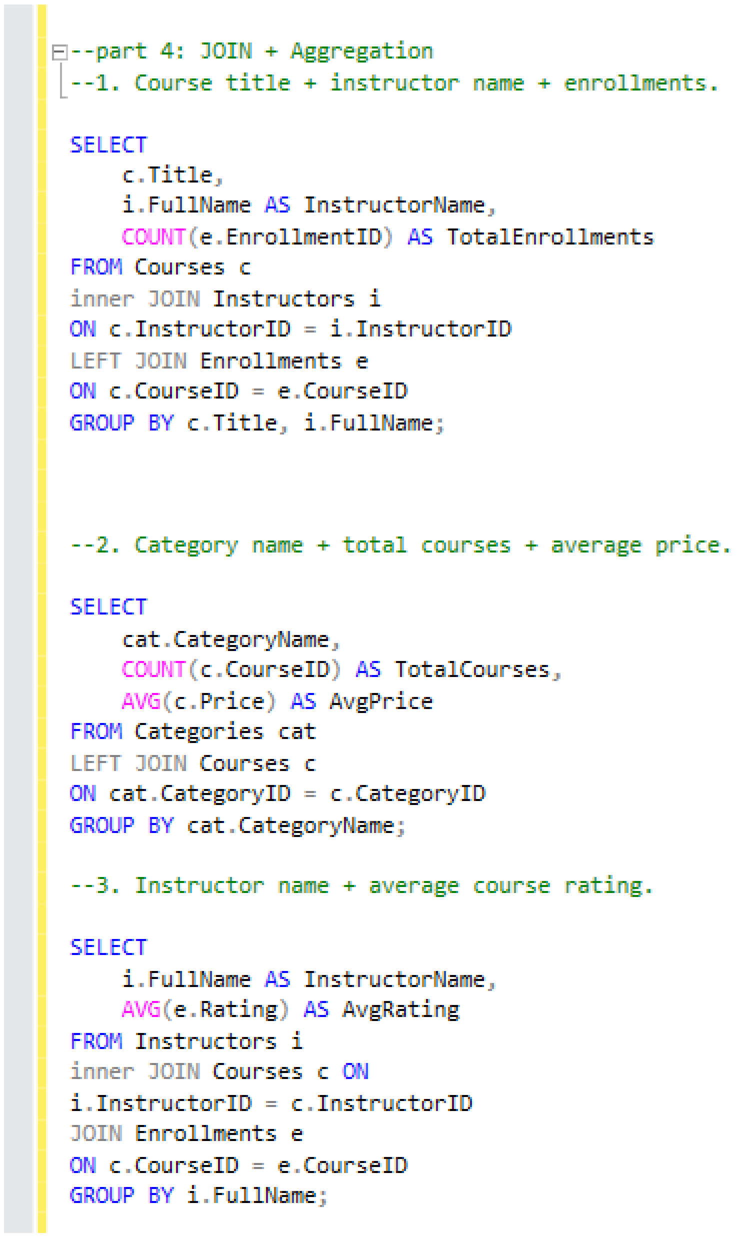
• What does GROUP BY do in your own words?

Collects rows that share the same value into groups and allows aggregation functions to calculate results.

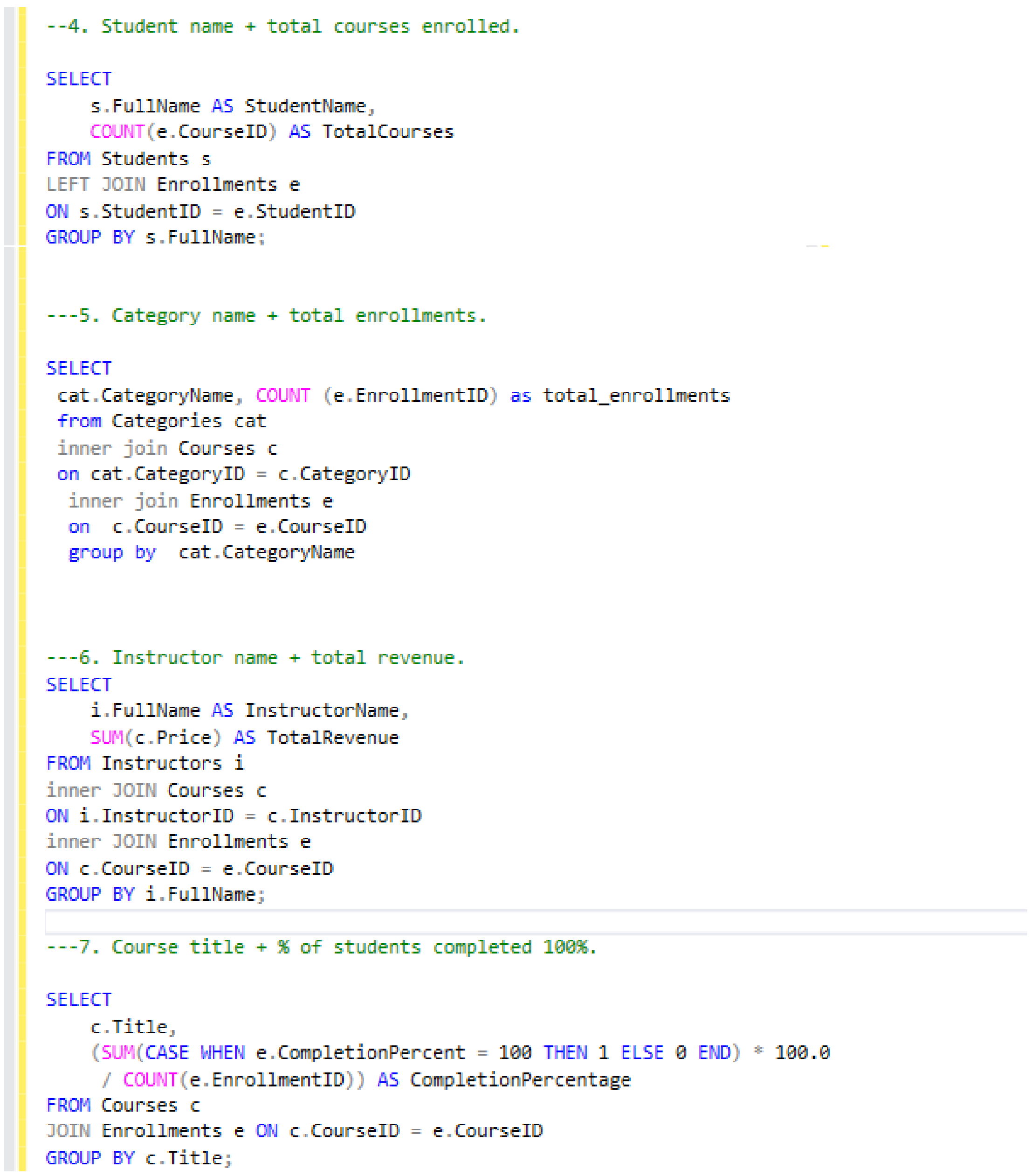


```
--Day 1 Mini Challenge
--Course Performance Snapshot
--Show:
--• Course title
--• Total enrollments
--• Average rating
--• Average completion %

SELECT
    c.Title,
    COUNT(e.StudentID) AS TotalEnrollments,
    AVG(e.Rating) AS AvgRating,
    AVG(e.CompletionPercent) AS AvgCompletion
FROM Courses c
LEFT JOIN Enrollments e
ON c.CourseID = e.CourseID
GROUP BY c.Title;
```

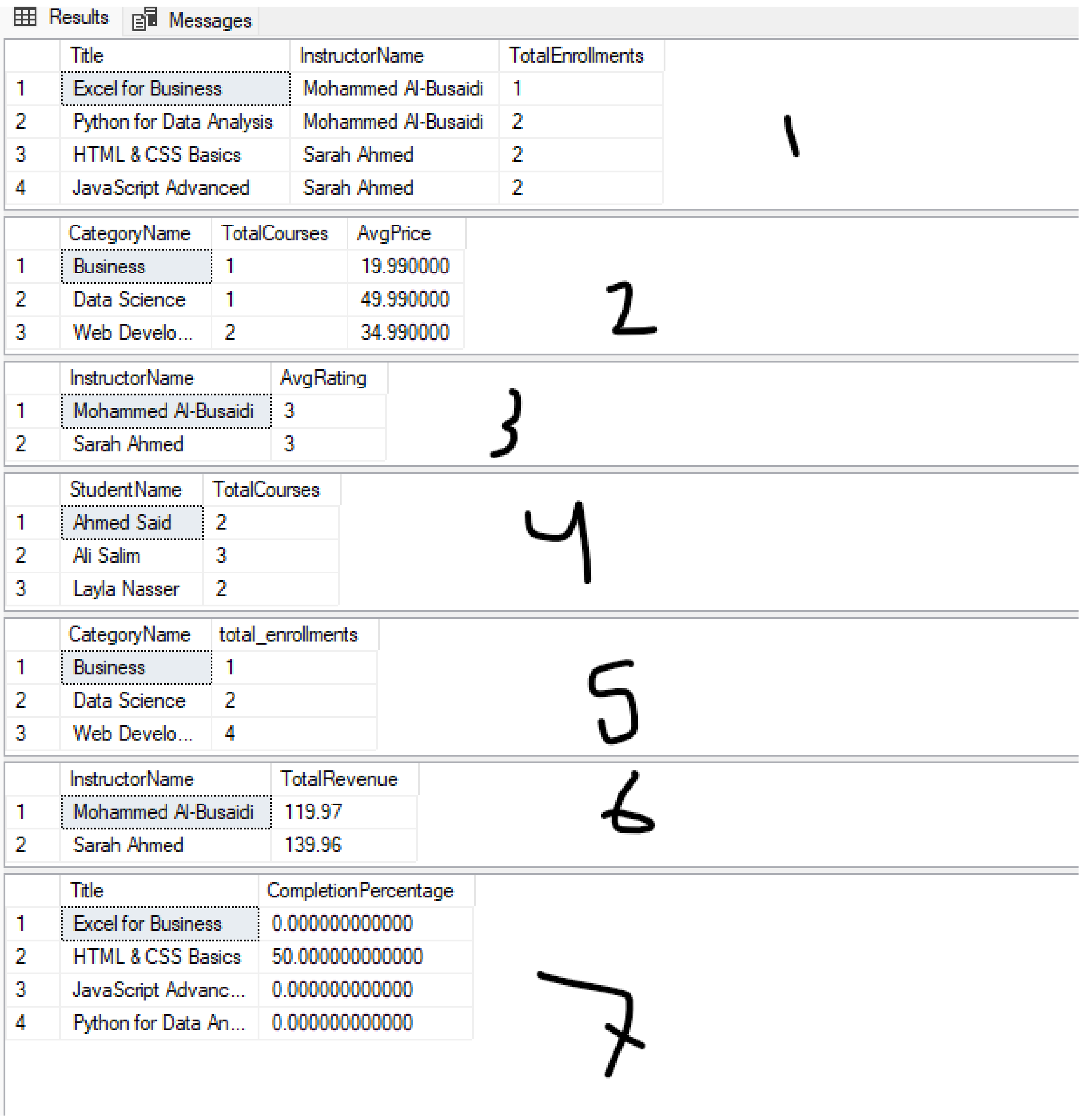
	Title	TotalEnrollments	AvgRating	AvgCompletion
1	Excel for Business	1	4	70
2	HTML & CSS Basics	2	4	95
3	JavaScript Advanced	2	2	45
4	Python for Data Analysis	2	3	65



```sql
--part 4: JOIN + Aggregation
--1. Course title + instructor name + enrollments.

SELECT
    c.Title,
    i.FullName AS InstructorName,
    COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Courses c
inner JOIN Instructors i
ON c.InstructorID = i.InstructorID
LEFT JOIN Enrollments e
ON c.CourseID = e.CourseID
GROUP BY c.Title, i.FullName;



--2. Category name + total courses + average price.

SELECT
    cat.CategoryName,
    COUNT(c.CourseID) AS TotalCourses,
    AVG(c.Price) AS AvgPrice
FROM Categories cat
LEFT JOIN Courses c
ON cat.CategoryID = c.CategoryID
GROUP BY cat.CategoryName;

--3. Instructor name + average course rating.

SELECT
    i.FullName AS InstructorName,
    AVG(e.Rating) AS AvgRating
FROM Instructors i
inner JOIN Courses c ON
i.InstructorID = c.InstructorID
JOIN Enrollments e
ON c.CourseID = e.CourseID
GROUP BY i.FullName;
```



```sql
--4. Student name + total courses enrolled.

SELECT
    s.FullName AS StudentName,
    COUNT(e.CourseID) AS TotalCourses
FROM Students s
LEFT JOIN Enrollments e
ON s.StudentID = e.StudentID
GROUP BY s.FullName;


---5. Category name + total enrollments.

SELECT
 cat.CategoryName, COUNT (e.EnrollmentID) as total_enrollments
 from Categories cat
 inner join Courses c
 on cat.CategoryID = c.CategoryID
  inner join Enrollments e
  on  c.CourseID = e.CourseID
  group by  cat.CategoryName



---6. Instructor name + total revenue.
SELECT
    i.FullName AS InstructorName,
    SUM(c.Price) AS TotalRevenue
FROM Instructors i
inner JOIN Courses c
ON i.InstructorID = c.InstructorID
inner JOIN Enrollments e
ON c.CourseID = e.CourseID
GROUP BY i.FullName;

---7. Course title + % of students completed 100%.

SELECT
    c.Title,
    (SUM(CASE WHEN e.CompletionPercent = 100 THEN 1 ELSE 0 END) * 100.0
     / COUNT(e.EnrollmentID)) AS CompletionPercentage
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
```



	Title	InstructorName	TotalEnrollments
1	Excel for Business	Mohammed Al-Busaidi	1
2	Python for Data Analysis	Mohammed Al-Busaidi	2
3	HTML & CSS Basics	Sarah Ahmed	2
4	JavaScript Advanced	Sarah Ahmed	2

1

	CategoryName	TotalCourses	AvgPrice
1	Business	1	19.990000
2	Data Science	1	49.990000
3	Web Develo...	2	34.990000

2

	InstructorName	AvgRating
1	Mohammed Al-Busaidi	3
2	Sarah Ahmed	3

3

	StudentName	TotalCourses
1	Ahmed Said	2
2	Ali Salim	3
3	Layla Nasser	2

4

	CategoryName	total_enrollments
1	Business	1
2	Data Science	2
3	Web Develo...	4

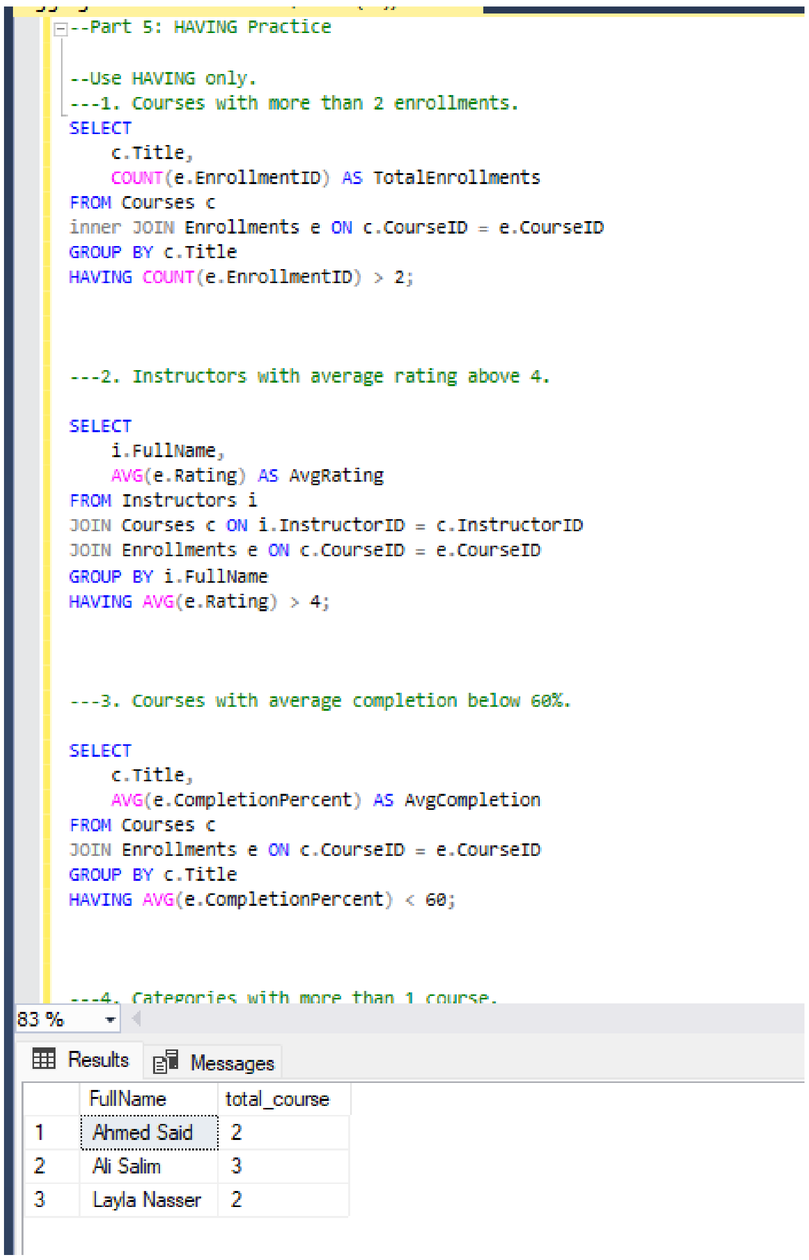
5

	InstructorName	TotalRevenue
1	Mohammed Al-Busaidi	119.97
2	Sarah Ahmed	139.96

6

	Title	CompletionPercentage
1	Excel for Business	0.000000000000
2	HTML & CSS Basics	50.000000000000
3	JavaScript Advanc...	0.000000000000
4	Python for Data An...	0.000000000000

7

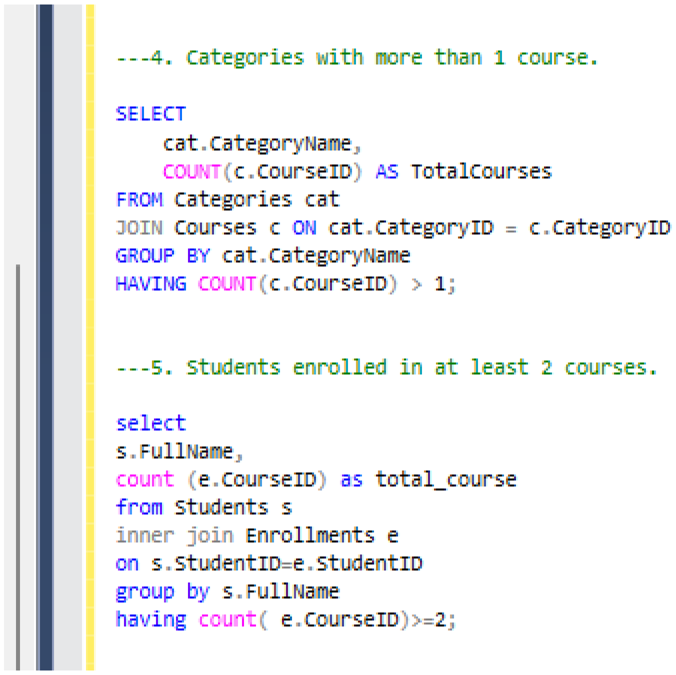


```
--Part 5: HAVING Practice

--Use HAVING only.
---1. Courses with more than 2 enrollments.
SELECT
    c.Title,
    COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Courses c
inner JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
HAVING COUNT(e.EnrollmentID) > 2;



---2. Instructors with average rating above 4.

SELECT
    i.FullName,
    AVG(e.Rating) AS AvgRating
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName
HAVING AVG(e.Rating) > 4;



---3. Courses with average completion below 60%.

SELECT
    c.Title,
    AVG(e.CompletionPercent) AS AvgCompletion
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
HAVING AVG(e.CompletionPercent) < 60;



---4. Categories with more than 1 course.
```

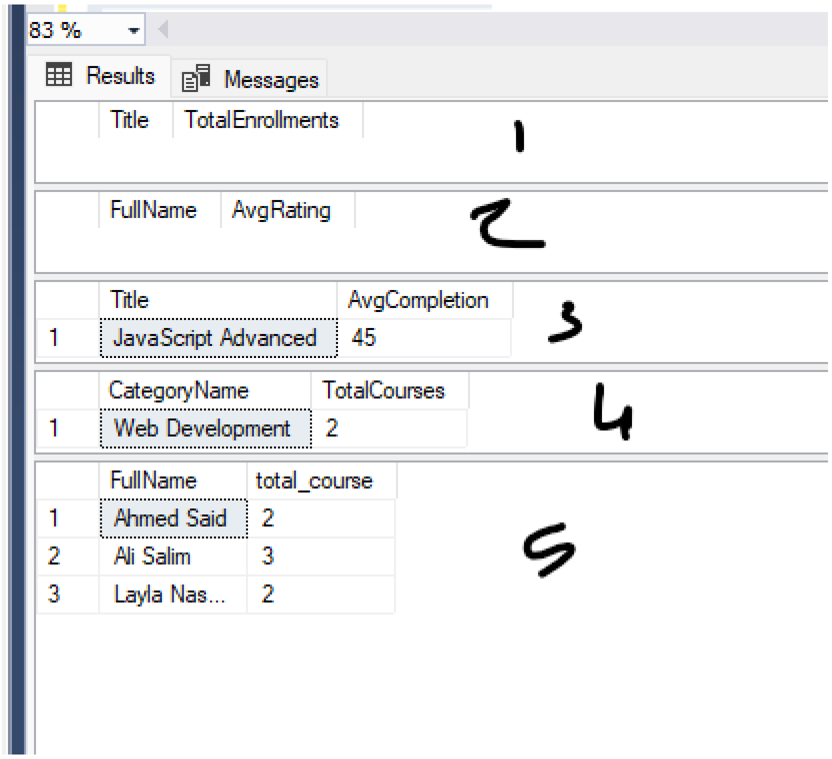
83 %

Results | Messages

	FullName	total_course
1	Ahmed Said	2
2	Ali Salim	3
3	Layla Nasser	2



```sql
---4. Categories with more than 1 course.

SELECT
    cat.CategoryName,
    COUNT(c.CourseID) AS TotalCourses
FROM Categories cat
JOIN Courses c ON cat.CategoryID = c.CategoryID
GROUP BY cat.CategoryName
HAVING COUNT(c.CourseID) > 1;


---5. Students enrolled in at least 2 courses.

select
s.FullName,
count (e.CourseID) as total_course
from Students s
inner join Enrollments e
on s.StudentID=e.StudentID
group by s.FullName
having count( e.CourseID)>=2;
```



83 %

Results | Messages

	Title	TotalEnrollments

1

	FullName	AvgRating

2

	Title	AvgCompletion
1	JavaScript Advanced	45

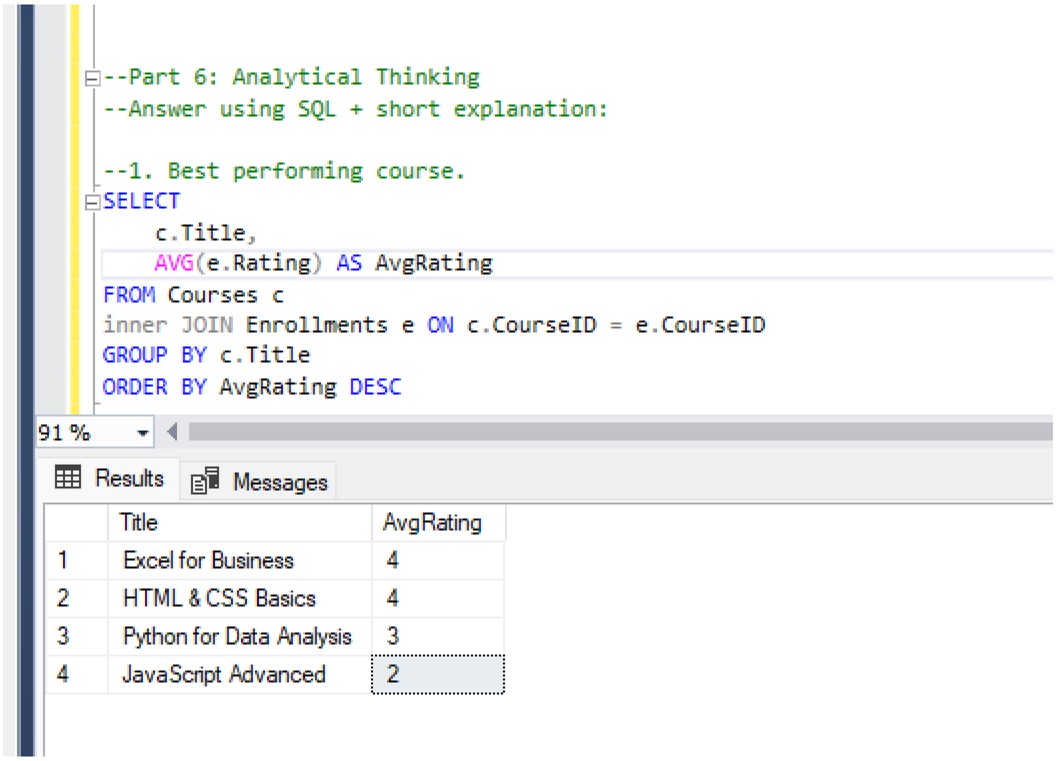
3

	CategoryName	TotalCourses
1	Web Development	2

4

	FullName	total_course
1	Ahmed Said	2
2	Ali Salim	3
3	Layla Nas...	2

5



```sql
--Part 6: Analytical Thinking
--Answer using SQL + short explanation:

--1. Best performing course.
SELECT
    c.Title,
    AVG(e.Rating) AS AvgRating
FROM Courses c
inner JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY AvgRating DESC
```

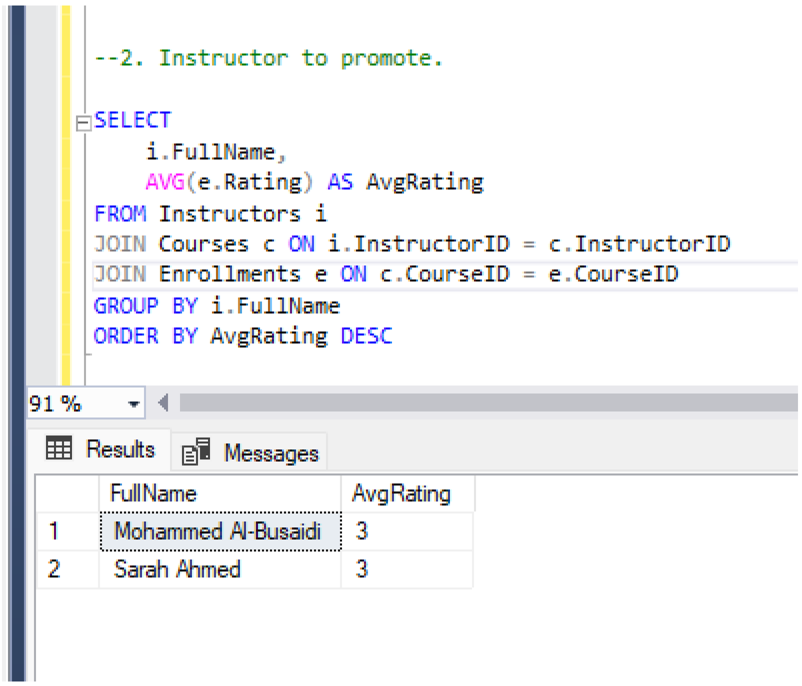
91 %

Results Messages

	Title	AvgRating
1	Excel for Business	4
2	HTML & CSS Basics	4
3	Python for Data Analysis	3
4	JavaScript Advanced	2

Explanation:

1-The best performing course is the one with the highest average student rating, so from the above result the Excel for Business and HTML&CSS Basics has the best performing course.



```sql
--2. Instructor to promote.

SELECT
    i.FullName,
    AVG(e.Rating) AS AvgRating
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName
ORDER BY AvgRating DESC
```

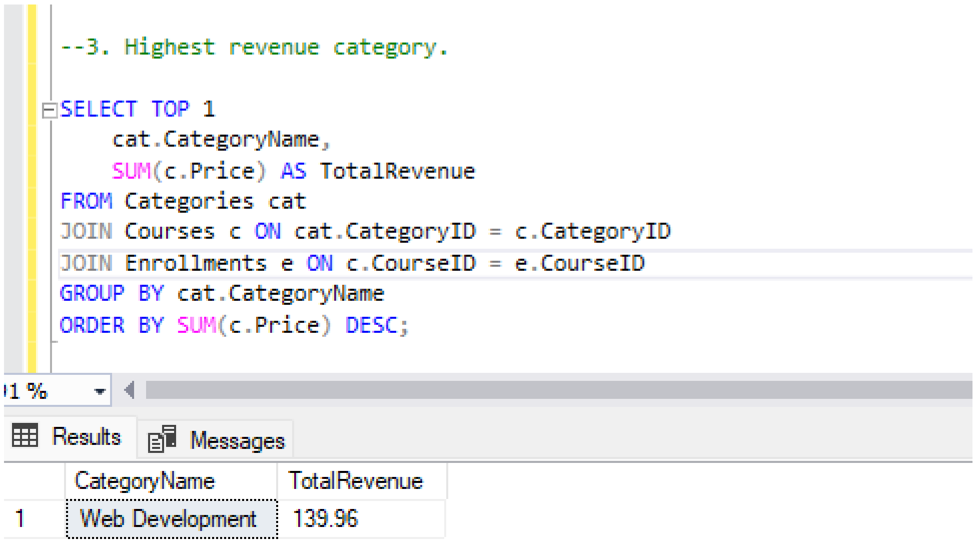
91 %

Results | Messages

	FullName	AvgRating
1	Mohammed Al-Busaidi	3
2	Sarah Ahmed	3

Explanation:

2-The instructor with the highest average course rating should be promoted because their courses perform best overall. so, Mohammed AL-Busaidi and Sarah Ahmed, they should be promoted.

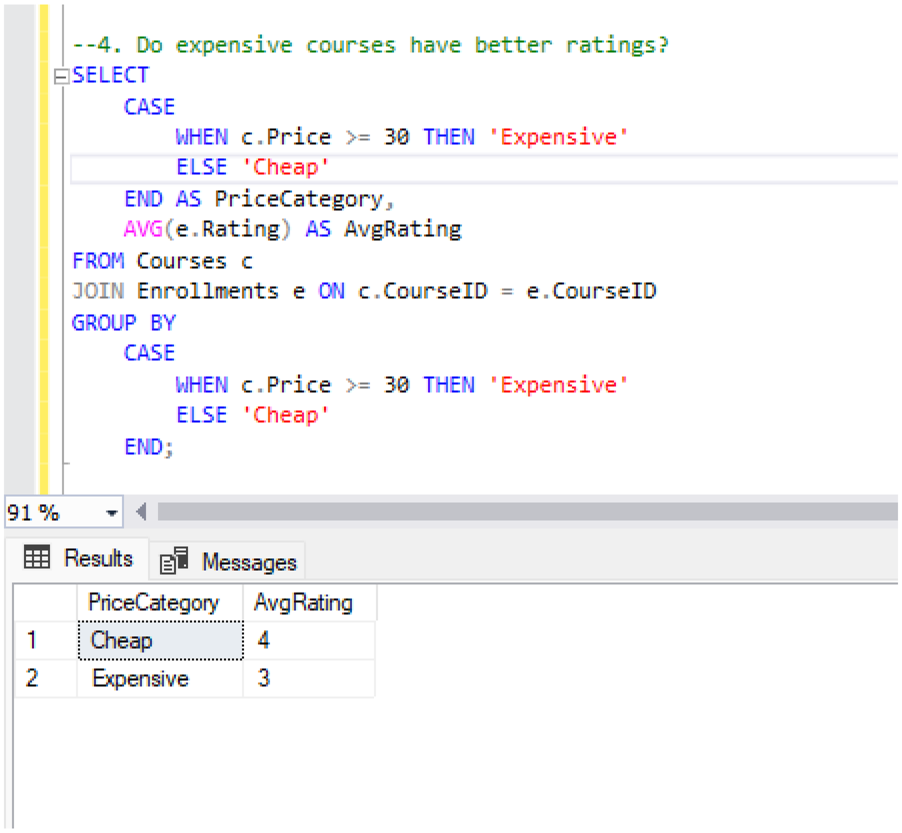


```
--3. Highest revenue category.

SELECT TOP 1
    cat.CategoryName,
    SUM(c.Price) AS TotalRevenue
FROM Categories cat
JOIN Courses c ON cat.CategoryID = c.CategoryID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY cat.CategoryName
ORDER BY SUM(c.Price) DESC;
```

	CategoryName	TotalRevenue
1	Web Development	139.96

Explanation:

3-The highest revenue category is the one that generates the largest total income from course prices and enrollments. So, Web Development has the largest income.

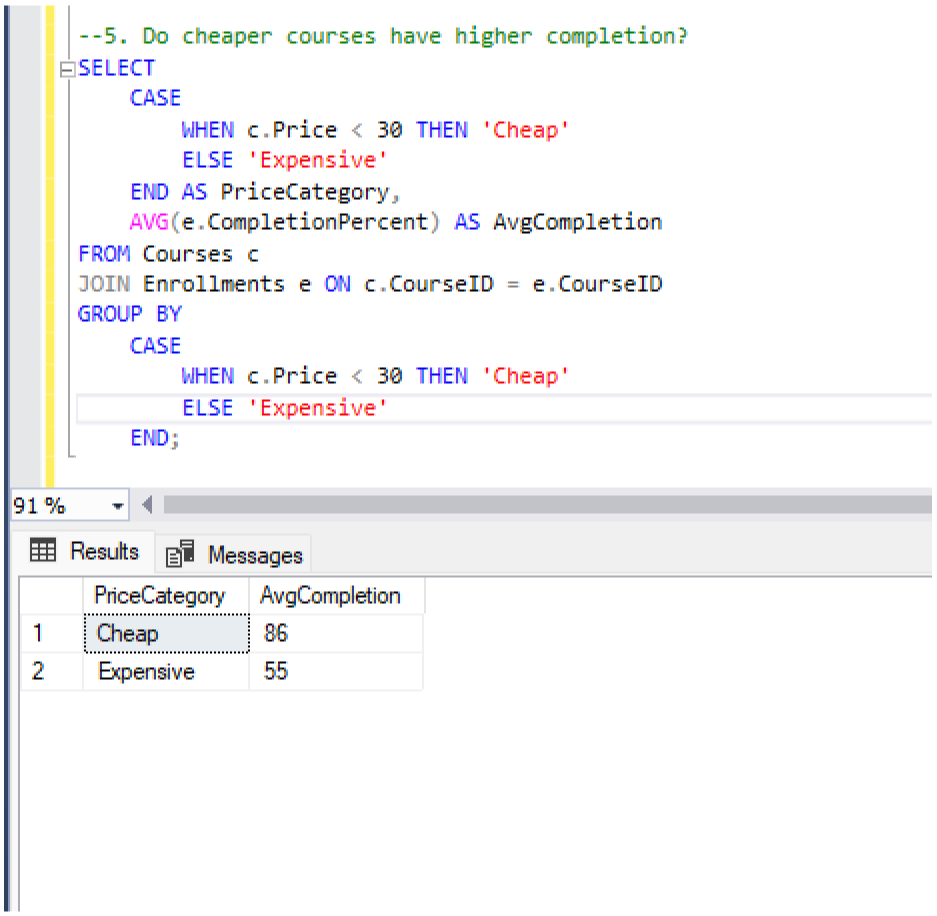


```sql
--4. Do expensive courses have better ratings?
SELECT
    CASE
        WHEN c.Price >= 30 THEN 'Expensive'
        ELSE 'Cheap'
    END AS PriceCategory,
    AVG(e.Rating) AS AvgRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY
    CASE
        WHEN c.Price >= 30 THEN 'Expensive'
        ELSE 'Cheap'
    END;
```

	PriceCategory	AvgRating
1	Cheap	4
2	Expensive	3

Explanation:

4-This query compares the average ratings of expensive and cheap courses to determine whether higher-priced courses receive better ratings, cheaper than the Avg Rating =4 and Expensive=3.

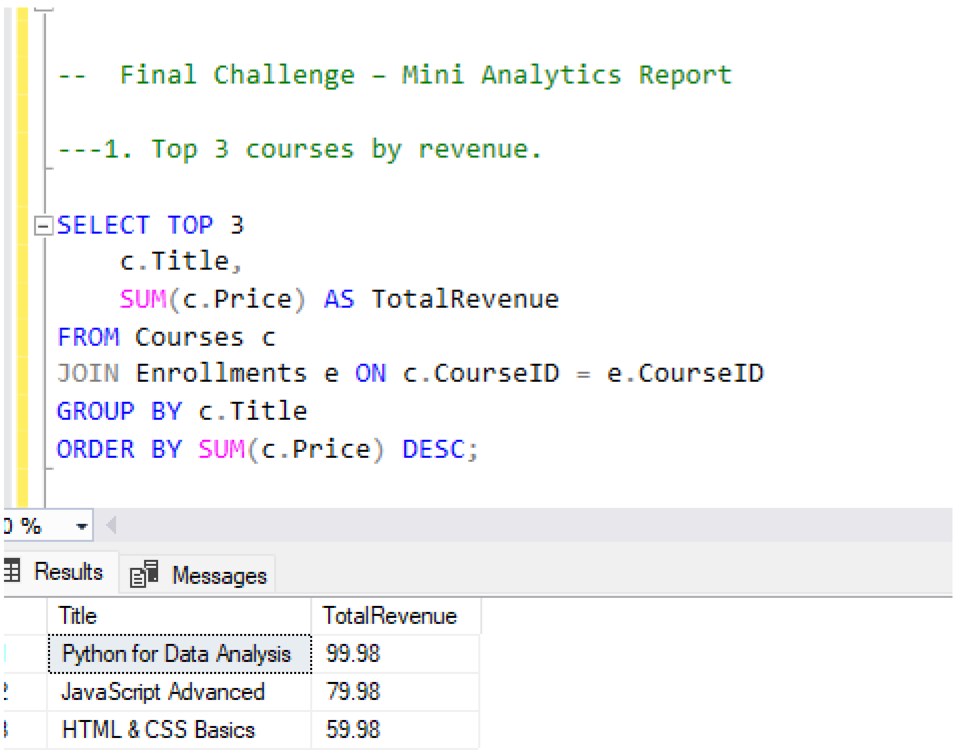


```sql
--5. Do cheaper courses have higher completion?
SELECT
    CASE
        WHEN c.Price < 30 THEN 'Cheap'
        ELSE 'Expensive'
    END AS PriceCategory,
    AVG(e.CompletionPercent) AS AvgCompletion
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY
    CASE
        WHEN c.Price < 30 THEN 'Cheap'
        ELSE 'Expensive'
    END;
```

	PriceCategory	AvgCompletion
1	Cheap	86
2	Expensive	55

Explanation:

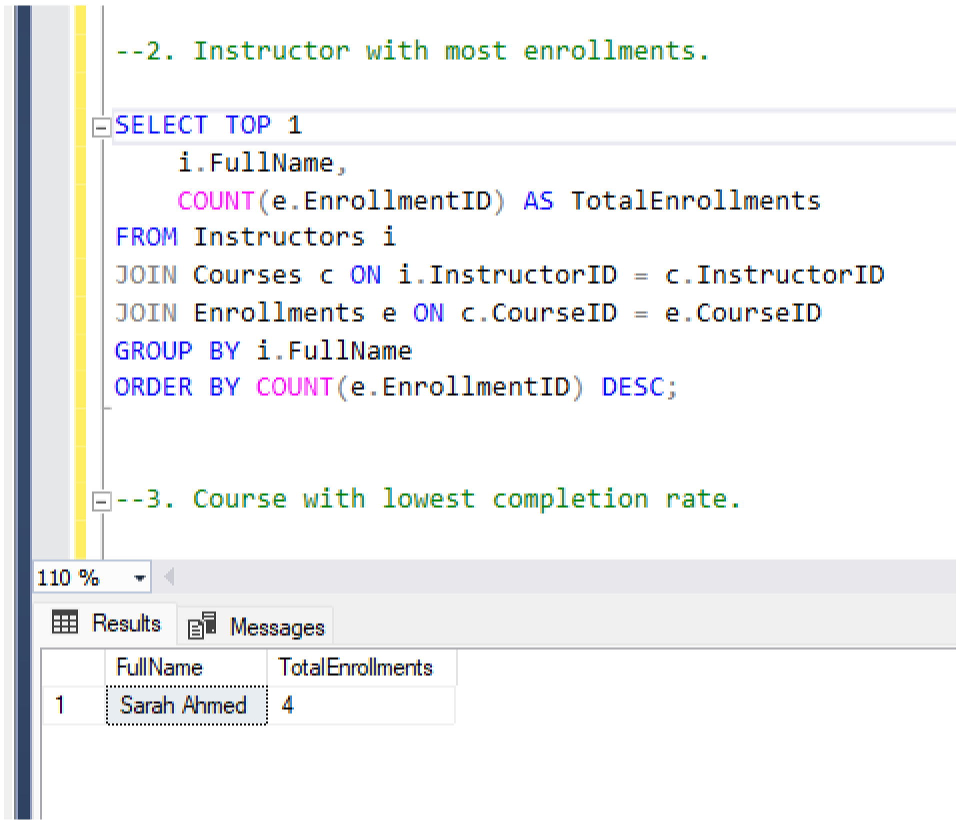
This query compares average completion percentages between cheaper and more expensive courses to see which price range students complete more often.



```
--  Final Challenge – Mini Analytics Report

---1. Top 3 courses by revenue.

SELECT TOP 3
    c.Title,
    SUM(c.Price) AS TotalRevenue
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY SUM(c.Price) DESC;
```

	Title	TotalRevenue
1	Python for Data Analysis	99.98
2	JavaScript Advanced	79.98
3	HTML & CSS Basics	59.98

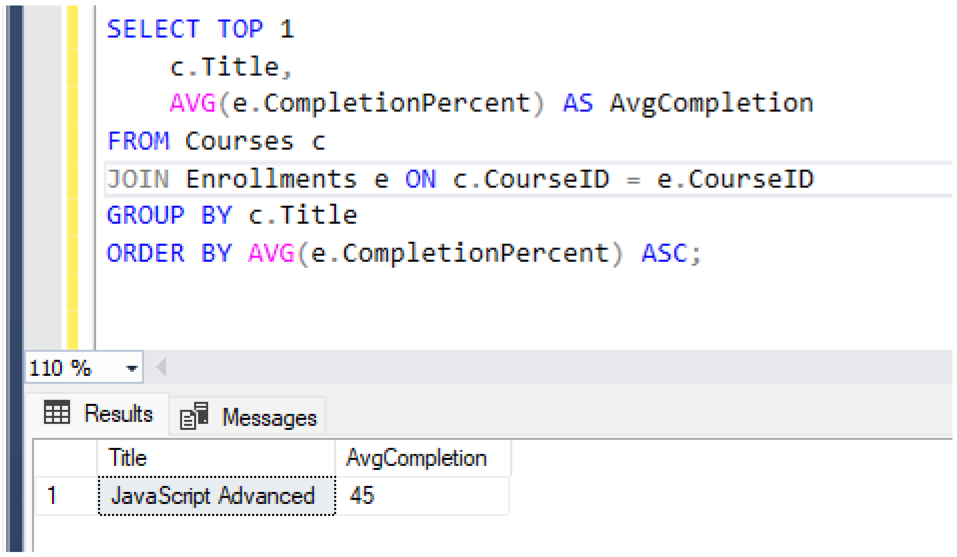
Explanation:
Shows the three courses that generate the highest total revenue from enrollments.



```sql
--2. Instructor with most enrollments.

SELECT TOP 1
    i.FullName,
    COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName
ORDER BY COUNT(e.EnrollmentID) DESC;


--3. Course with lowest completion rate.
```

	FullName	TotalEnrollments
1	Sarah Ahmed	4

Explanation:
Identifies the instructor whose courses have the highest total number of student enrollments.



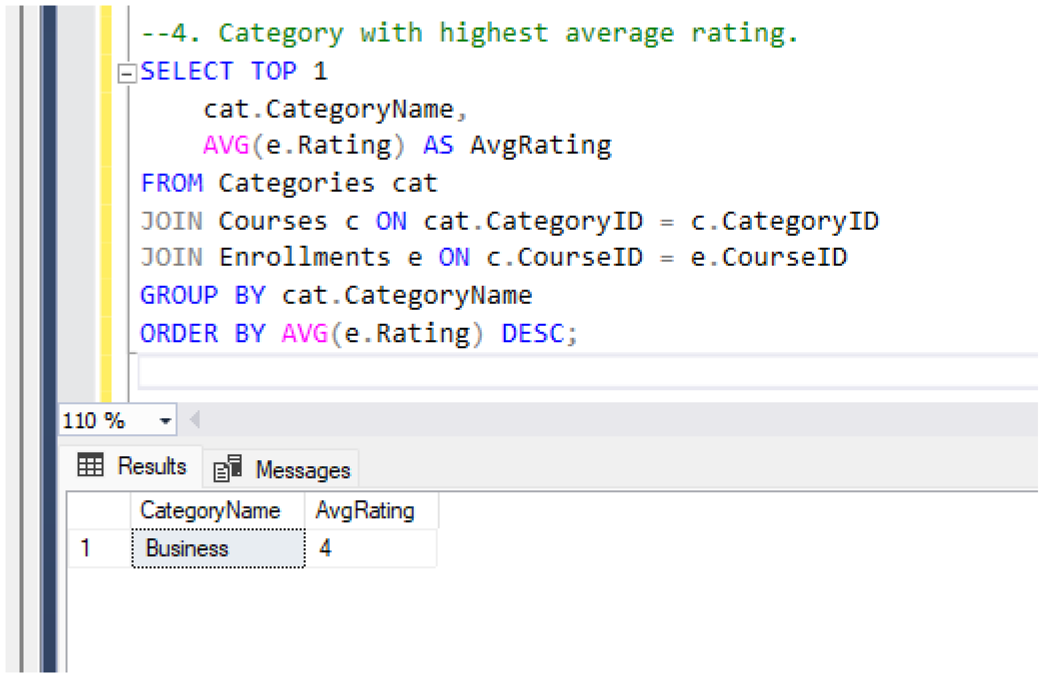
```sql
SELECT TOP 1
    c.Title,
    AVG(e.CompletionPercent) AS AvgCompletion
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY AVG(e.CompletionPercent) ASC;
```

110 %

Results | Messages

	Title	AvgCompletion
1	JavaScript Advanced	45

Explanation:
Finds the course with the lowest average completion percentage, indicating low student engagement.



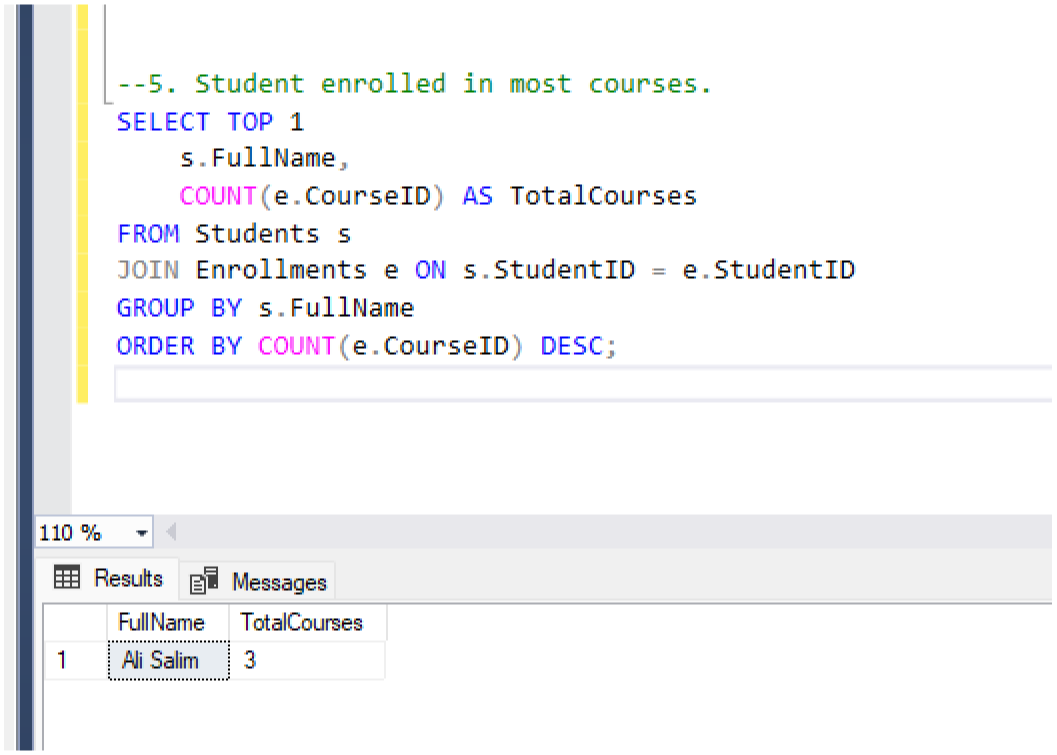
```sql
--4. Category with highest average rating.
SELECT TOP 1
    cat.CategoryName,
    AVG(e.Rating) AS AvgRating
FROM Categories cat
JOIN Courses c ON cat.CategoryID = c.CategoryID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY cat.CategoryName
ORDER BY AVG(e.Rating) DESC;
```

110 %

Results | Messages

	CategoryName	AvgRating
1	Business	4

Explanation:
Shows the category whose courses receive the highest average student ratings.



```sql
--5. Student enrolled in most courses.
SELECT TOP 1
    s.FullName,
    COUNT(e.CourseID) AS TotalCourses
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName
ORDER BY COUNT(e.CourseID) DESC;
```

	FullName	TotalCourses
1	Ali Salim	3

Explanation:

Identifies the student who enrolled in the largest number of courses.