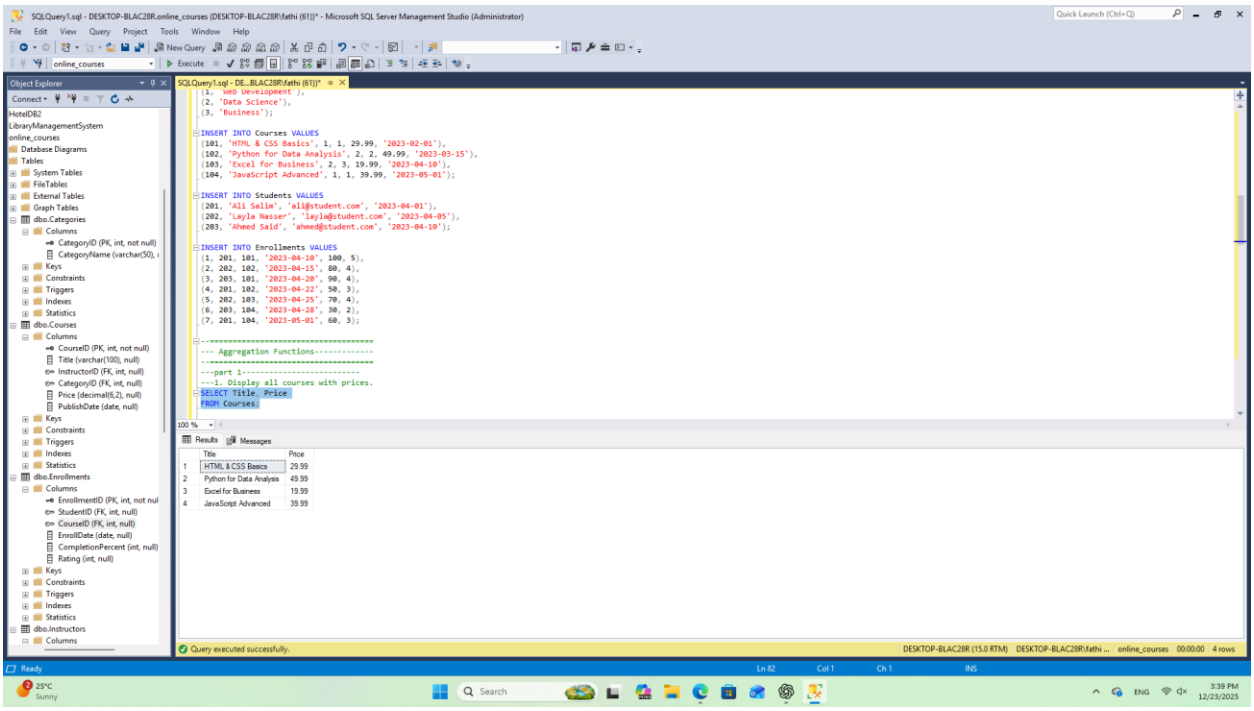
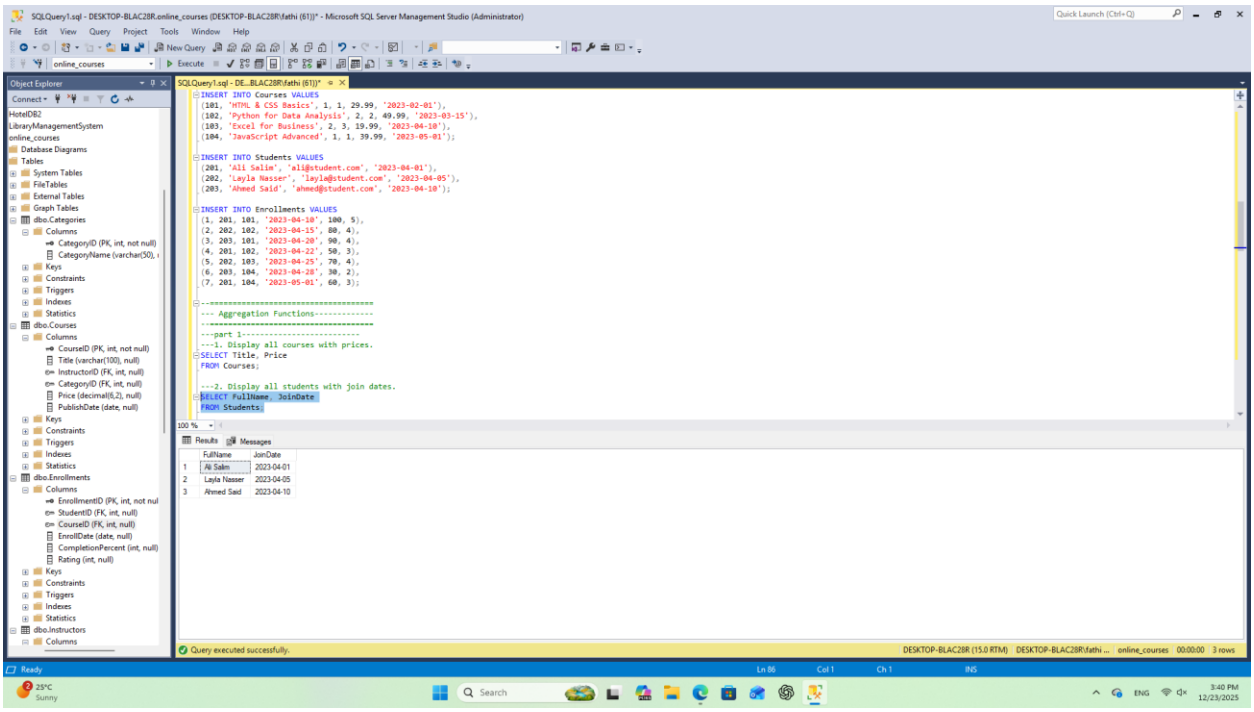

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
Object Explorer
SQLQuery1.sql - DE...BLAC28R\fathi (61))*
(1, 'Web Development'),
(2, 'Data Science'),
(3, 'Business');
INSERT INTO Courses VALUES
(101, 'HTML & CSS Basics', 1, 1, 29.99, '2023-02-01'),
(102, 'Python for Data Analysis', 2, 2, 49.99, '2023-03-15'),
(103, 'Excel for Business', 2, 3, 19.99, '2023-04-10'),
(104, 'JavaScript Advanced', 1, 1, 39.99, '2023-05-01');
INSERT INTO Students VALUES
(201, 'Ali Salim', 'ali@student.com', '2023-04-01'),
(202, 'Layla Nasser', 'layla@student.com', '2023-04-05'),
(203, 'Ahmed Said', 'ahmed@student.com', '2023-04-10');
INSERT INTO Enrollments VALUES
(1, 201, 101, '2023-04-10', 100, 5),
(2, 202, 102, '2023-04-15', 80, 4),
(3, 203, 101, '2023-04-20', 90, 4),
(4, 201, 102, '2023-04-22', 50, 3),
(5, 202, 103, '2023-04-25', 70, 4),
(6, 203, 104, '2023-04-28', 30, 2),
(7, 201, 104, '2023-05-01', 60, 3);

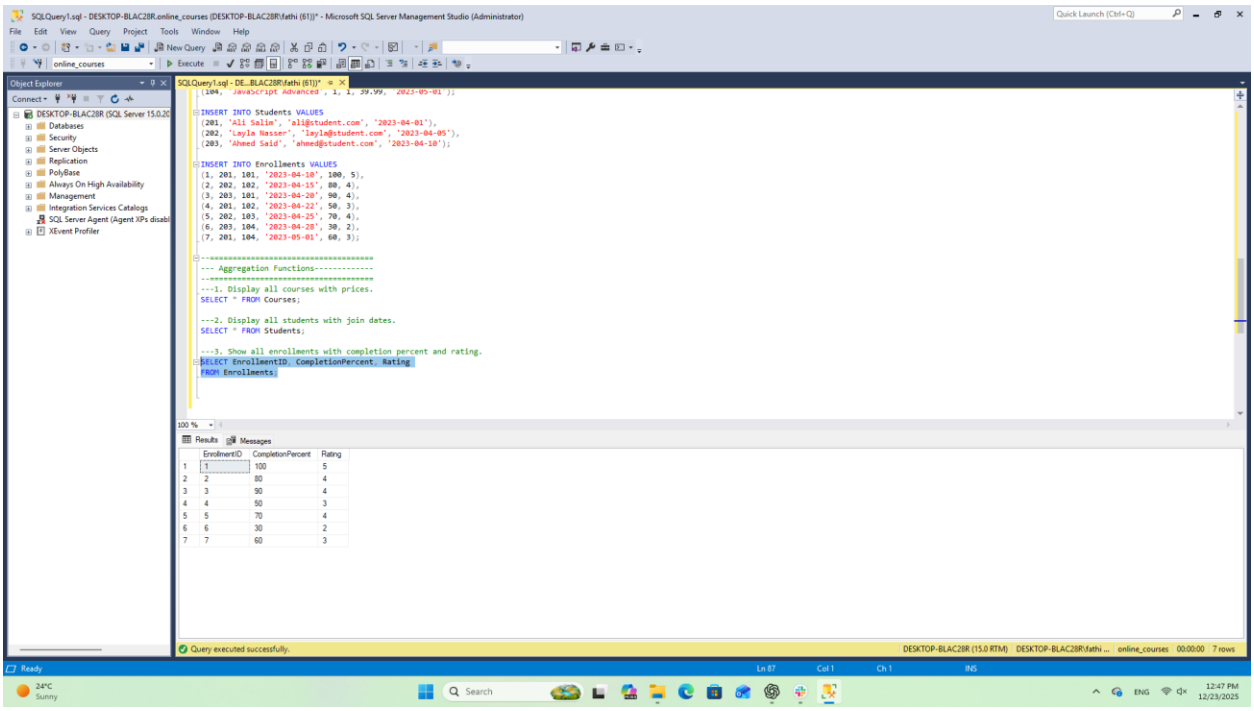
--- Aggregation Functions------------

---part 1----------------------
---1. Display all courses with prices.
SELECT Title, Price
FROM Courses;
Results
Messages
Title
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
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
online_courses
00:00:00
4 rows
Ready
Ln 82
Col 1
Ch 1
INS
3:39 PM
12/23/2025


SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
Object Explorer
SQLQuery1.sql - DE...BLAC28R\fathi (61))*
INSERT INTO Courses VALUES
(101, 'HTML & CSS Basics', 1, 1, 29.99, '2023-02-01'),
(102, 'Python for Data Analysis', 2, 2, 49.99, '2023-03-15'),
(103, 'Excel for Business', 2, 3, 19.99, '2023-04-10'),
(104, 'JavaScript Advanced', 1, 1, 39.99, '2023-05-01');
INSERT INTO Students VALUES
(201, 'Ali Salim', 'ali@student.com', '2023-04-01'),
(202, 'Layla Nasser', 'layla@student.com', '2023-04-05'),
(203, 'Ahmed Said', 'ahmed@student.com', '2023-04-10');
INSERT INTO Enrollments VALUES
(1, 201, 101, '2023-04-10', 100, 5),
(2, 202, 102, '2023-04-15', 80, 4),
(3, 203, 101, '2023-04-20', 90, 4),
(4, 201, 102, '2023-04-22', 50, 3),
(5, 202, 103, '2023-04-25', 70, 4),
(6, 203, 104, '2023-04-28', 30, 2),
(7, 201, 104, '2023-05-01', 60, 3);

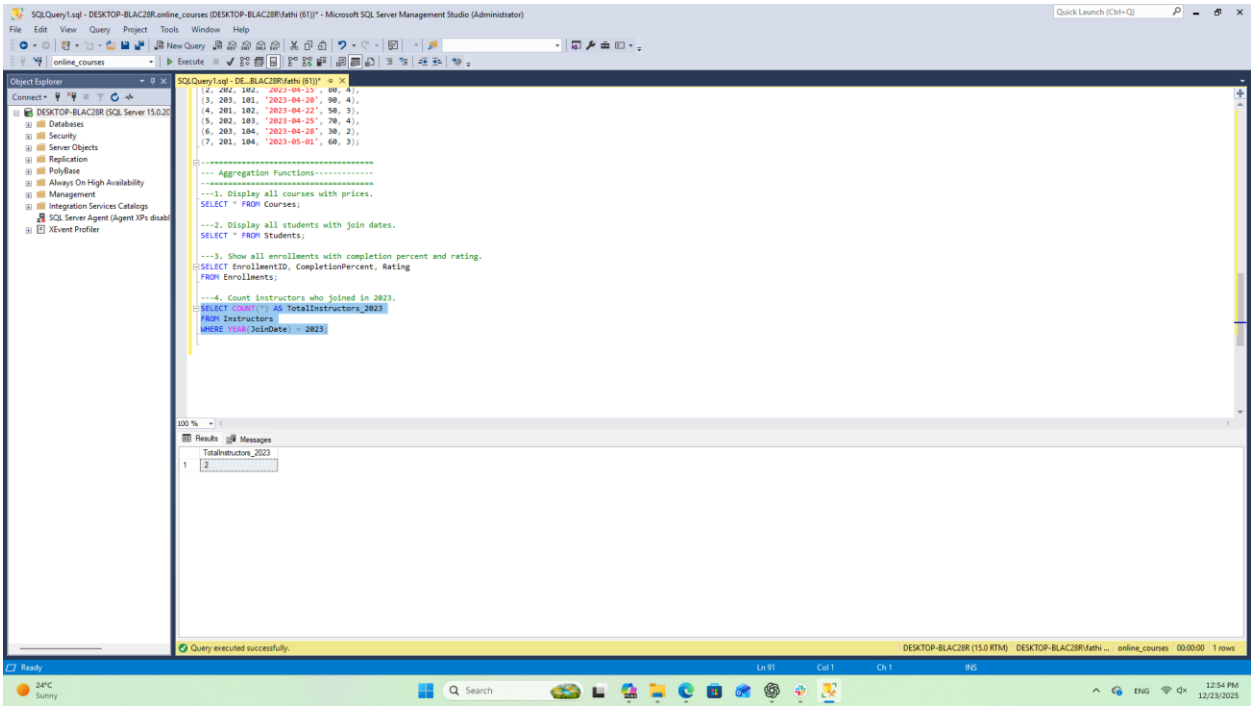
--- Aggregation Functions------------

---part 1----------------------
---1. Display all courses with prices.
SELECT Title, Price
FROM Courses;
---2. Display all students with join dates.
SELECT FullName, JoinDate
FROM Students;
Results
Messages
FullName
JoinDate
1 Ali Salim 2023-04-01
2 Layla Nasser 2023-04-05
3 Ahmed Said 2023-04-10
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
online_courses
00:00:00
3 rows
Ready
Ln 86
Col 1
Ch 1
INS
3:40 PM
12/23/2025


SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
SQLQuery1.sql - DE...BLAC28R\fathi (61))*
(104, 'Javascript Advanced', 1, 1, 59.99, '2023-05-01');
INSERT INTO Students VALUES
(201, 'Ali Salim', 'ali@student.com', '2023-04-01'),
(202, 'Layla Nasser', 'layla@student.com', '2023-04-05'),
(203, 'Ahmed Said', 'ahmed@student.com', '2023-04-10');
INSERT INTO Enrollments VALUES
(1, 201, 101, '2023-04-10', 100, 5),
(2, 202, 102, '2023-04-15', 80, 4),
(3, 203, 101, '2023-04-20', 90, 4),
(4, 201, 102, '2023-04-22', 50, 3),
(5, 202, 103, '2023-04-25', 70, 4),
(6, 203, 104, '2023-04-28', 30, 2),
(7, 201, 104, '2023-05-01', 60, 3);

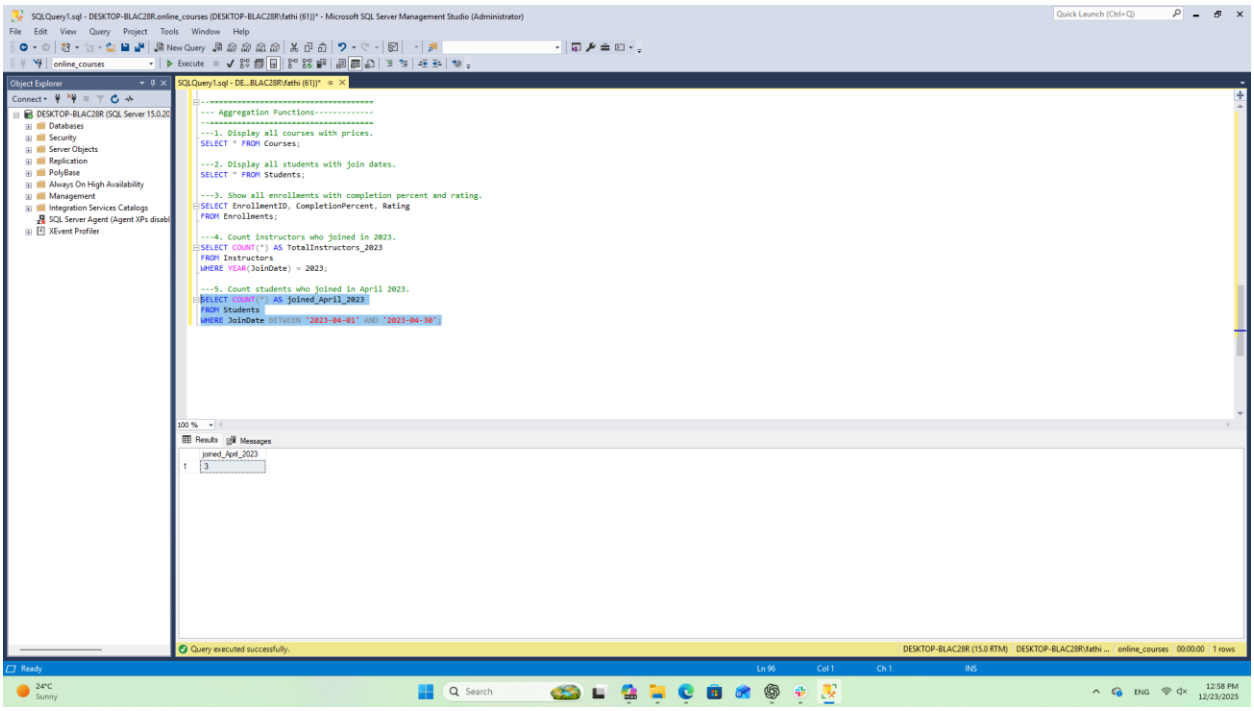
--- Aggregation Functions------------

---1. Display all courses with prices.
SELECT * FROM Courses;
---2. Display all students with join dates.
SELECT * FROM Students;
---3. Show all enrollments with completion percent and rating.
SELECT EnrollmentID, CompletionPercent, Rating
FROM Enrollments;
100 %
Results
Messages
EnrollmentID CompletionPercent Rating
1 1 100 5
2 2 80 4
3 3 90 4
4 4 50 3
5 5 70 4
6 6 30 2
7 7 60 3
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\fathi ... online_courses 00:00:00 7 rows
Ready
Ln 87 Col 1 Ch 1 INS
24°C Sunny
Search
ENG
12:47 PM 12/23/2025


SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
SQLQuery1.sql - DE...BLAC28R\fathi (61))*
(2, 202, 102, '2023-04-15', 80, 4),
(3, 203, 101, '2023-04-20', 90, 4),
(4, 201, 102, '2023-04-22', 50, 3),
(5, 202, 103, '2023-04-25', 70, 4),
(6, 203, 104, '2023-04-28', 30, 2),
(7, 201, 104, '2023-05-01', 60, 3);

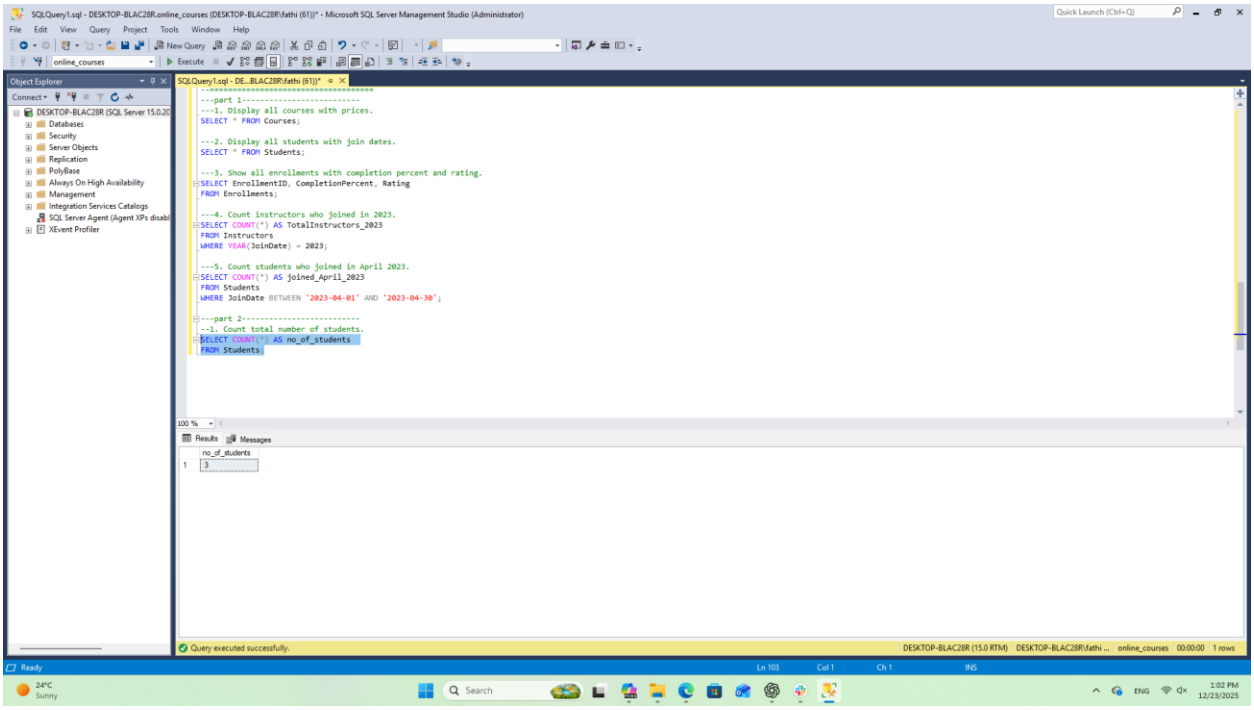
--- Aggregation Functions------------

---1. Display all courses with prices.
SELECT * FROM Courses;
---2. Display all students with join dates.
SELECT * FROM Students;
---3. Show all enrollments with completion percent and rating.
SELECT EnrollmentID, CompletionPercent, Rating
FROM Enrollments;
---4. Count instructors who joined in 2023.
SELECT COUNT(*) AS TotalInstructors_2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;
100 %
Results
Messages
TotalInstructors_2023
1 2
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\fathi ... online_courses 00:00:00 1 rows
Ready
Ln 91 Col 1 Ch 1 INS
24°C Sunny
Search
ENG
12:54 PM 12/23/2025

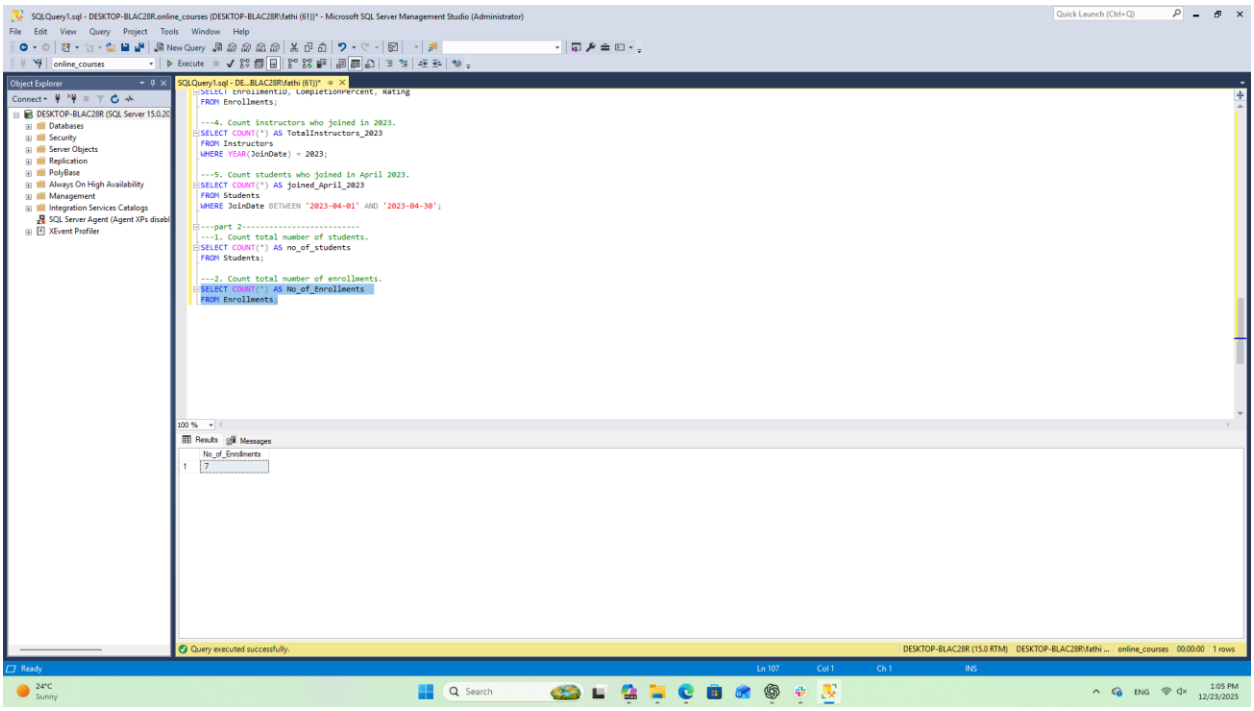

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
Object Explorer
SQLQuery1.sql - DE...BLAC28R\fathi (61))*

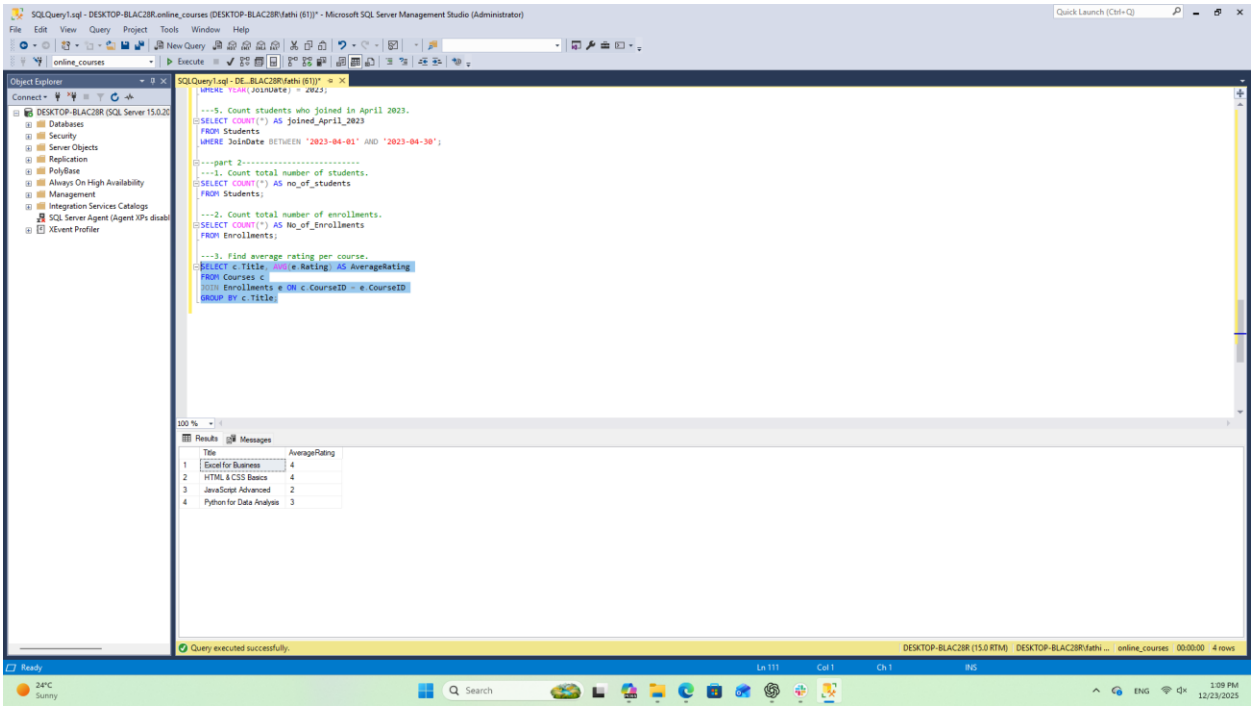
--- Aggregation Functions------------

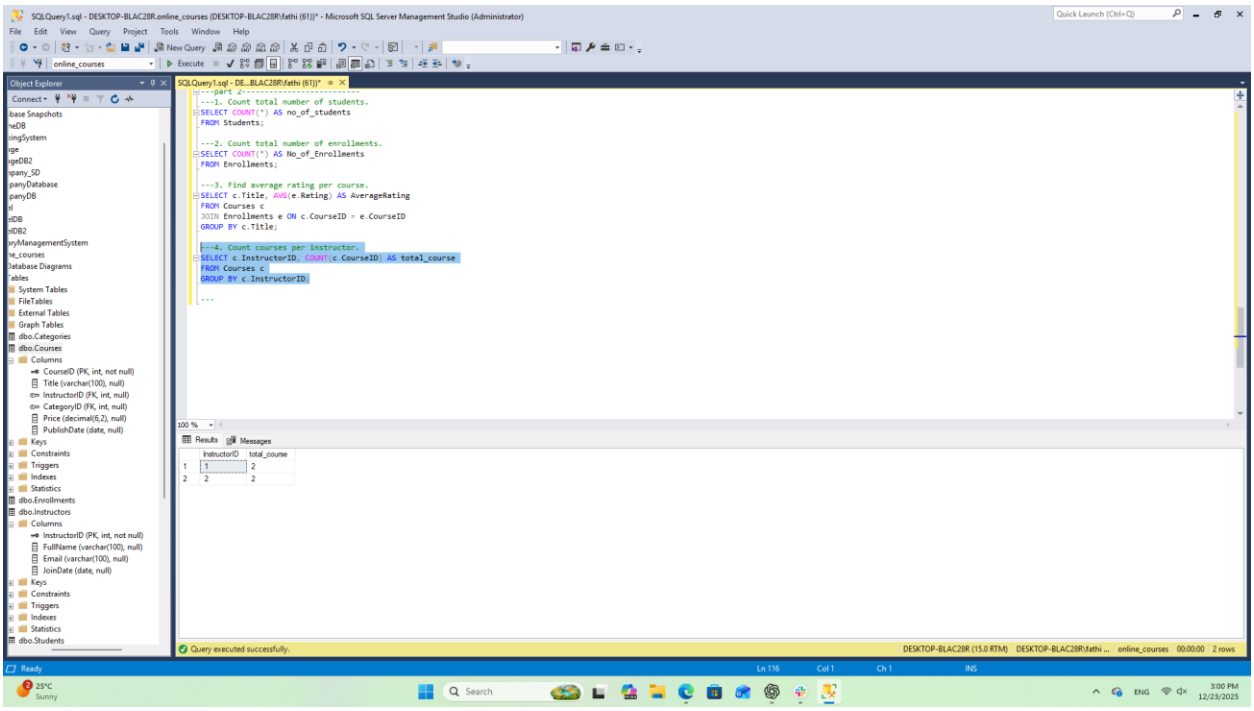
---1. Display all courses with prices.
SELECT * FROM Courses;
---2. Display all students with join dates.
SELECT * FROM Students;
---3. Show all enrollments with completion percent and rating.
SELECT EnrollmentID, CompletionPercent, Rating
FROM Enrollments;
---4. Count instructors who joined in 2023.
SELECT COUNT(*) AS TotalInstructors_2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;
---5. Count students who joined in April 2023.
SELECT COUNT(*) AS joined_April_2023
FROM Students
WHERE JoinDate BETWEEN '2023-04-01' AND '2023-04-30';
Results
Messages
joined_April_2023
1
3
Query executed successfully.
Ready


SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
Object Explorer
SQLQuery1.sql - DE...BLAC28R\fathi (61))*

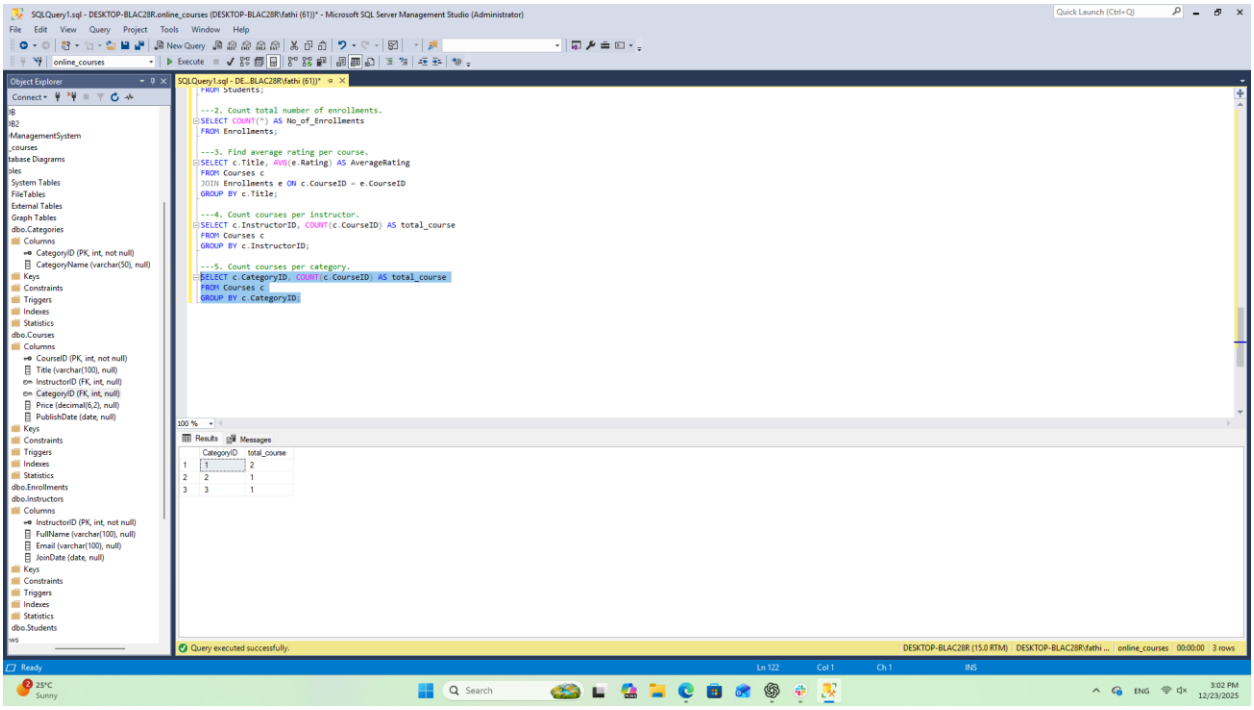
---part 1-------------------------
---1. Display all courses with prices.
SELECT * FROM Courses;
---2. Display all students with join dates.
SELECT * FROM Students;
---3. Show all enrollments with completion percent and rating.
SELECT EnrollmentID, CompletionPercent, Rating
FROM Enrollments;
---4. Count instructors who joined in 2023.
SELECT COUNT(*) AS TotalInstructors_2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;
---5. Count students who joined in April 2023.
SELECT COUNT(*) AS joined_April_2023
FROM Students
WHERE JoinDate BETWEEN '2023-04-01' AND '2023-04-30';
---part 2-------------------------
--1. Count total number of students.
SELECT COUNT(*) AS no_of_students
FROM Students
Results
Messages
no_of_students
1
3
Query executed successfully.
Ready

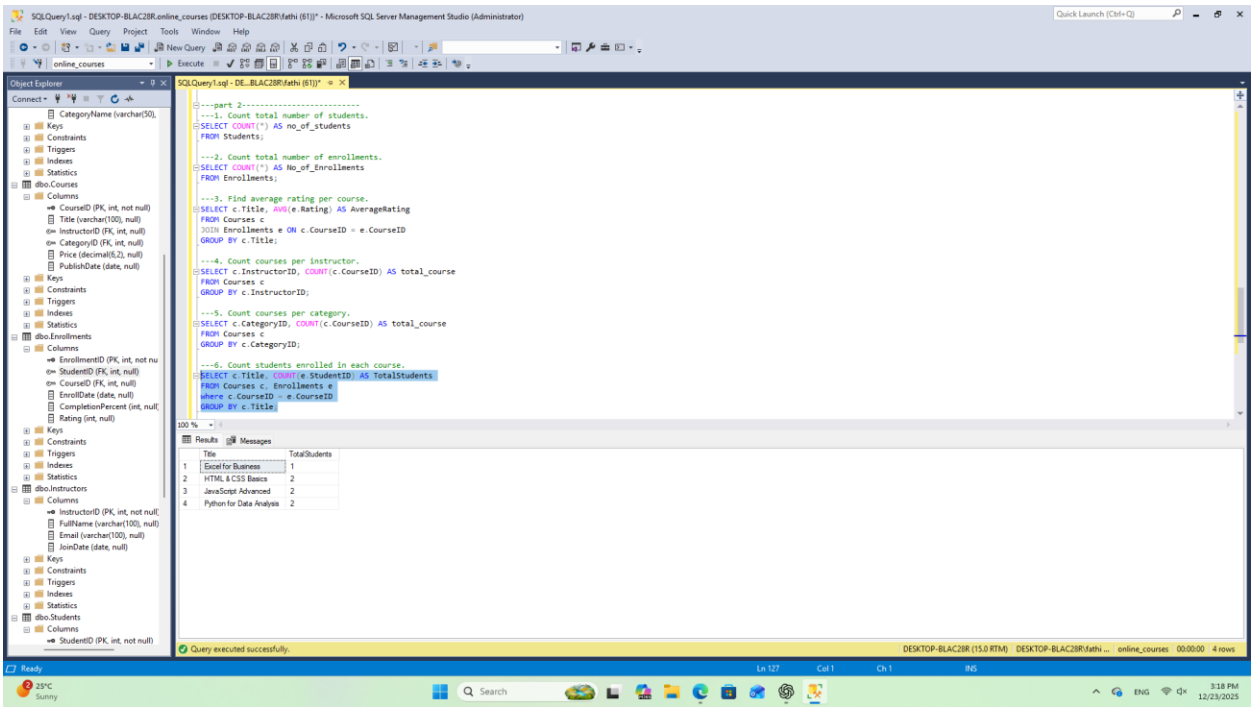

SELECT EnrollmentID, CompletionPercent, Rating
FROM Enrollments;
---4. Count instructors who joined in 2023.
SELECT COUNT(*) AS TotalInstructors_2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;
---5. Count students who joined in April 2023.
SELECT COUNT(*) AS joined_April_2023
FROM Students
WHERE JoinDate BETWEEN '2023-04-01' AND '2023-04-30';
---part 2-------------------------
---1. Count total number of students.
SELECT COUNT(*) AS no_of_students
FROM Students;
---2. Count total number of enrollments.
SELECT COUNT(*) AS No_of_Enrollments
FROM Enrollments;
Results
Messages
No_of_Enrollments
1
7
Query executed successfully.

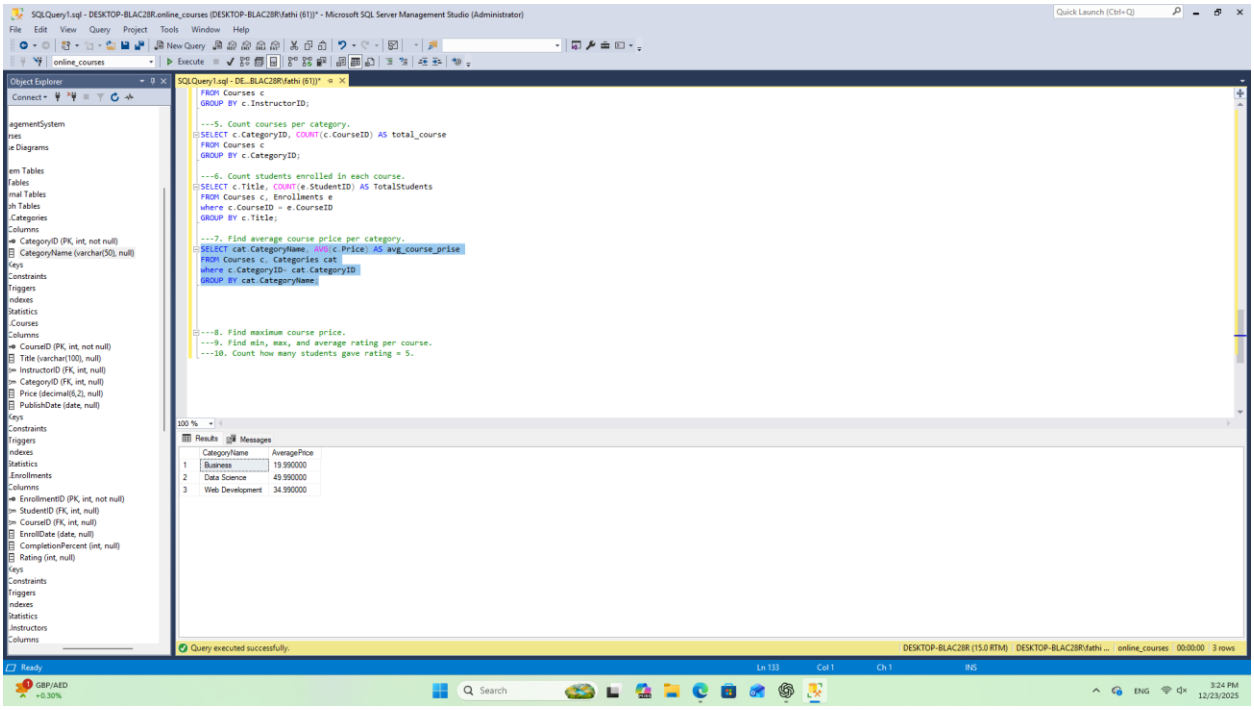

WHERE YEAR(JoinDate) = 2023;
---5. Count students who joined in April 2023.
SELECT COUNT(*) AS joined_April_2023
FROM Students
WHERE JoinDate BETWEEN '2023-04-01' AND '2023-04-30';
---part 2-------------------------
---1. Count total number of students.
SELECT COUNT(*) AS no_of_students
FROM Students;
---2. Count total number of enrollments.
SELECT COUNT(*) AS No_of_Enrollments
FROM Enrollments;
---3. Find average rating per course.
SELECT c.Title, AVG(e.Rating) AS AverageRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
Results
Messages
Title
AverageRating
1
Excel for Business
4
2
HTML & CSS Basics
4
3
JavaScript Advanced
2
4
Python for Data Analysis
3
Query executed successfully.

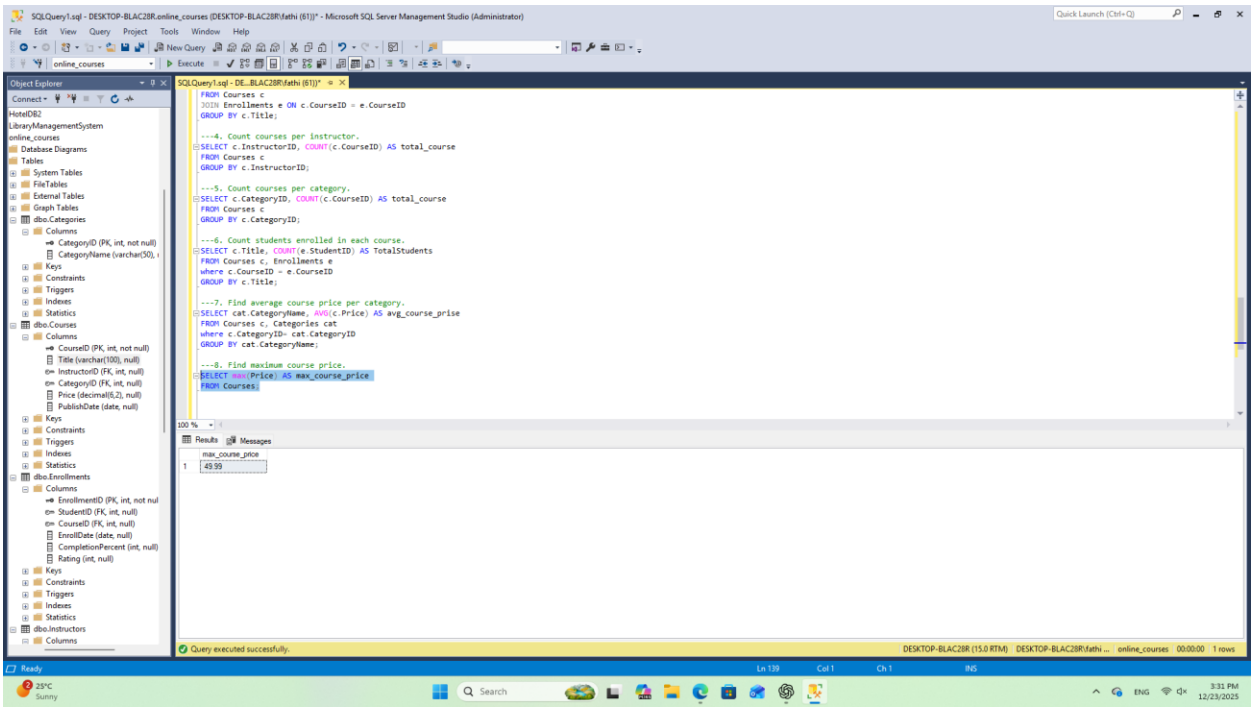

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
---part 2----------
---1. Count total number of students.
SELECT COUNT(*) AS no_of_students
FROM Students;
---2. Count total number of enrollments.
SELECT COUNT(*) AS No_of_Enrollments
FROM Enrollments;
---3. Find average rating per course.
SELECT c.Title, AVG(e.Rating) AS AverageRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---4. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;

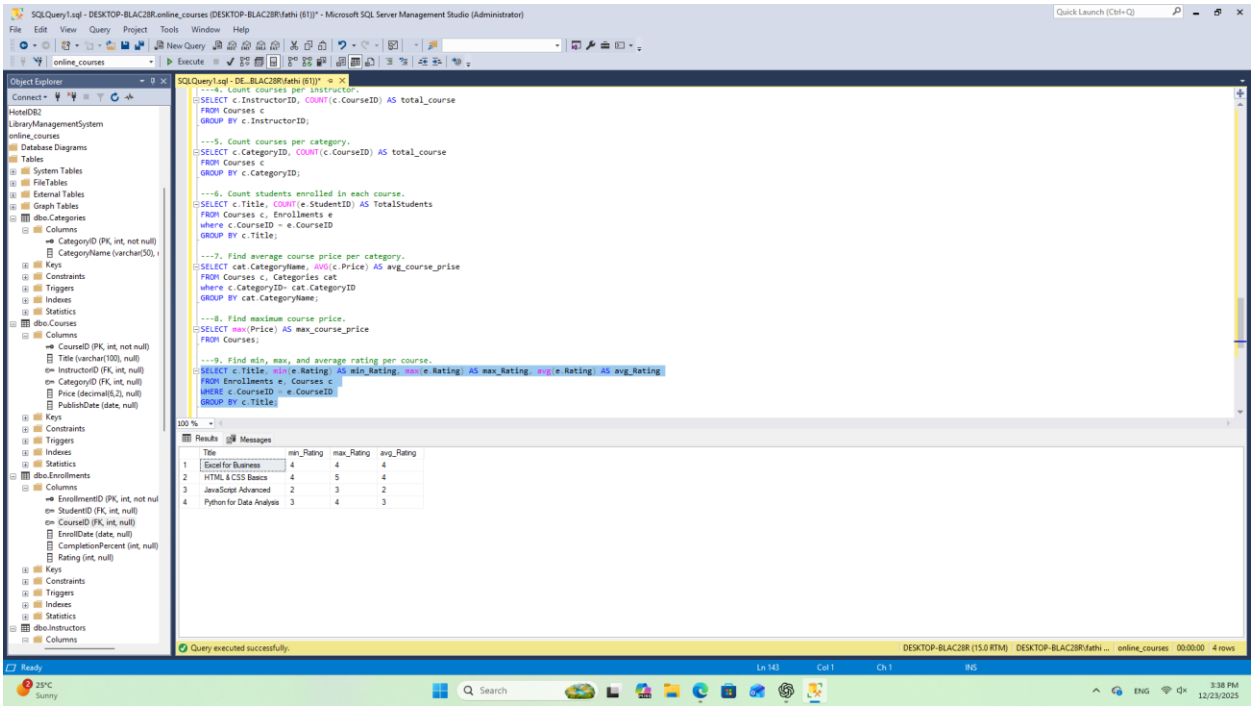
InstructorID total_course
1 1 2
2 2 2
Query executed successfully.
Ln 116 Col 1 Ch 1 INS

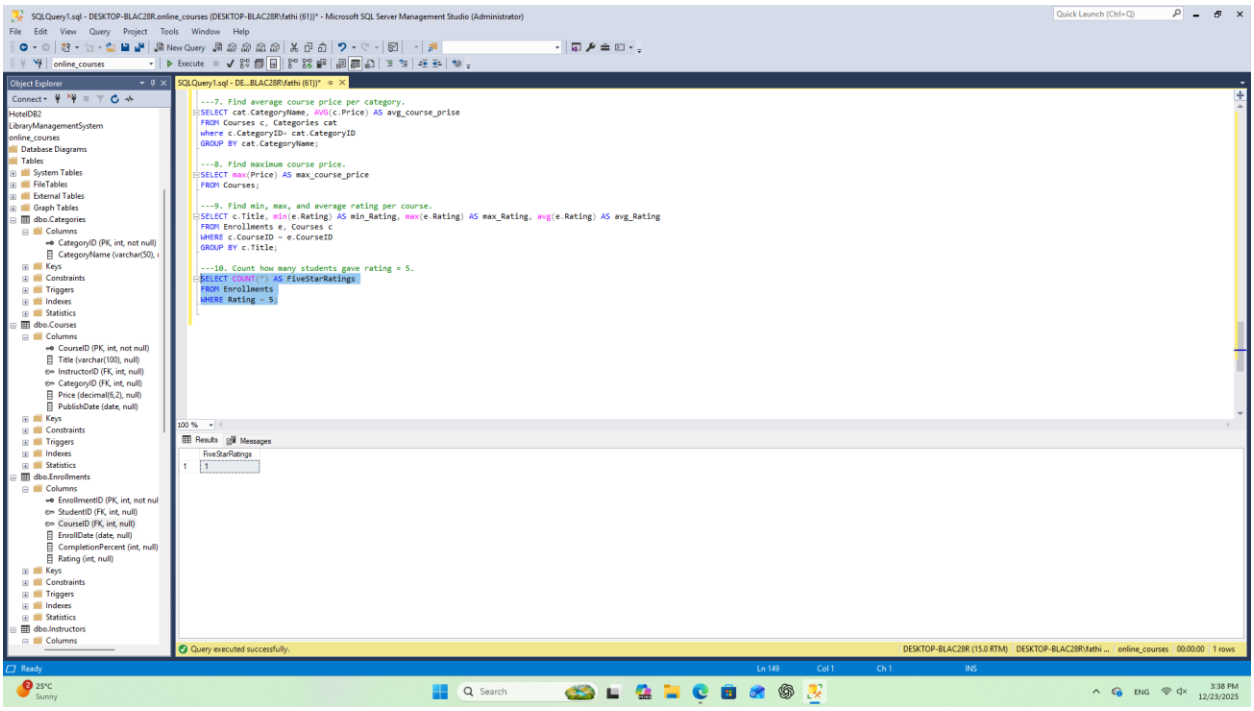

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
FROM Students;
---2. Count total number of enrollments.
SELECT COUNT(*) AS No_of_Enrollments
FROM Enrollments;
---3. Find average rating per course.
SELECT c.Title, AVG(e.Rating) AS AverageRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---4. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;
---5. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;
CategoryID total_course
1 1 2
2 2 1
3 3 1
Query executed successfully.
Ln 122 Col 1 Ch 1 INS

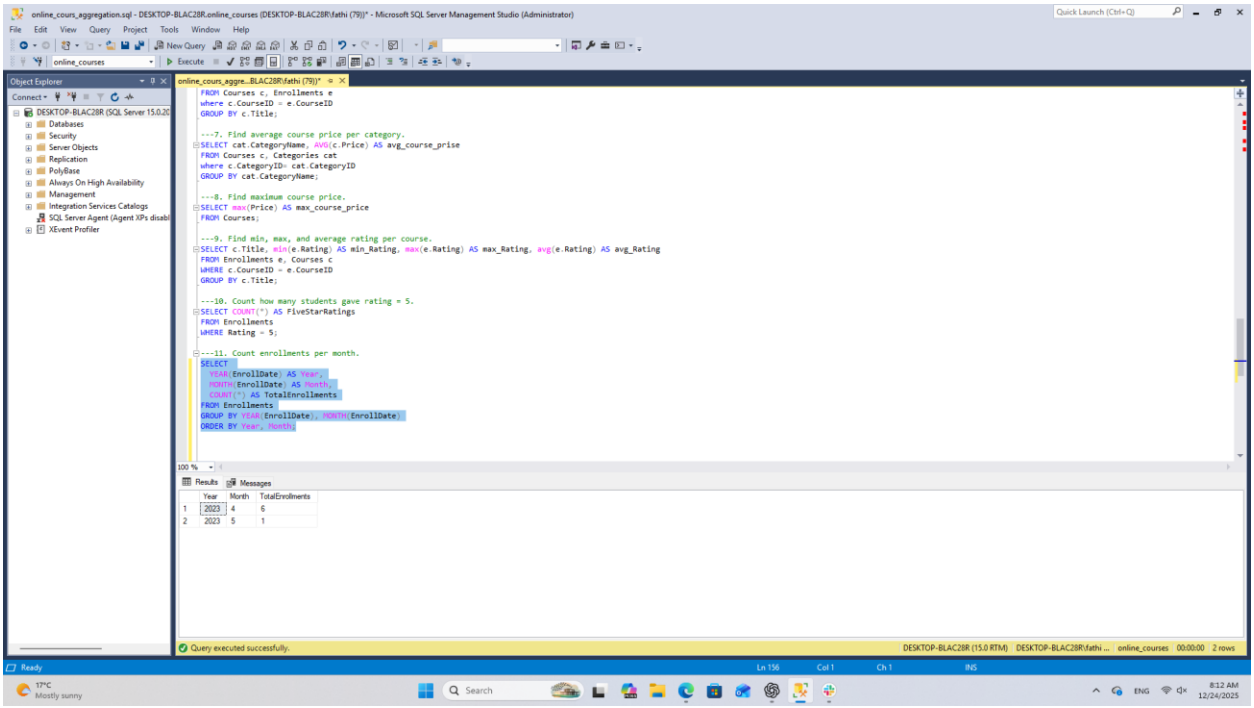

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
---part 2------------------------
---1. Count total number of students.
SELECT COUNT(*) AS no_of_students
FROM Students;
---2. Count total number of enrollments.
SELECT COUNT(*) AS No_of_Enrollments
FROM Enrollments;
---3. Find average rating per course.
SELECT c.Title, AVG(e.Rating) AS AverageRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---4. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;
---5. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;
---6. Count students enrolled in each course.
SELECT c.Title, COUNT(e.StudentID) AS TotalStudents
FROM Courses c, Enrollments e
where c.CourseID = e.CourseID
GROUP BY c.Title;
Title
TotalStudents
1 Excel for Business 1
2 HTML & CSS Basics 2
3 JavaScript Advanced 2
4 Python for Data Analysis 2
Query executed successfully.

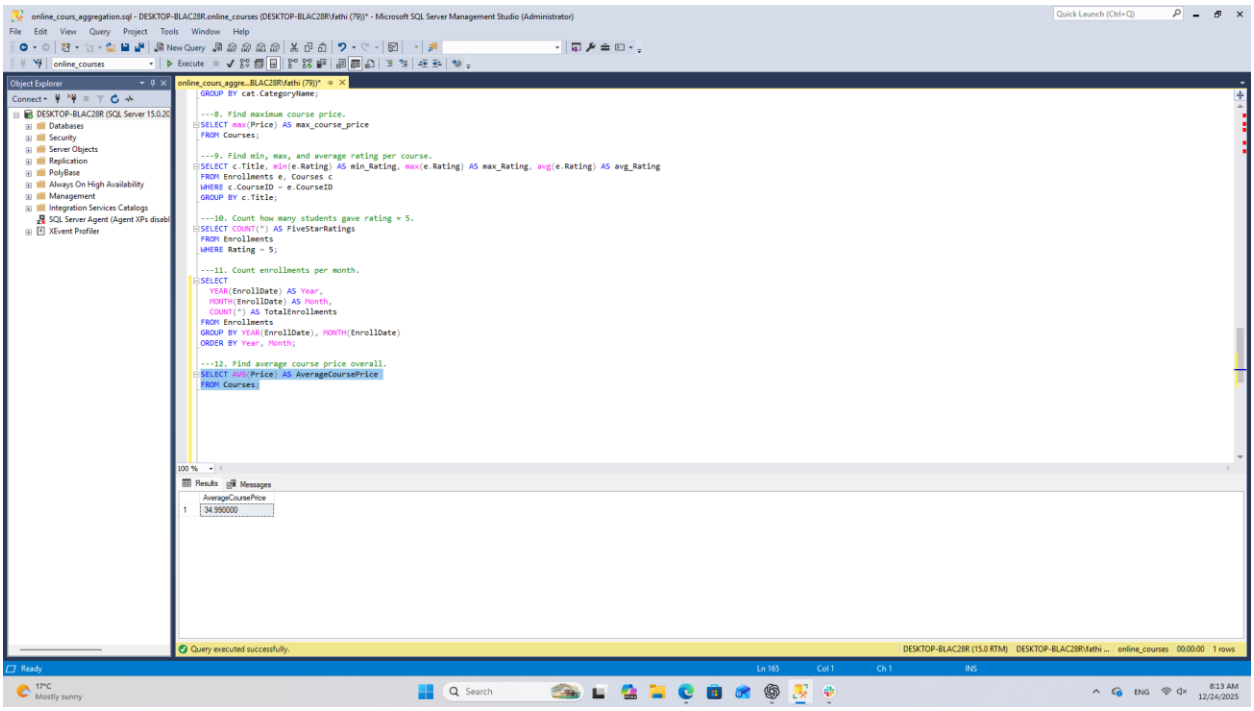

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
FROM Courses c
GROUP BY c.InstructorID;
---5. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;
---6. Count students enrolled in each course.
SELECT c.Title, COUNT(e.StudentID) AS TotalStudents
FROM Courses c, Enrollments e
where c.CourseID = e.CourseID
GROUP BY c.Title;
---7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_prise
FROM Courses c, Categories cat
where c.CategoryID= cat.CategoryID
GROUP BY cat.CategoryName;
---8. Find maximum course price.
---9. Find min, max, and average rating per course.
---10. Count how many students gave rating = 5.
CategoryName
AveragePrice
1 Business 19.990000
2 Data Science 49.990000
3 Web Development 34.990000
Query executed successfully.

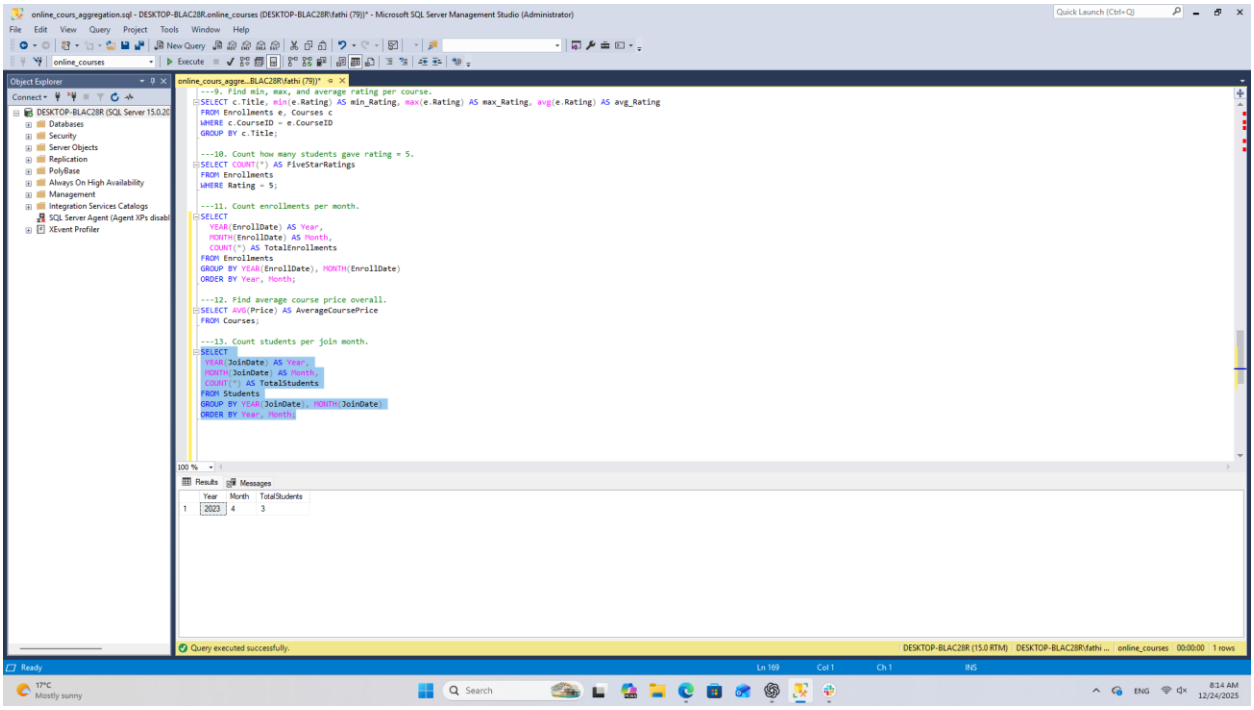

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---4. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;
---5. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;
---6. Count students enrolled in each course.
SELECT c.Title, COUNT(e.StudentID) AS TotalStudents
FROM Courses c, Enrollments e
where c.CourseID = e.CourseID
GROUP BY c.Title;
---7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_prise
FROM Courses c, Categories cat
where c.CategoryID= cat.CategoryID
GROUP BY cat.CategoryName;
---8. Find maximum course price.
SELECT max(Price) AS max_course_price
FROM Courses;
Results
Messages
max_course_price
1
49.99
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
online_courses
00:00:00
1 rows
Ready
Ln 139
Col 1
Ch 1
INS
3:31 PM
12/23/2025

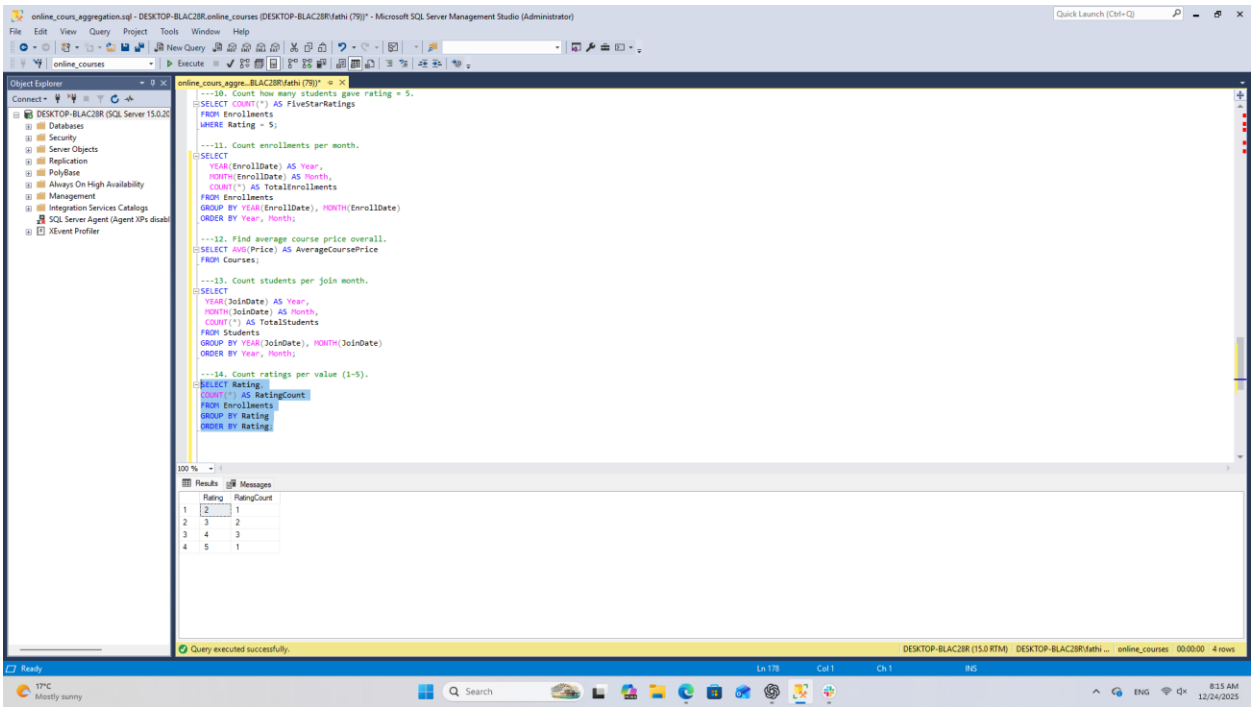

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
---4. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;
---5. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;
---6. Count students enrolled in each course.
SELECT c.Title, COUNT(e.StudentID) AS TotalStudents
FROM Courses c, Enrollments e
where c.CourseID = e.CourseID
GROUP BY c.Title;
---7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_prise
FROM Courses c, Categories cat
where c.CategoryID= cat.CategoryID
GROUP BY cat.CategoryName;
---8. Find maximum course price.
SELECT max(Price) AS max_course_price
FROM Courses;
---9. Find min, max, and average rating per course.
SELECT c.Title, min(e.Rating) AS min_Rating, max(e.Rating) AS max_Rating, avg(e.Rating) AS avg_Rating
FROM Enrollments e, Courses c
WHERE c.CourseID = e.CourseID
GROUP BY c.Title;
Results
Messages
Title
min_Rating
max_Rating
avg_Rating
1
Excel for Business
4
4
4
2
HTML & CSS Basics
4
5
4
3
JavaScript Advanced
2
3
2
4
Python for Data Analysis
3
4
3
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
online_courses
00:00:00
4 rows
Ready
Ln 143
Col 1
Ch 1
INS
3:38 PM
12/23/2025

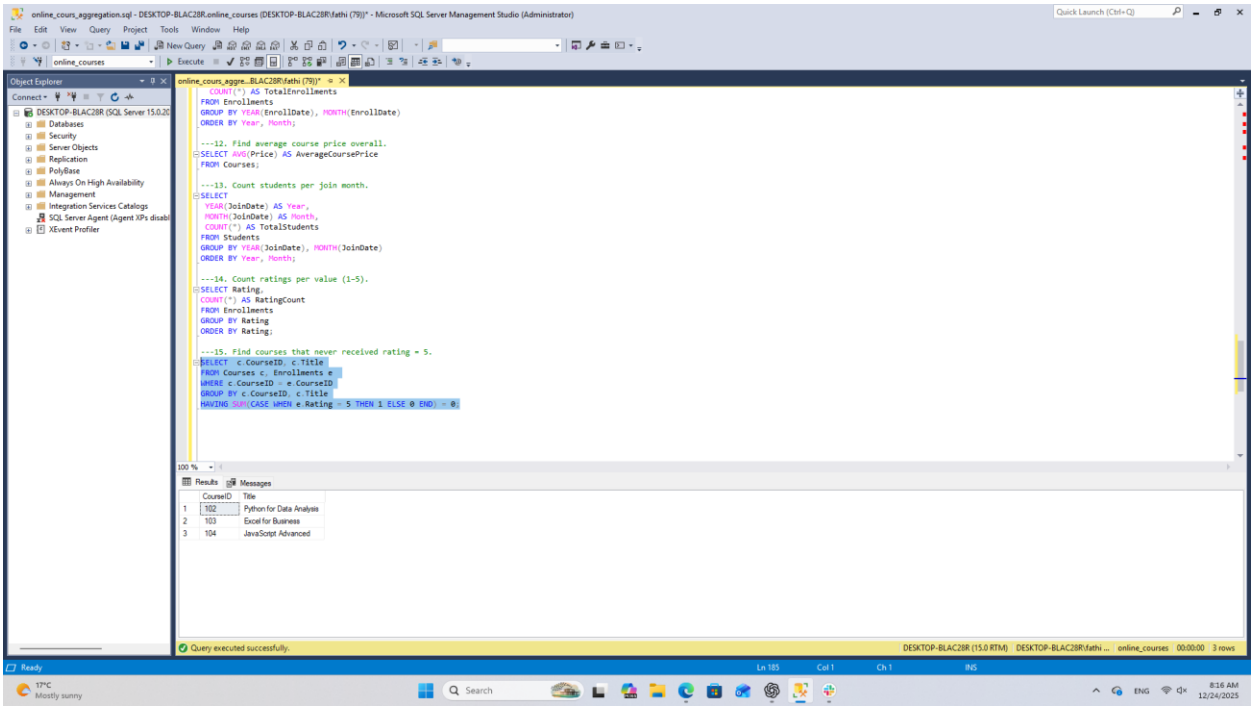

SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (61))* - Microsoft SQL Server Management Studio (Administrator)
---7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_prise
FROM Courses c, Categories cat
where c.CategoryID= cat.CategoryID
GROUP BY cat.CategoryName;
---8. Find maximum course price.
SELECT max(Price) AS max_course_price
FROM Courses;
---9. Find min, max, and average rating per course.
SELECT c.Title, min(e.Rating) AS min_Rating, max(e.Rating) AS max_Rating, avg(e.Rating) AS avg_Rating
FROM Enrollments e, Courses c
WHERE c.CourseID = e.CourseID
GROUP BY c.Title;
---10. Count how many students gave rating = 5.
SELECT COUNT(*) AS FiveStarRatings
FROM Enrollments
WHERE Rating = 5;
FiveStarRatings
1
Query executed successfully.

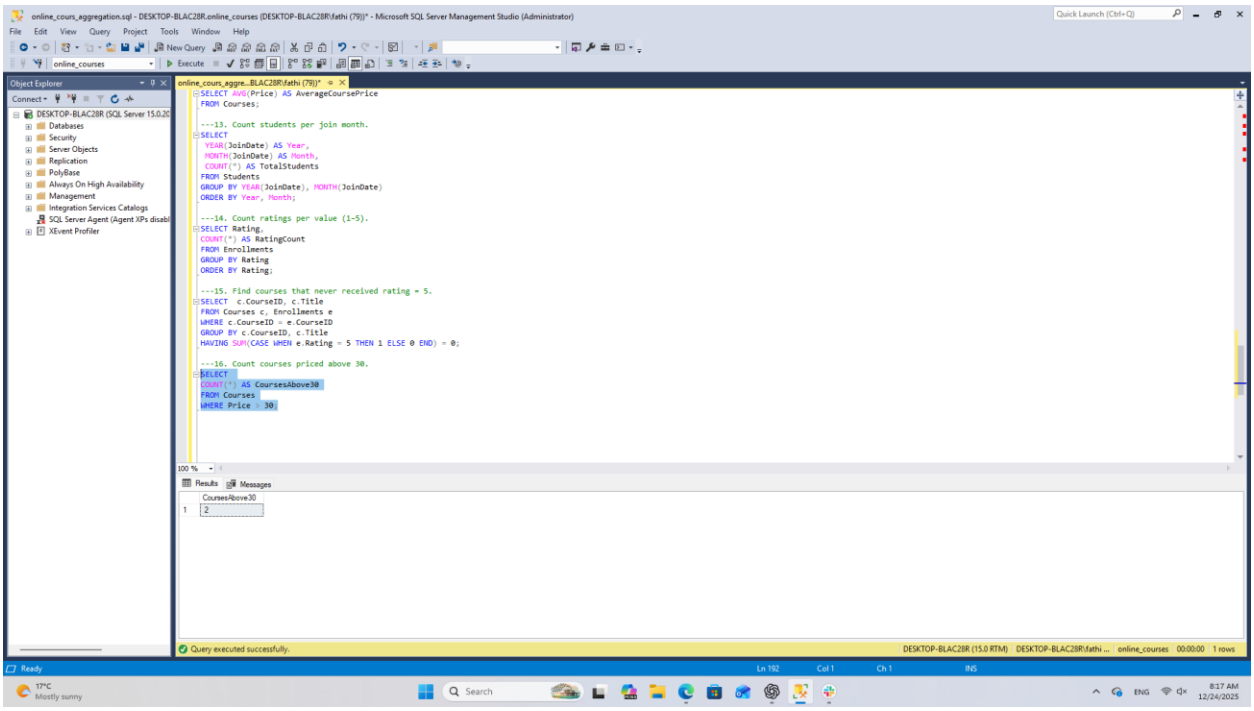

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_prise
FROM Courses c, Categories cat
where c.CategoryID= cat.CategoryID
GROUP BY cat.CategoryName;
---8. Find maximum course price.
SELECT max(Price) AS max_course_price
FROM Courses;
---9. Find min, max, and average rating per course.
SELECT c.Title, min(e.Rating) AS min_Rating, max(e.Rating) AS max_Rating, avg(e.Rating) AS avg_Rating
FROM Enrollments e, Courses c
WHERE c.CourseID = e.CourseID
GROUP BY c.Title;
---10. Count how many students gave rating = 5.
SELECT COUNT(*) AS FiveStarRatings
FROM Enrollments
WHERE Rating = 5;
---11. Count enrollments per month.
SELECT
YEAR(EnrollDate) AS Year,
MONTH(EnrollDate) AS Month,
COUNT(*) AS TotalEnrollments
FROM Enrollments
GROUP BY YEAR(EnrollDate), MONTH(EnrollDate)
ORDER BY Year, Month;
Year Month TotalEnrollments
1 2023 4 6
2 2023 5 1
Query executed successfully.

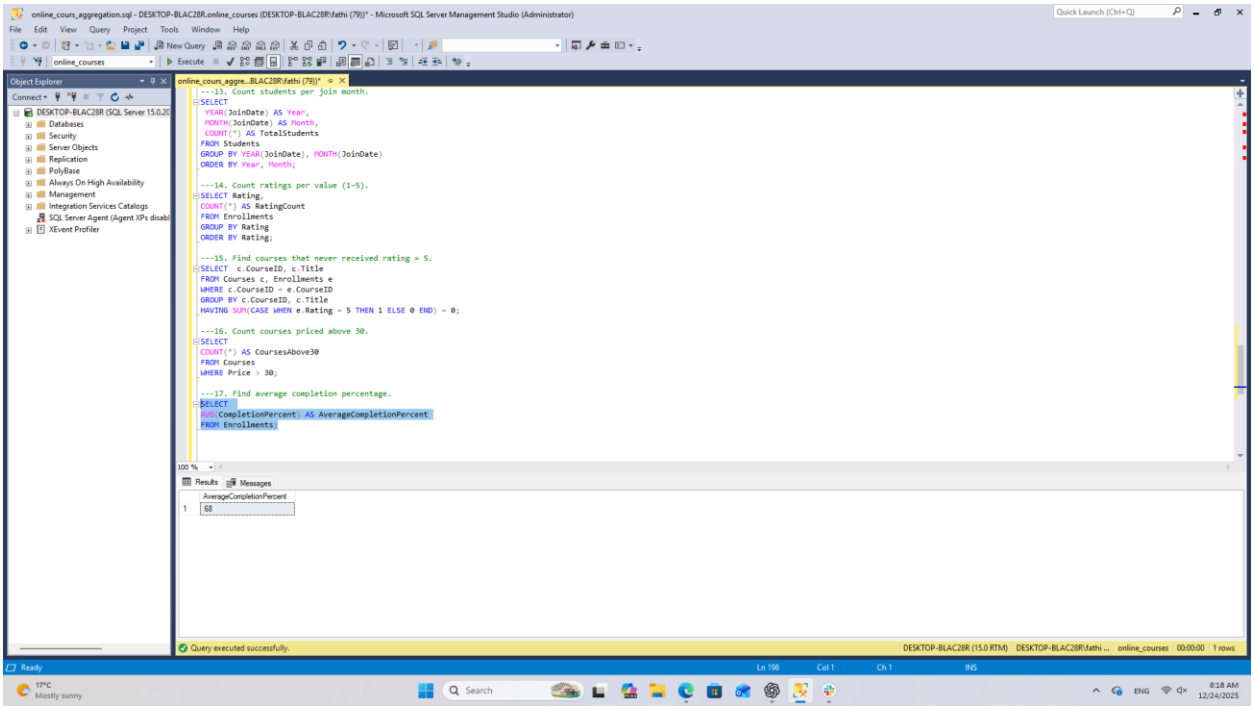

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
GROUP BY cat.CategoryName;
---8. Find maximum course price.
SELECT max(Price) AS max_course_price
FROM Courses;
---9. Find min, max, and average rating per course.
SELECT c.Title, min(e.Rating) AS min_Rating, max(e.Rating) AS max_Rating, avg(e.Rating) AS avg_Rating
FROM Enrollments e, Courses c
WHERE c.CourseID = e.CourseID
GROUP BY c.Title;
---10. Count how many students gave rating = 5.
SELECT COUNT(*) AS FiveStarRatings
FROM Enrollments
WHERE Rating = 5;
---11. Count enrollments per month.
SELECT
YEAR(EnrollDate) AS Year,
MONTH(EnrollDate) AS Month,
COUNT(*) AS TotalEnrollments
FROM Enrollments
GROUP BY YEAR(EnrollDate), MONTH(EnrollDate)
ORDER BY Year, Month;
---12. Find average course price overall.
SELECT AVG(Price) AS AverageCoursePrice
FROM Courses;
100 %
Results
Messages
AverageCoursePrice
1
34.990000
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
1 rows
Ready
Ln 165
Col 1
Ch 1
INS
17°C
Mostly sunny
Search
ENG
8:13 AM
12/24/2025

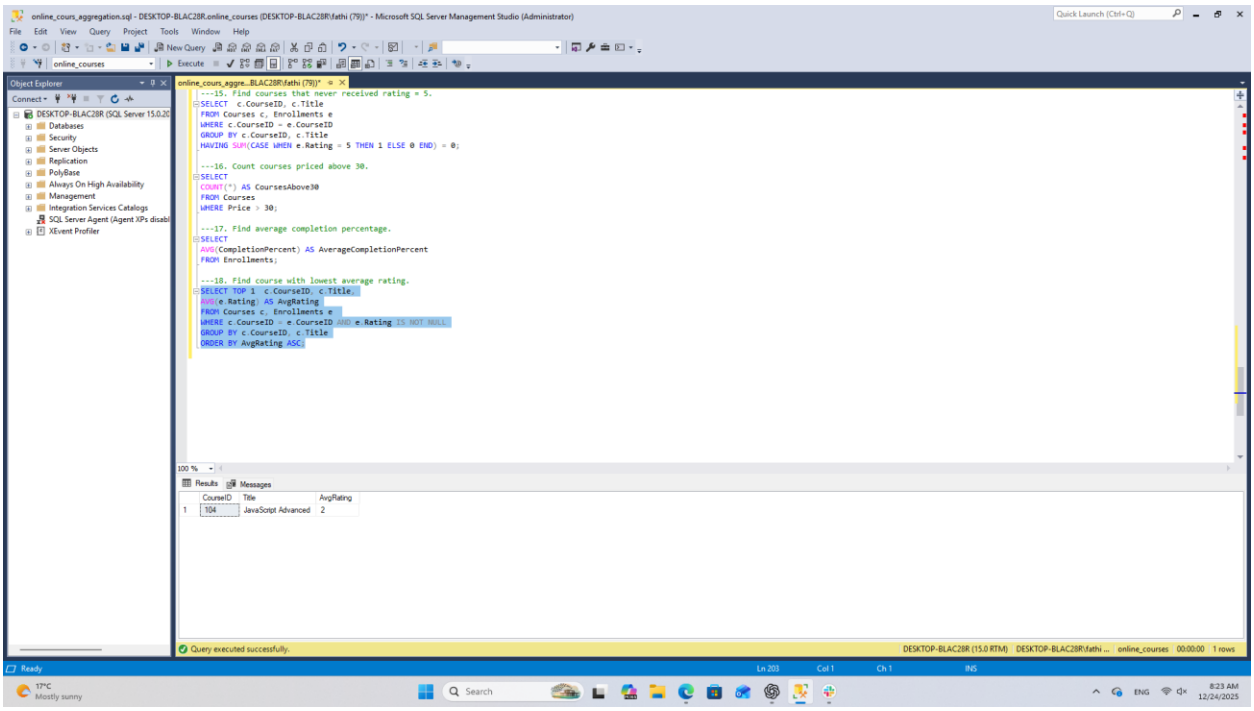

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
---9. Find min, max, and average rating per course.
SELECT c.Title, min(e.Rating) AS min_Rating, max(e.Rating) AS max_Rating, avg(e.Rating) AS avg_Rating
FROM Enrollments e, Courses c
WHERE c.CourseID = e.CourseID
GROUP BY c.Title;
---10. Count how many students gave rating = 5.
SELECT COUNT(*) AS FiveStarRatings
FROM Enrollments
WHERE Rating = 5;
---11. Count enrollments per month.
SELECT
YEAR(EnrollDate) AS Year,
MONTH(EnrollDate) AS Month,
COUNT(*) AS TotalEnrollments
FROM Enrollments
GROUP BY YEAR(EnrollDate), MONTH(EnrollDate)
ORDER BY Year, Month;
---12. Find average course price overall.
SELECT AVG(Price) AS AverageCoursePrice
FROM Courses;
---13. Count students per join month.
SELECT
YEAR(JoinDate) AS Year,
MONTH(JoinDate) AS Month,
COUNT(*) AS TotalStudents
FROM Students
GROUP BY YEAR(JoinDate), MONTH(JoinDate)
ORDER BY Year, Month;
100 %
Results
Messages
Year
Month
TotalStudents
1
2023
4
3
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
1 rows
Ready
Ln 169
Col 1
Ch 1
INS
17°C
Mostly sunny
Search
ENG
8:14 AM
12/24/2025


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---10. Count how many students gave rating = 5.
SELECT COUNT(*) AS FiveStarRatings
FROM Enrollments
WHERE Rating = 5;
---11. Count enrollments per month.
SELECT
YEAR(EnrollDate) AS Year,
MONTH(EnrollDate) AS Month,
COUNT(*) AS TotalEnrollments
FROM Enrollments
GROUP BY YEAR(EnrollDate), MONTH(EnrollDate)
ORDER BY Year, Month;
---12. Find average course price overall.
SELECT AVG(Price) AS AverageCoursePrice
FROM Courses;
---13. Count students per join month.
SELECT
YEAR(JoinDate) AS Year,
MONTH(JoinDate) AS Month,
COUNT(*) AS TotalStudents
FROM Students
GROUP BY YEAR(JoinDate), MONTH(JoinDate)
ORDER BY Year, Month;
---14. Count ratings per value (1-5).
SELECT Rating,
COUNT(*) AS RatingCount
FROM Enrollments
GROUP BY Rating
ORDER BY Rating;
Rating RatingCount
1 2 1
2 3 2
3 4 3
4 5 1
Query executed successfully.
Ln 178 Col 1 Ch 1 INS


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
COUNT(*) AS TotalEnrollments
FROM Enrollments
GROUP BY YEAR(EnrollDate), MONTH(EnrollDate)
ORDER BY Year, Month;
---12. Find average course price overall.
SELECT AVG(Price) AS AverageCoursePrice
FROM Courses;
---13. Count students per join month.
SELECT
YEAR(JoinDate) AS Year,
MONTH(JoinDate) AS Month,
COUNT(*) AS TotalStudents
FROM Students
GROUP BY YEAR(JoinDate), MONTH(JoinDate)
ORDER BY Year, Month;
---14. Count ratings per value (1-5).
SELECT Rating,
COUNT(*) AS RatingCount
FROM Enrollments
GROUP BY Rating
ORDER BY Rating;
---15. Find courses that never received rating = 5.
SELECT c.CourseID, c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.CourseID, c.Title
HAVING SUM(CASE WHEN e.Rating = 5 THEN 1 ELSE 0 END) = 0;
CourseID Title
1 102 Python for Data Analysis
2 103 Excel for Business
3 104 JavaScript Advanced
Query executed successfully.
Ln 185 Col 1 Ch 1 INS


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
SELECT AVG(Price) AS AverageCoursePrice
FROM Courses;
---13. Count students per join month.
SELECT
YEAR(JoinDate) AS Year,
MONTH(JoinDate) AS Month,
COUNT(*) AS TotalStudents
FROM Students
GROUP BY YEAR(JoinDate), MONTH(JoinDate)
ORDER BY Year, Month;
---14. Count ratings per value (1-5).
SELECT Rating,
COUNT(*) AS RatingCount
FROM Enrollments
GROUP BY Rating
ORDER BY Rating;
---15. Find courses that never received rating = 5.
SELECT c.CourseID, c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.CourseID, c.Title
HAVING SUM(CASE WHEN e.Rating = 5 THEN 1 ELSE 0 END) = 0;
---16. Count courses priced above 30.
SELECT
COUNT(*) AS CoursesAbove30
FROM Courses
WHERE Price > 30
Results
Messages
CoursesAbove30
1
2
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
1 rows
Ready
Ln 192
Col 1
Ch 1
INS
8:17 AM
12/24/2025


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---13. Count students per join month.
SELECT
YEAR(JoinDate) AS Year,
MONTH(JoinDate) AS Month,
COUNT(*) AS TotalStudents
FROM Students
GROUP BY YEAR(JoinDate), MONTH(JoinDate)
ORDER BY Year, Month;
---14. Count ratings per value (1-5).
SELECT Rating,
COUNT(*) AS RatingCount
FROM Enrollments
GROUP BY Rating
ORDER BY Rating;
---15. Find courses that never received rating = 5.
SELECT c.CourseID, c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.CourseID, c.Title
HAVING SUM(CASE WHEN e.Rating = 5 THEN 1 ELSE 0 END) = 0;
---16. Count courses priced above 30.
SELECT
COUNT(*) AS CoursesAbove30
FROM Courses
WHERE Price > 30;
---17. Find average completion percentage.
SELECT
AVG(CompletionPercent) AS AverageCompletionPercent
FROM Enrollments;
Results
Messages
AverageCompletionPercent
1
68
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
1 rows
Ready
Ln 198
Col 1
Ch 1
INS
8:18 AM
12/24/2025



Answer briefly:

• What was easiest?

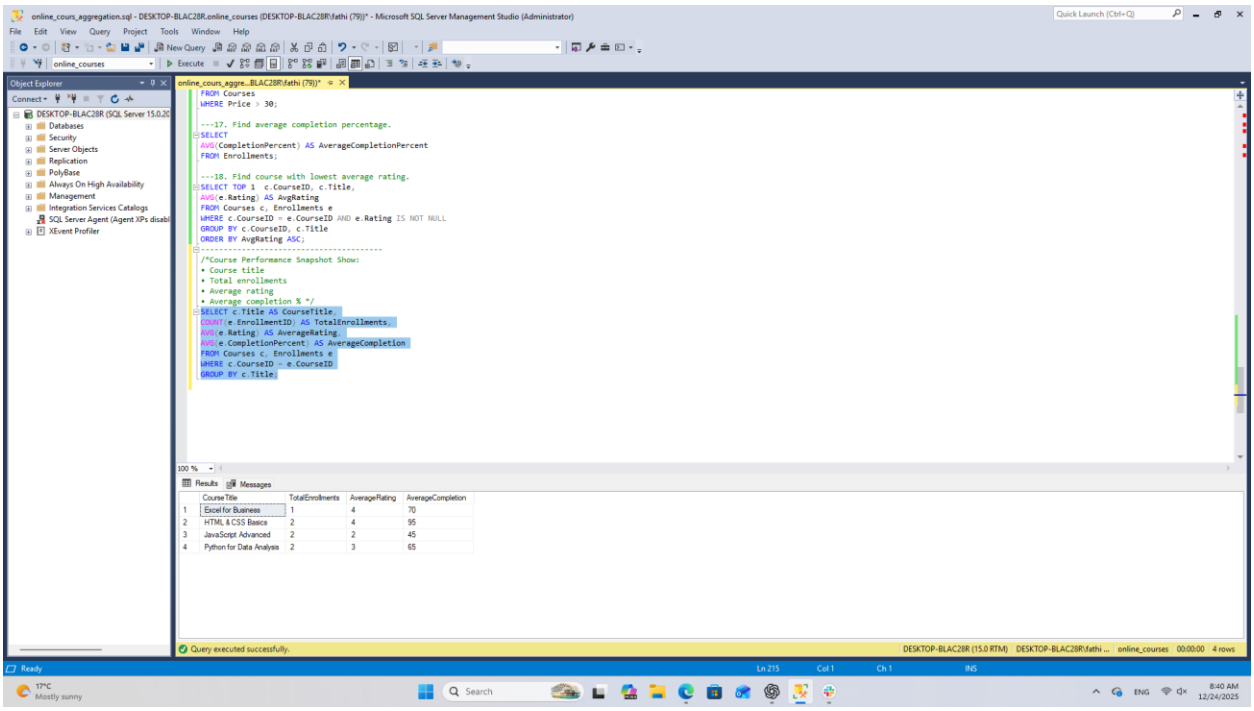
Simple counts like total students or enrollments.

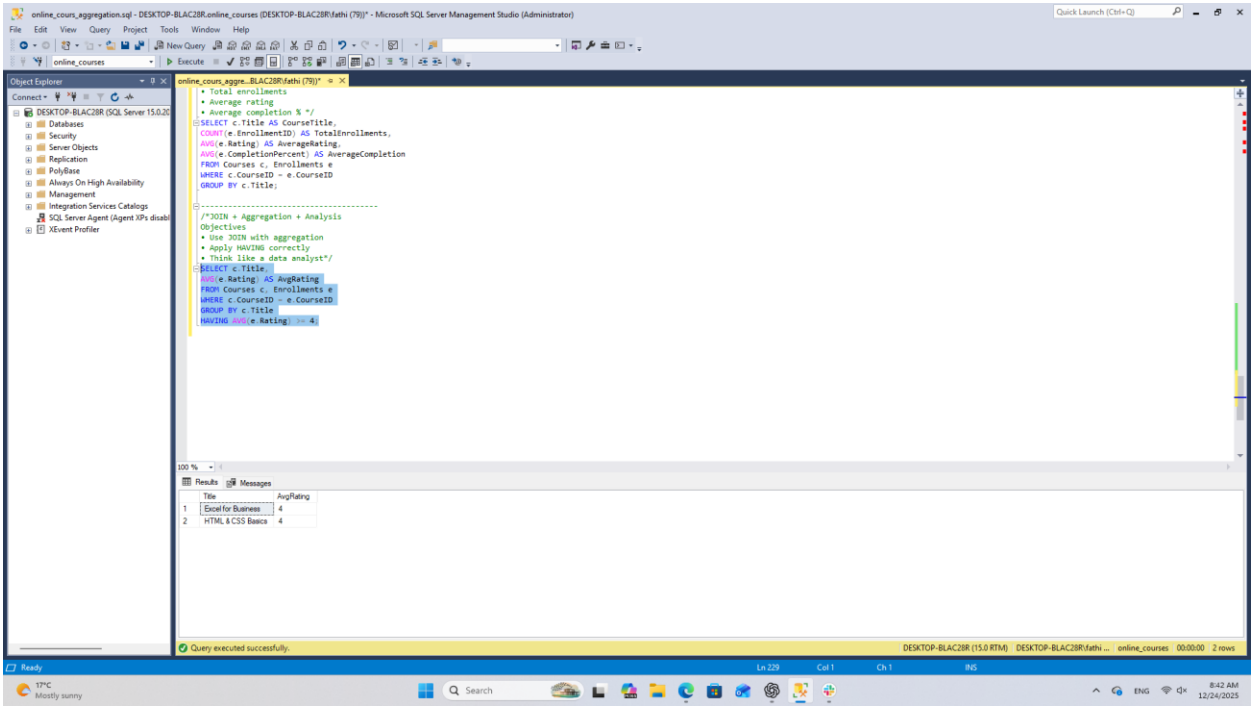
• What was hardest?

Queries combining `JOIN + GROUP BY + HAVING` (especially averages per course).

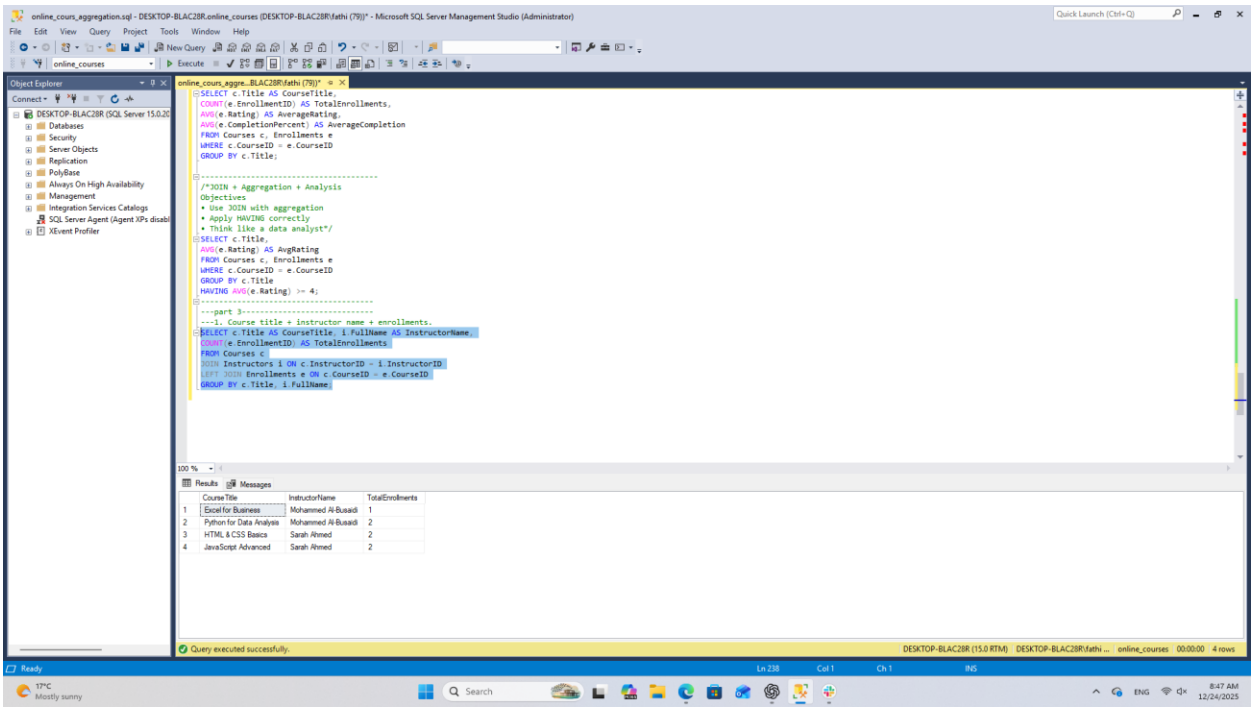
• What does GROUP BY do in your own words?

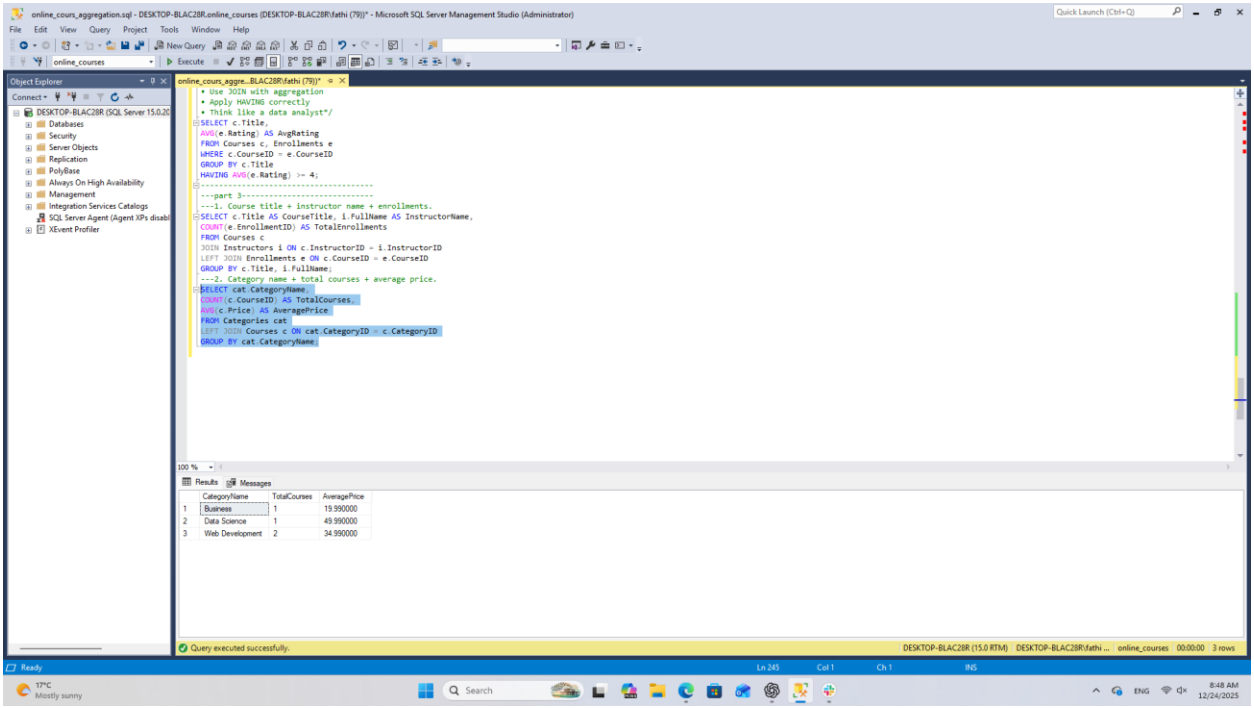
It groups rows that share the same value so aggregate functions (COUNT, AVG, MAX…) work **per group instead of the whole table**.

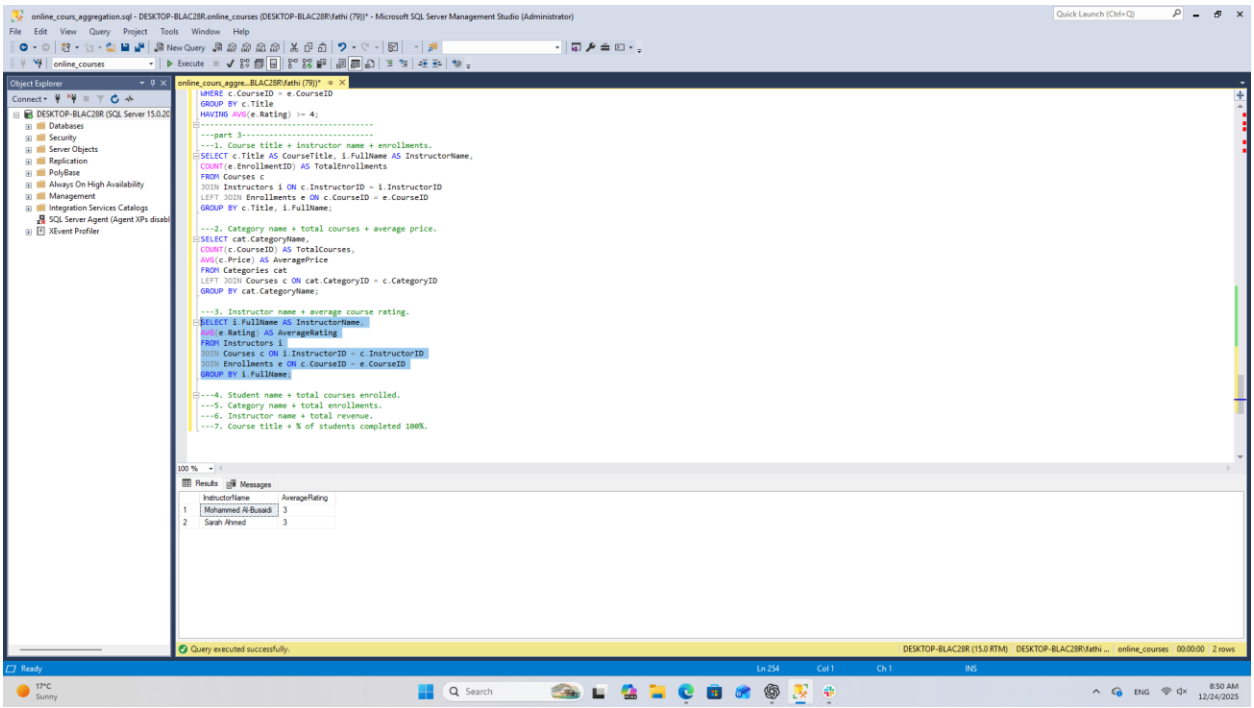


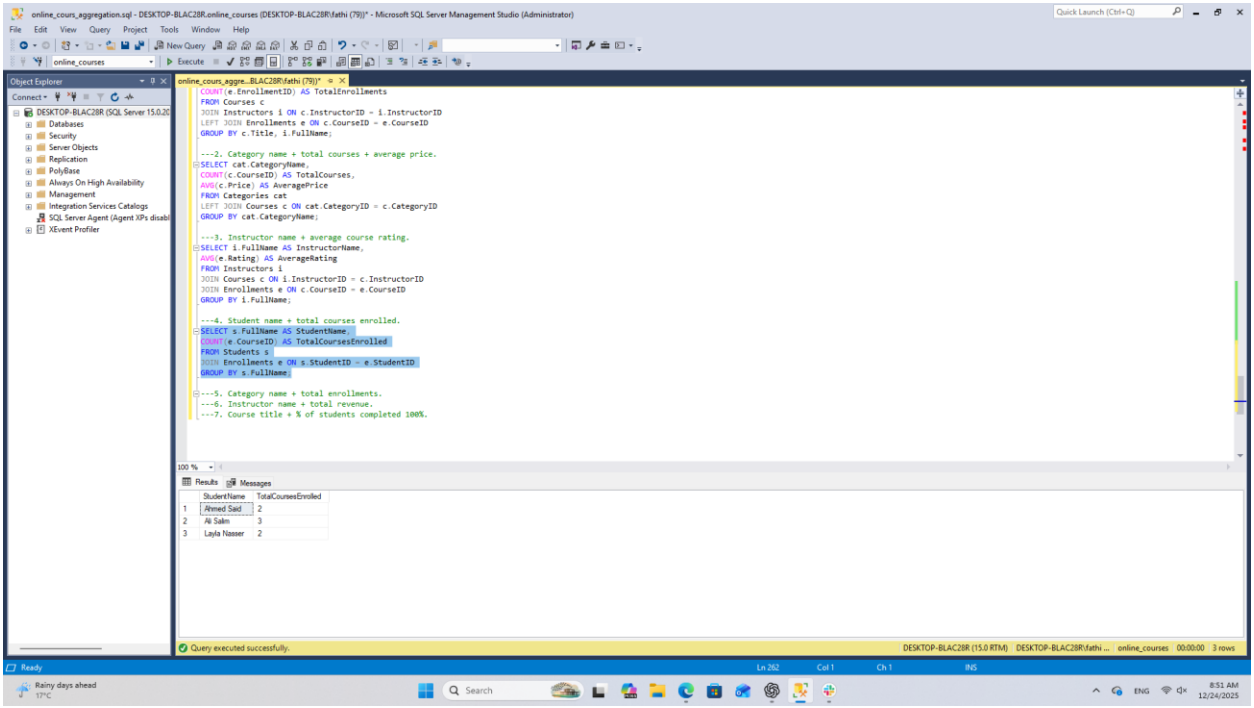


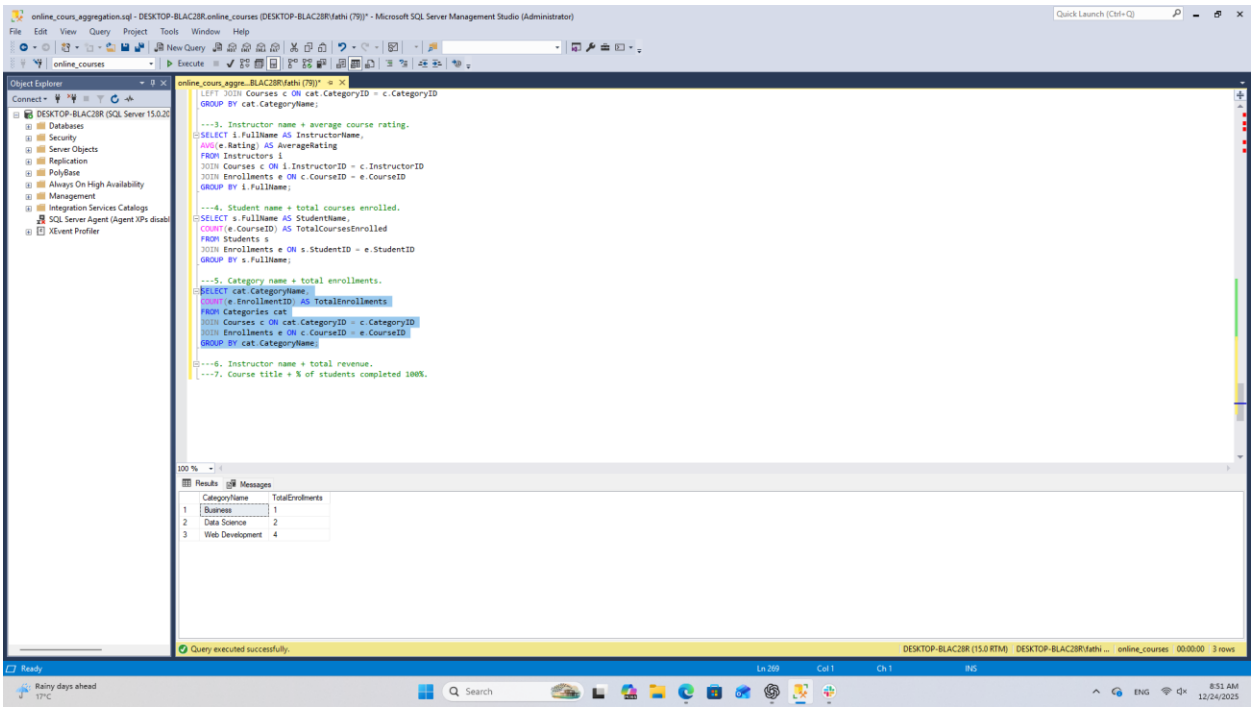
Courses with **average rating ≥ 4**

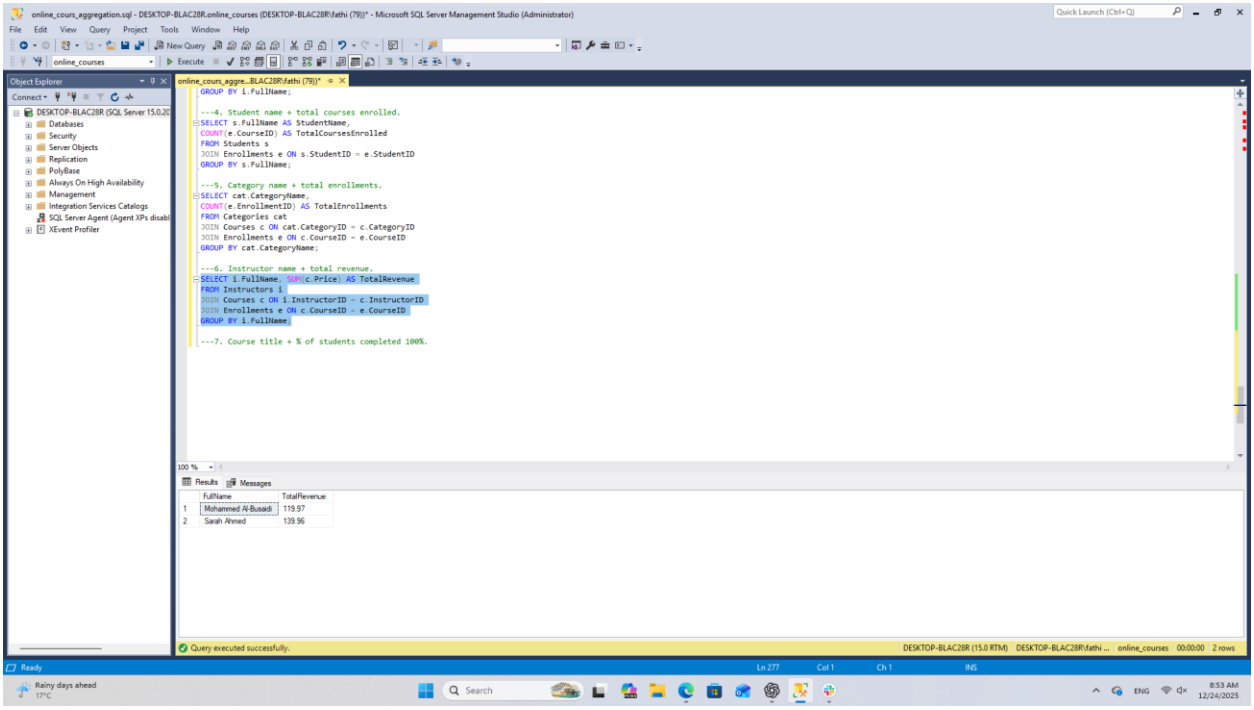

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
SELECT c.Title AS CourseTitle,
COUNT(e.EnrollmentID) AS TotalEnrollments,
AVG(e.Rating) AS AverageRating,
AVG(e.CompletionPercent) AS AverageCompletion
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title;
/*JOIN + Aggregation + Analysis
Objectives
• Use JOIN with aggregation
• Apply HAVING correctly
• Think like a data analyst*/
SELECT c.Title,
AVG(e.Rating) AS AvgRating
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING AVG(e.Rating) >= 4;
---part 3---
---1. Course title + instructor name + enrollments.
SELECT c.Title AS CourseTitle, i.FullName AS InstructorName,
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Courses c
JOIN Instructors i ON c.InstructorID = i.InstructorID
LEFT JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title, i.FullName
Results
Messages
CourseTitle 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
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\fathi ... online_courses 00:00:00 4 rows
Ready
Ln 238 Col 1 Ch 1 INS
8:47 AM 12/24/2025

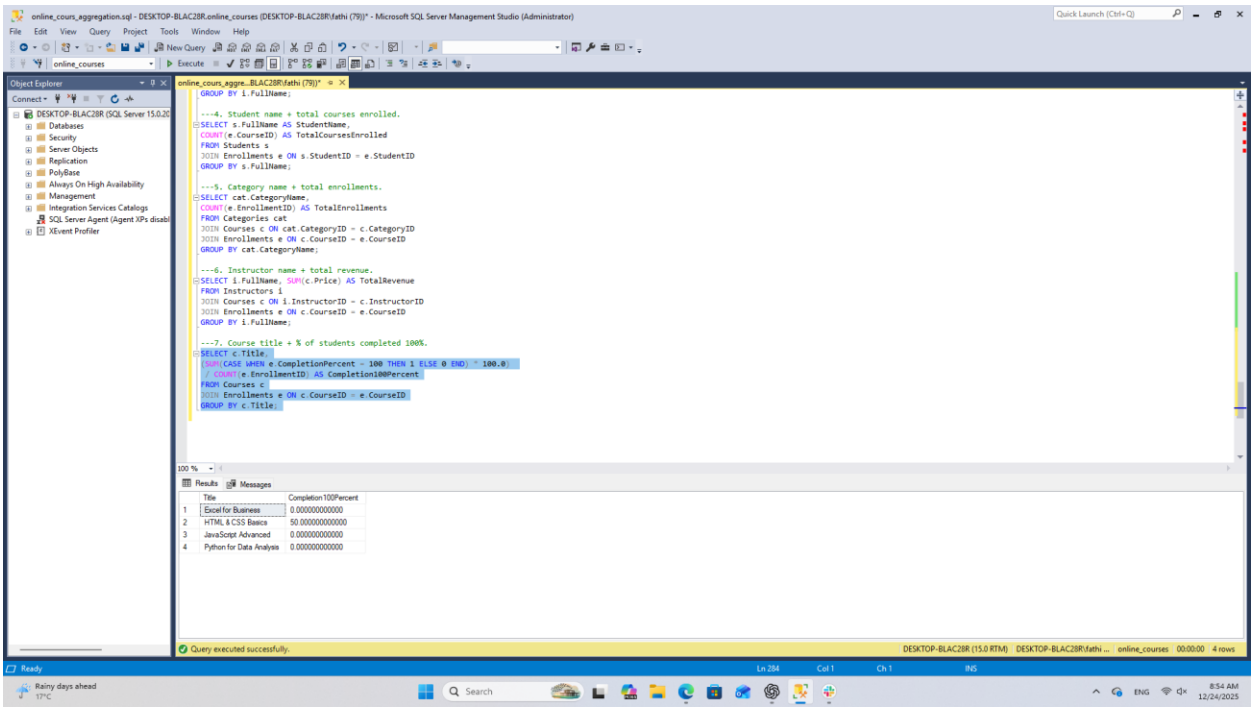

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
• Use JOIN with aggregation
• Apply HAVING correctly
• Think like a data analyst*/
SELECT c.Title,
AVG(e.Rating) AS AvgRating
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING AVG(e.Rating) >= 4;
---part 3---
---1. Course title + instructor name + enrollments.
SELECT c.Title AS CourseTitle, i.FullName AS InstructorName,
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Courses c
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 AveragePrice
FROM Categories cat
LEFT JOIN Courses c ON cat.CategoryID = c.CategoryID
GROUP BY cat.CategoryName;
Results
Messages
CategoryName TotalCourses AveragePrice
1 Business 1 19.990000
2 Data Science 1 49.990000
3 Web Development 2 34.990000
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\fathi ... online_courses 00:00:00 3 rows
Ready
Ln 245 Col 1 Ch 1 INS
8:48 AM 12/24/2025

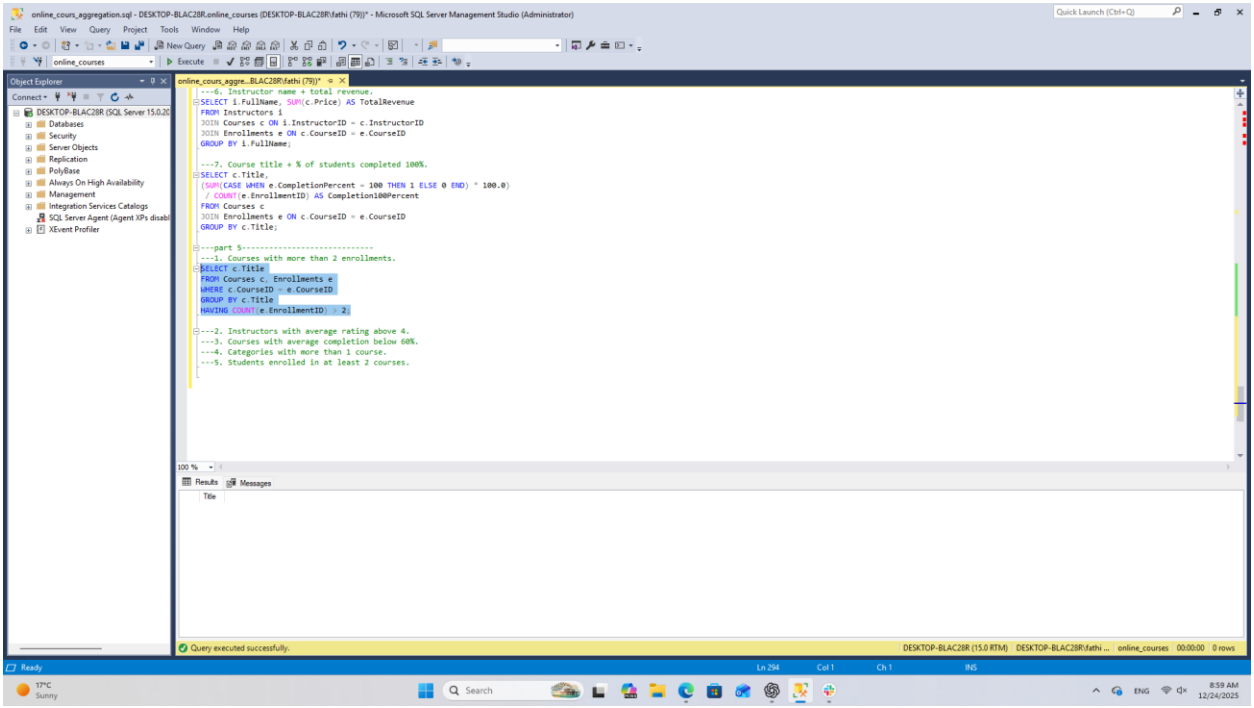

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING AVG(e.Rating) >= 4;
---part 3----------
---1. Course title + instructor name + enrollments.
SELECT c.Title AS CourseTitle, i.FullName AS InstructorName,
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Courses c
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 AveragePrice
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 AverageRating
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName;
---4. Student name + total courses enrolled.
---5. Category name + total enrollments.
---6. Instructor name + total revenue.
---7. Course title + % of students completed 100%.
Results
Messages
InstructorName
AverageRating
1 Mohammed Al-Busaidi 3
2 Sarah Ahmed 3
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
2 rows
Ready
Ln 254
Col 1
Ch 1
INS
8:50 AM
12/24/2025

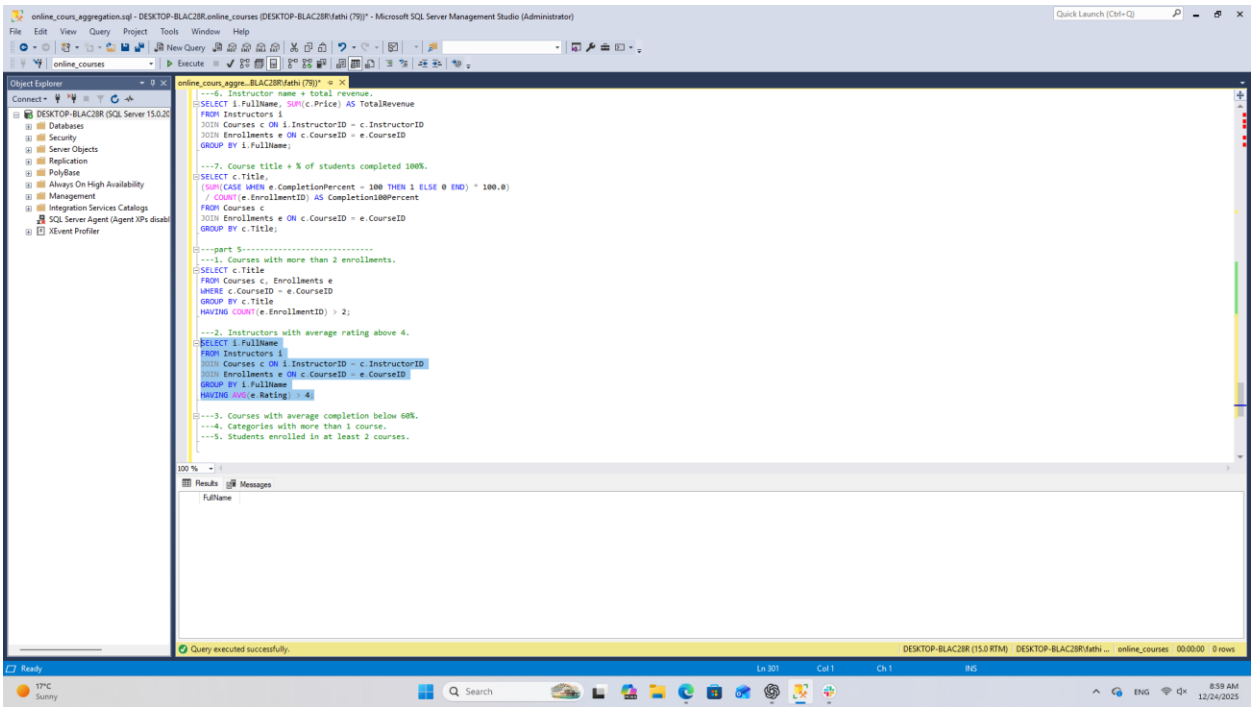

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Courses c
JOIN Instructors i ON c.InstructorID = i.InstructorID
LEFT JOIN Enrollments e ON c.CourseID = e.CourseID
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 AveragePrice
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 AverageRating
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName;
---4. Student name + total courses enrolled.
SELECT s.FullName AS StudentName,
COUNT(e.CourseID) AS TotalCoursesEnrolled
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName;
---5. Category name + total enrollments.
---6. Instructor name + total revenue.
---7. Course title + % of students completed 100%.
Results
Messages
StudentName
TotalCoursesEnrolled
1 Ahmed Said 2
2 Ali Salim 3
3 Layla Nasser 2
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
3 rows
Ready
Ln 262
Col 1
Ch 1
INS
8:51 AM
12/24/2025

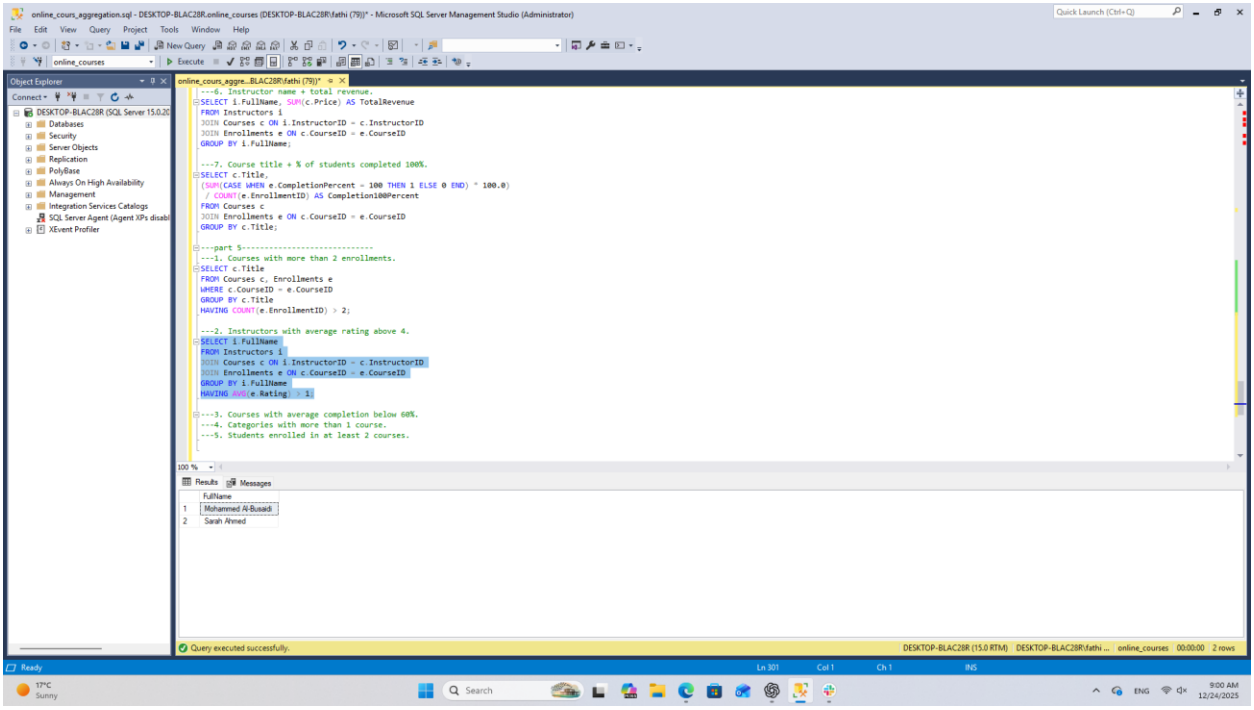

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
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 AverageRating
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName;
---4. Student name + total courses enrolled.
SELECT s.FullName AS StudentName,
COUNT(e.CourseID) AS TotalCoursesEnrolled
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName;
---5. Category name + total enrollments.
SELECT cat.CategoryName,
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Categories cat
JOIN Courses c ON cat.CategoryID = c.CategoryID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY cat.CategoryName;
---6. Instructor name + total revenue.
---7. Course title + % of students completed 100%.
100 %
Results
Messages
CategoryName TotalEnrollments
1 Business 1
2 Data Science 2
3 Web Development 4
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\fathi ... online_courses 00:00:00 3 rows
Ready
Ln 269 Col 1 Ch 1 INS
Rainy days ahead
17°C
Search
ENG
8:51 AM
12/24/2025

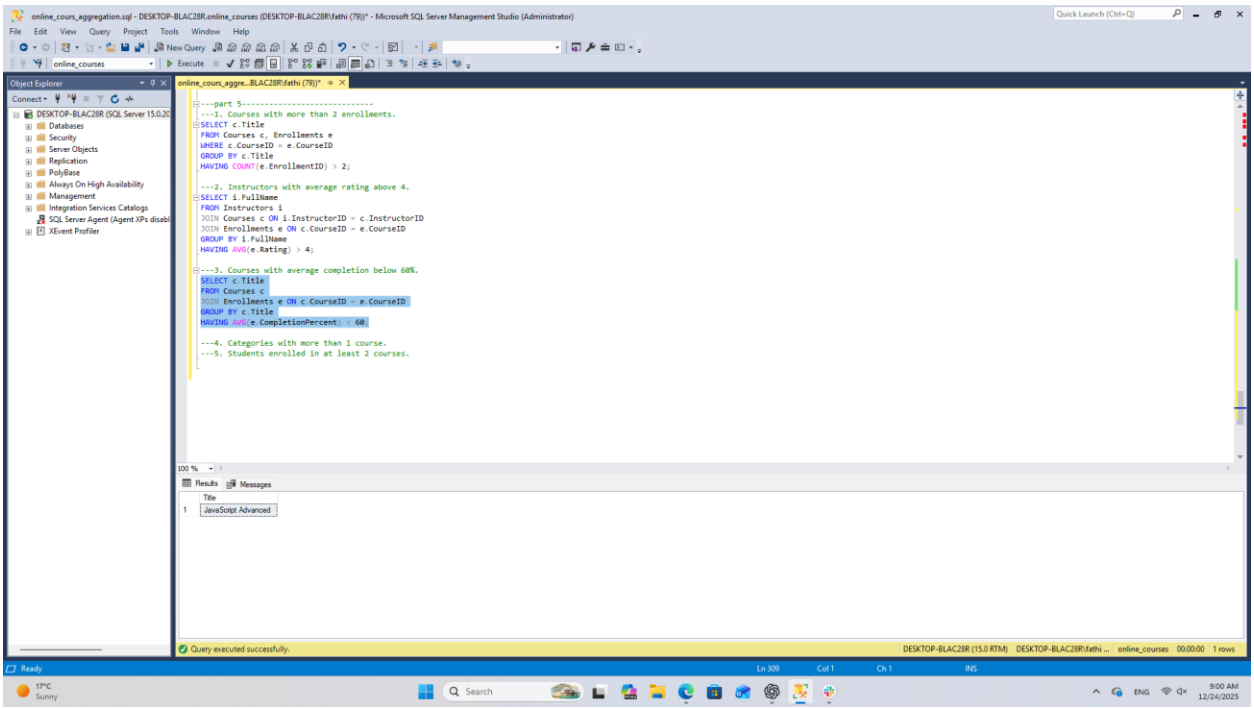

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
GROUP BY i.FullName;
---4. Student name + total courses enrolled.
SELECT s.FullName AS StudentName,
COUNT(e.CourseID) AS TotalCoursesEnrolled
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName;
---5. Category name + total enrollments.
SELECT cat.CategoryName,
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Categories cat
JOIN Courses c ON cat.CategoryID = c.CategoryID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY cat.CategoryName;
---6. Instructor name + total revenue.
SELECT i.FullName, SUM(c.Price) AS TotalRevenue
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY i.FullName
---7. Course title + % of students completed 100%.
100 %
Results
Messages
FullName TotalRevenue
1 Mohammed Al-Busaidi 119.97
2 Sarah Ahmed 139.96
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\fathi ... online_courses 00:00:00 2 rows
Ready
Ln 277 Col 1 Ch 1 INS
Rainy days ahead
17°C
Search
ENG
8:53 AM
12/24/2025

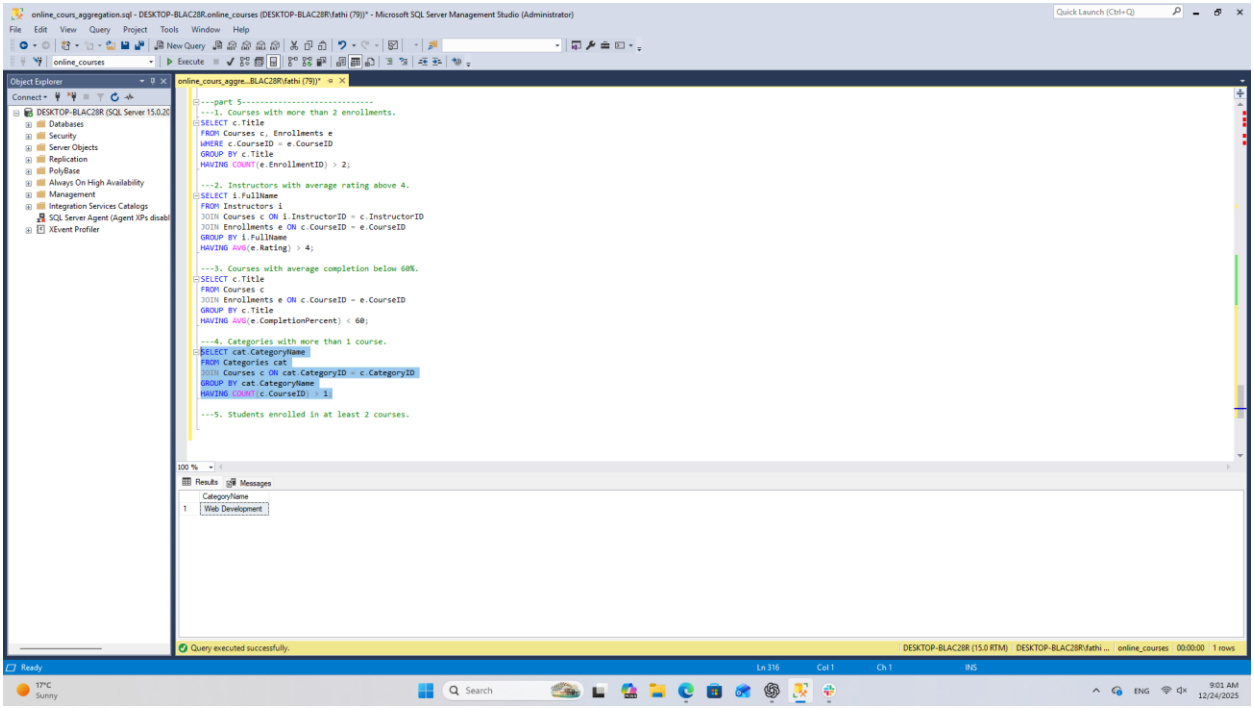

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
GROUP BY i.FullName;
---4. Student name + total courses enrolled.
SELECT s.FullName AS StudentName,
COUNT(e.CourseID) AS TotalCoursesEnrolled
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName;
---5. Category name + total enrollments.
SELECT cat.CategoryName,
COUNT(e.EnrollmentID) AS TotalEnrollments
FROM Categories cat
JOIN Courses c ON cat.CategoryID = c.CategoryID
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY cat.CategoryName;
---6. Instructor name + total revenue.
SELECT i.FullName, SUM(c.Price) AS TotalRevenue
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
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 Completion100Percent
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
Results
Messages
Title
Completion100Percent
1 Excel for Business 0.000000000000
2 HTML & CSS Basics 50.000000000000
3 JavaScript Advanced 0.000000000000
4 Python for Data Analysis 0.000000000000
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
4 rows
Ready
Ln 284
Col 1
Ch 1
INS
Rainy days ahead
17°C
Search
ENG
8:54 AM
12/24/2025

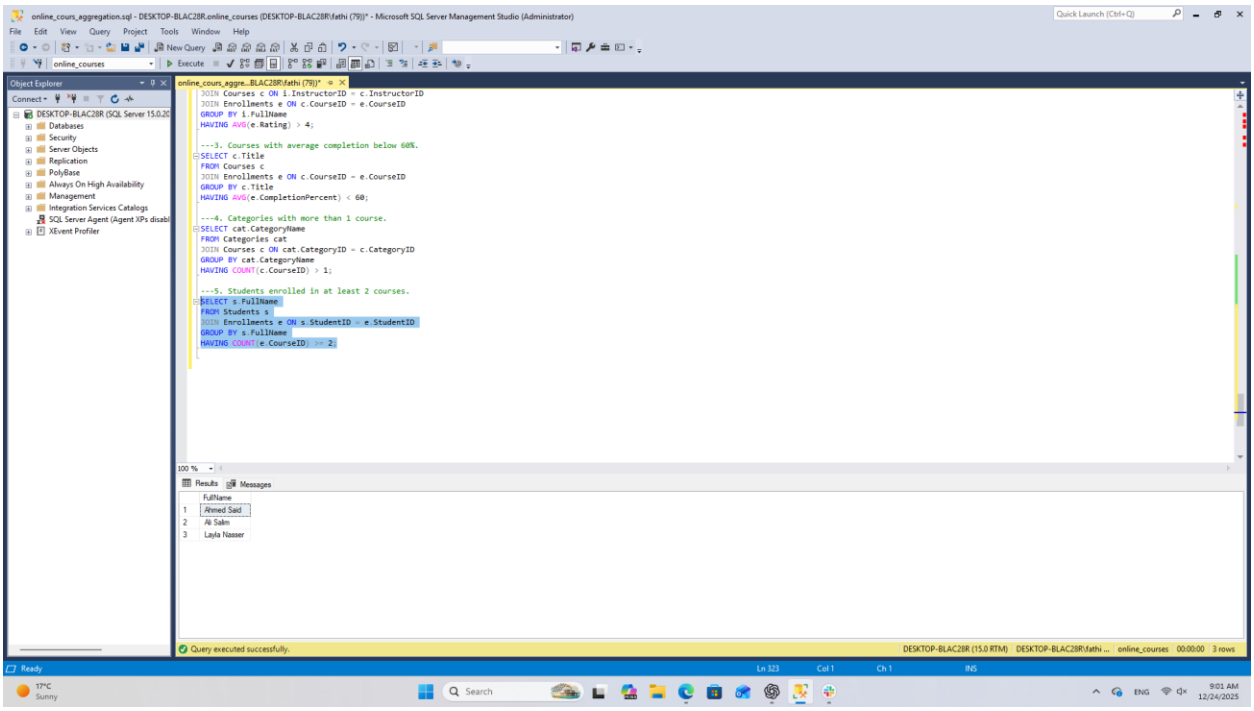

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---6. Instructor name + total revenue.
SELECT i.FullName, SUM(c.Price) AS TotalRevenue
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
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 Completion100Percent
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---part 5----------------------------
---1. Courses with more than 2 enrollments.
SELECT c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING COUNT(e.EnrollmentID) > 2;
---2. Instructors with average rating above 4.
---3. Courses with average completion below 60%.
---4. Categories with more than 1 course.
---5. Students enrolled in at least 2 courses.
Results
Messages
Title
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
0 rows
Ready
Ln 294
Col 1
Ch 1
INS
17°C
Sunny
Search
ENG
8:59 AM
12/24/2025

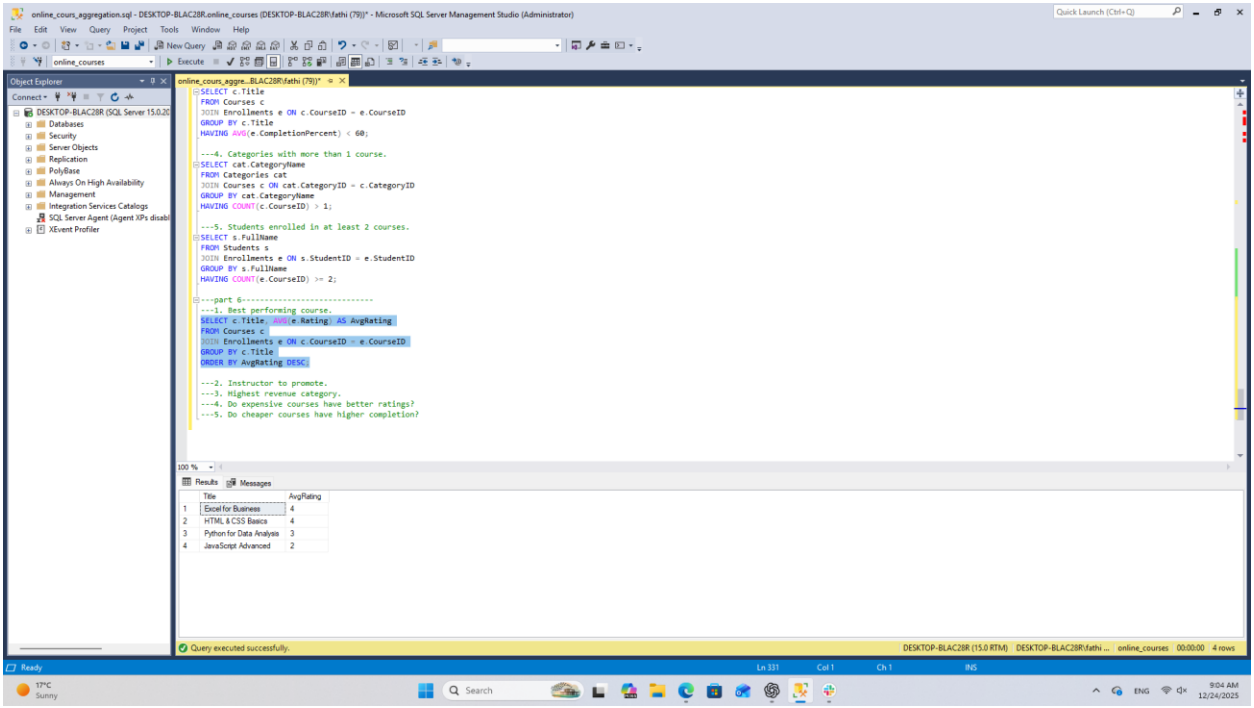

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
---6. Instructor name + total revenue.
SELECT i.FullName, SUM(c.Price) AS TotalRevenue
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
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 Completion100Percent
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---part 5----------------------------
---1. Courses with more than 2 enrollments.
SELECT c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING COUNT(e.EnrollmentID) > 2;
---2. Instructors with average rating above 4.
SELECT i.FullName
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%.
---4. Categories with more than 1 course.
---5. Students enrolled in at least 2 courses.
100 %
Results
Messages
FullName
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
0 rows
Ready
Ln 301
Col 1
Ch 1
INS
17°C
Sunny
Search
ENG
8:59 AM
12/24/2025


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
---6. Instructor name + total revenue.
SELECT i.FullName, SUM(c.Price) AS TotalRevenue
FROM Instructors i
JOIN Courses c ON i.InstructorID = c.InstructorID
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 Completion100Percent
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title;
---part 5----------------------------
---1. Courses with more than 2 enrollments.
SELECT c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING COUNT(e.EnrollmentID) > 2;
---2. Instructors with average rating above 4.
SELECT i.FullName
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) > 1
---3. Courses with average completion below 60%.
---4. Categories with more than 1 course.
---5. Students enrolled in at least 2 courses.
100 %
Results
Messages
FullName
1 Mohammed Al-Busaidi
2 Sarah Ahmed
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
2 rows
Ready
Ln 301
Col 1
Ch 1
INS
17°C
Sunny
Search
ENG
9:00 AM
12/24/2025

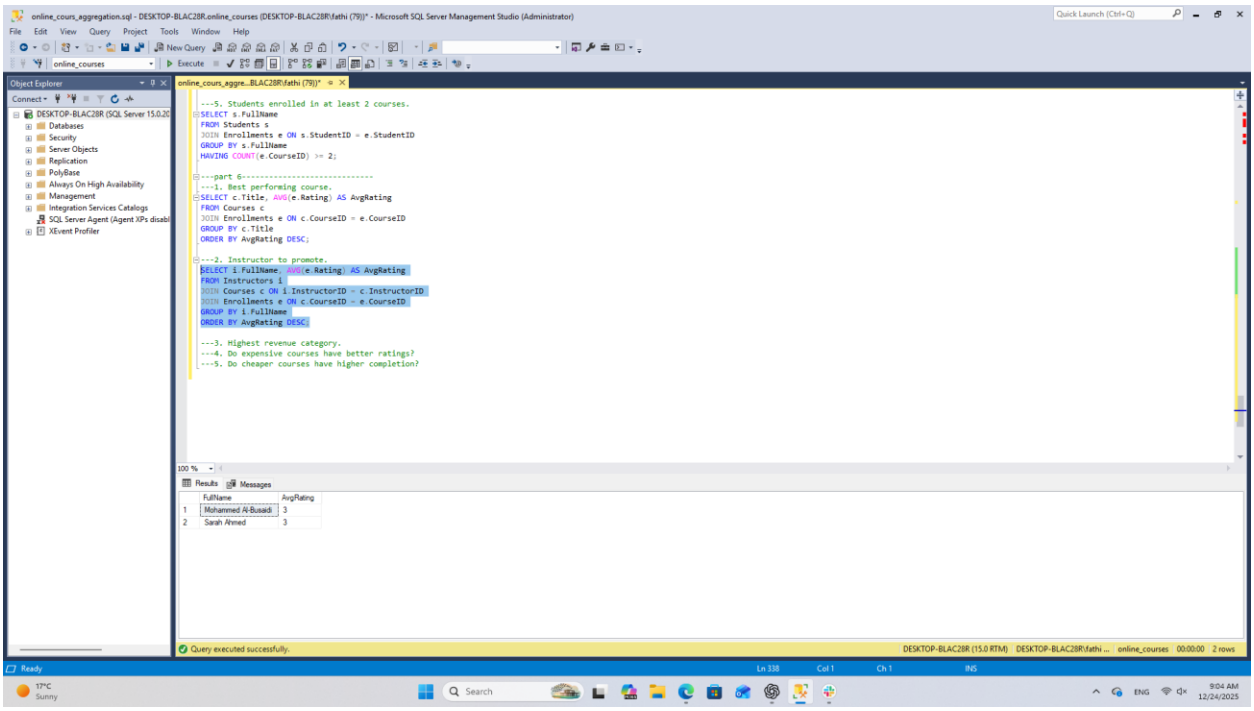

online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
---part 5----------
---1. Courses with more than 2 enrollments.
SELECT c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING COUNT(e.EnrollmentID) > 2;
---2. Instructors with average rating above 4.
SELECT i.FullName
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
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.
---5. Students enrolled in at least 2 courses.
100 %
Results
Messages
Title
1
JavaScript Advanced
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
1 rows
Ready
Ln 309
Col 1
Ch 1
INS
17°C
Sunny
Search
ENG
9:00 AM
12/24/2025


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
online_courses
Object Explorer
Connect
DESKTOP-BLAC28R (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
online_cours_aggre...BLAC28R\fathi (79))*
---part 5----------
---1. Courses with more than 2 enrollments.
SELECT c.Title
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID
GROUP BY c.Title
HAVING COUNT(e.EnrollmentID) > 2;
---2. Instructors with average rating above 4.
SELECT i.FullName
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
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.
SELECT cat.CategoryName
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.
100 %
Results
Messages
CategoryName
1
Web Development
Query executed successfully.
DESKTOP-BLAC28R (15.0 RTM)
DESKTOP-BLAC28R\fathi ...
online_courses
00:00:00
1 rows
Ready
Ln 316
Col 1
Ch 1
INS
17°C
Sunny
Search
ENG
9:01 AM
12/24/2025

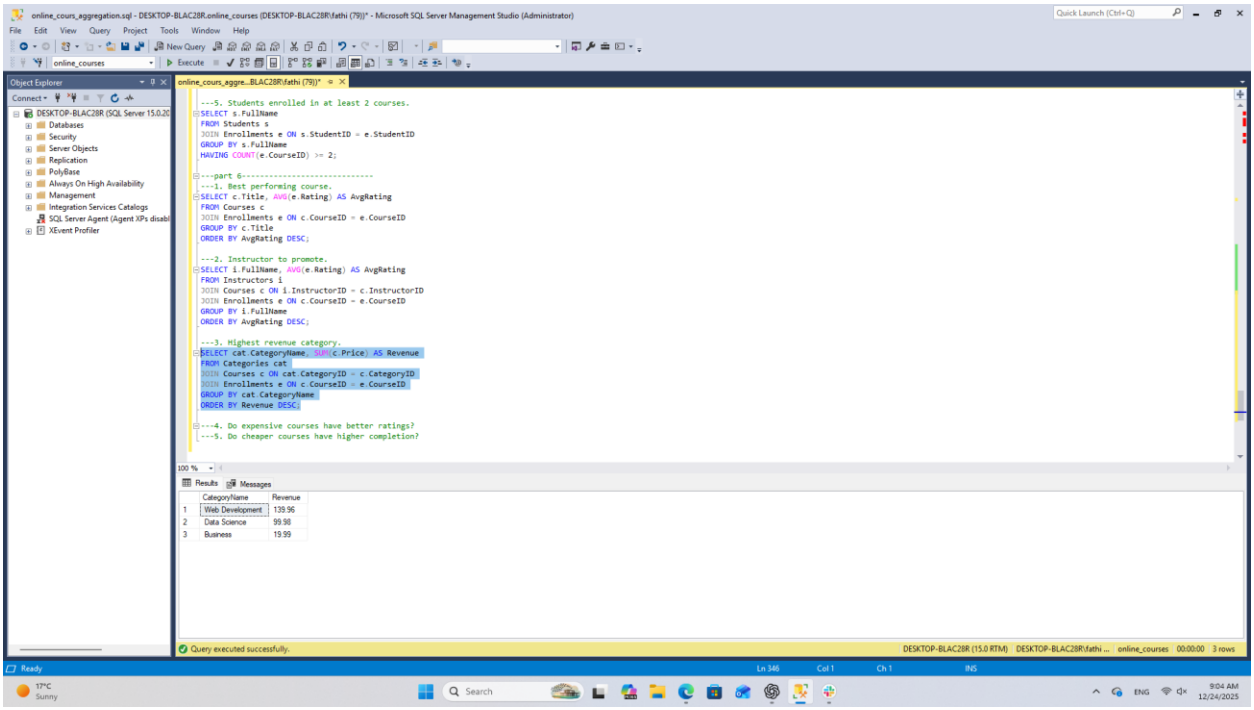




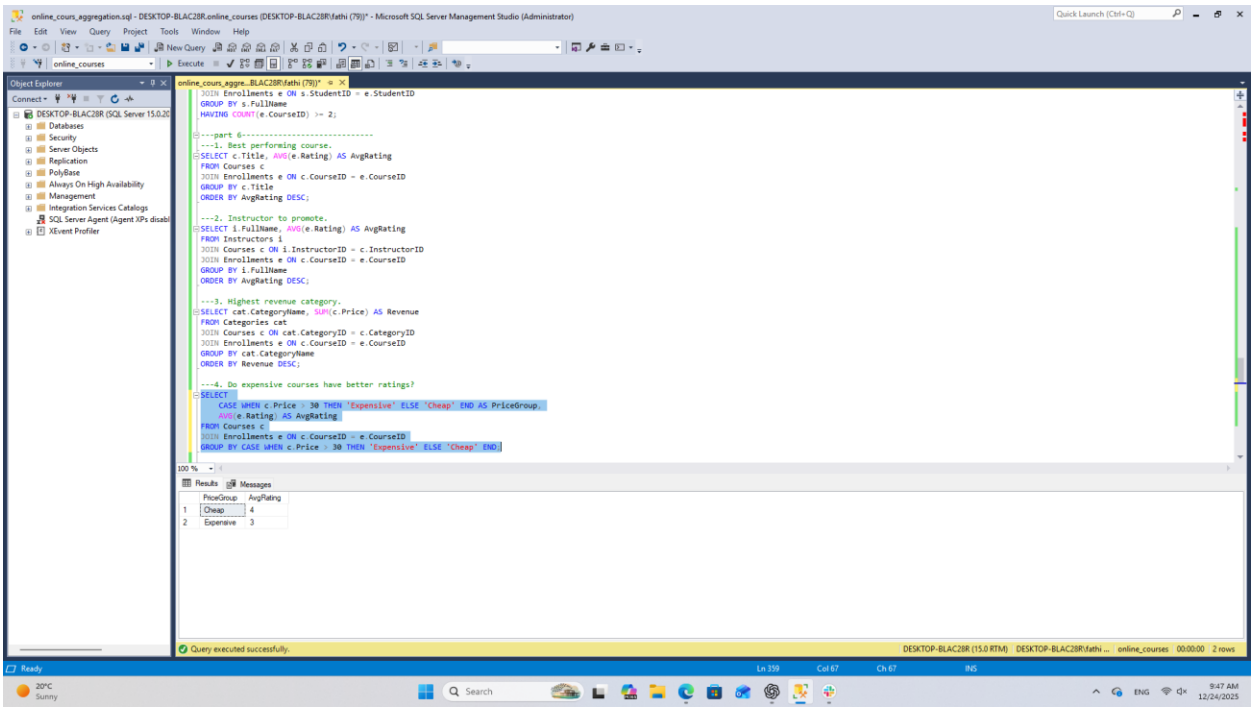
Explanation: Highest average rating.



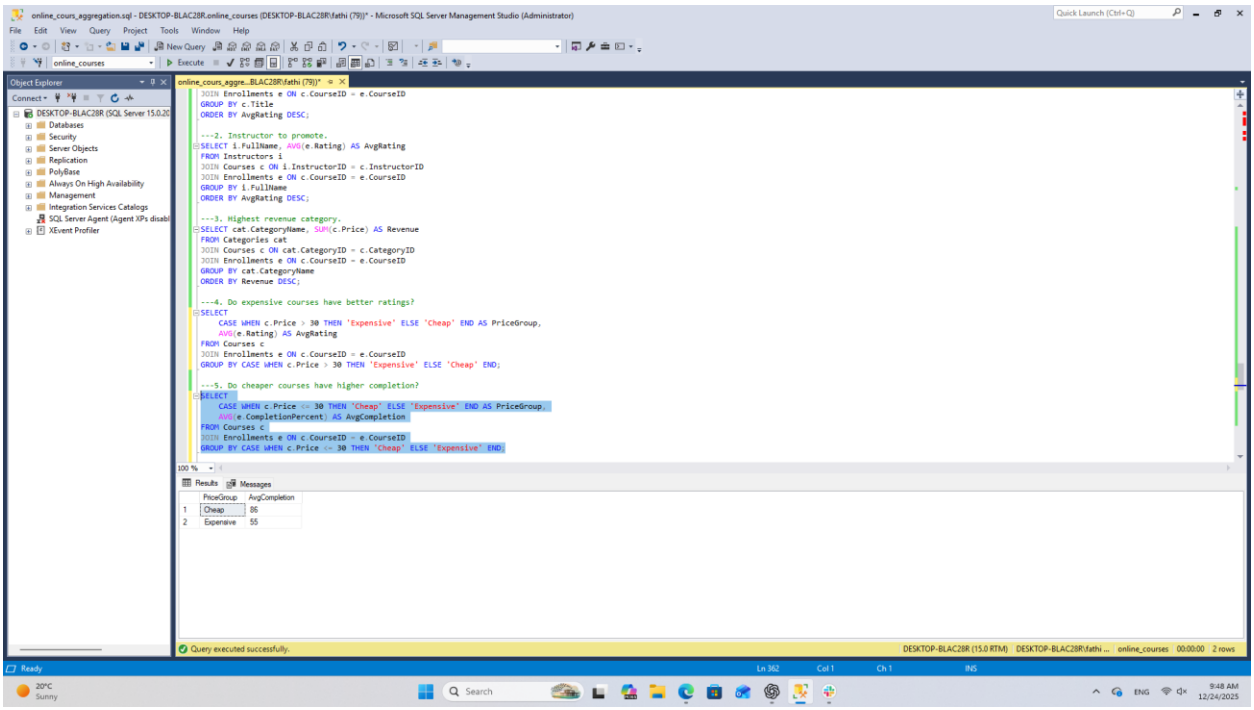
Explanation: Instructor with best student feedback.



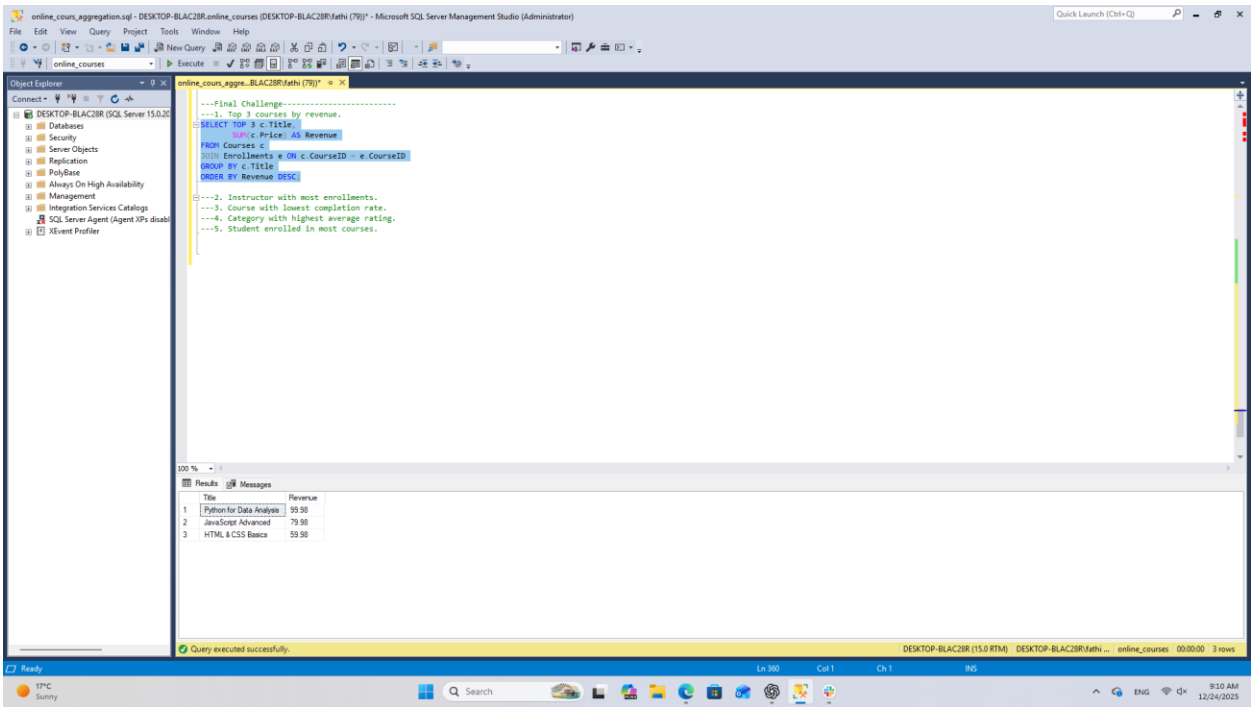
Explanation: Category earning the most money.

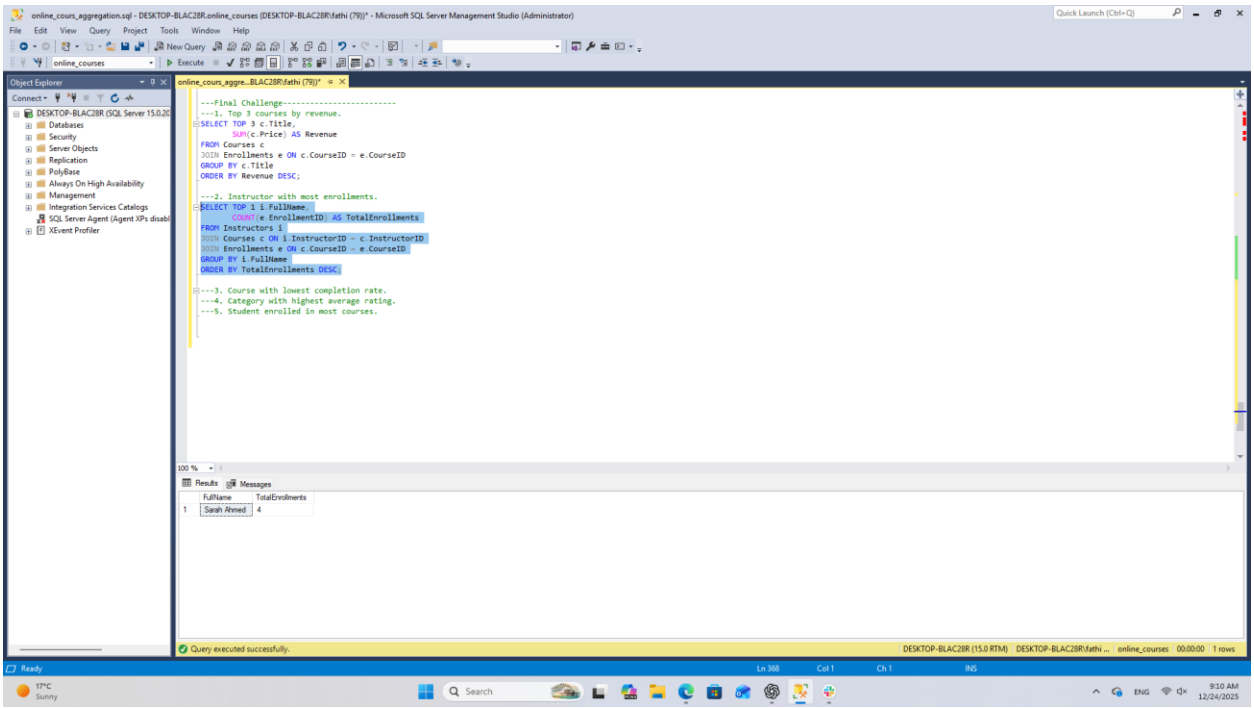


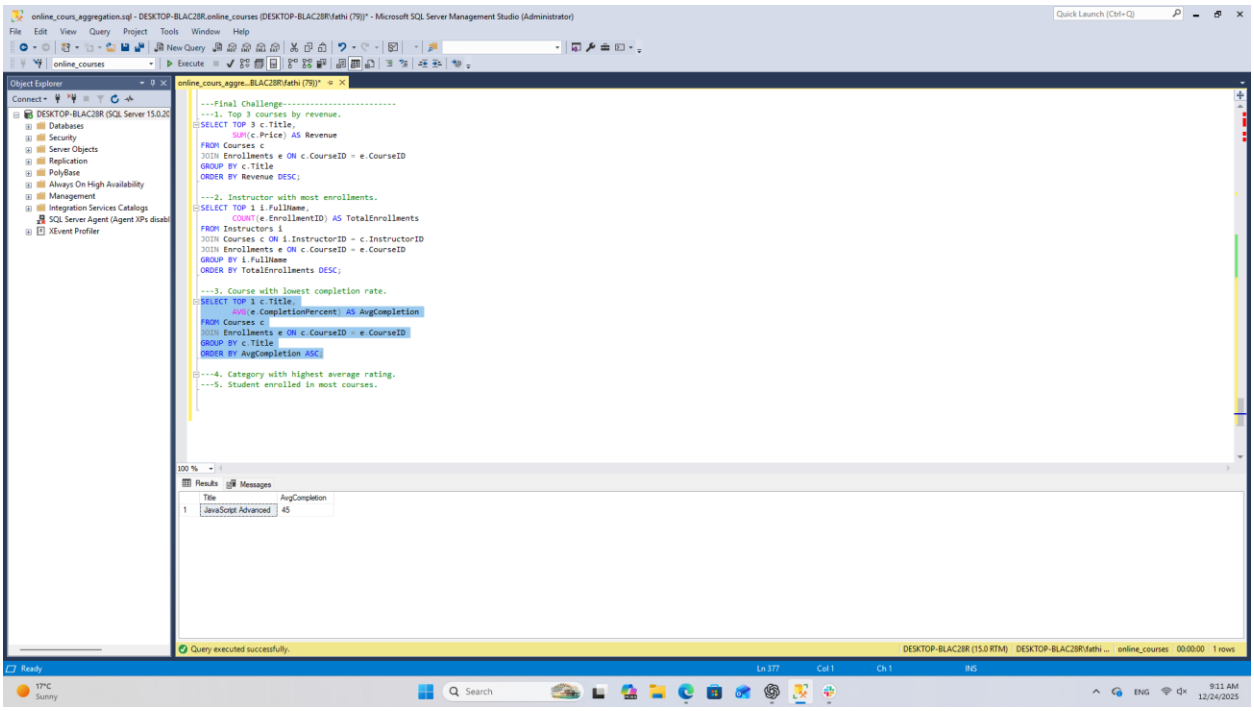
Explanation: Compares ratings by price group.

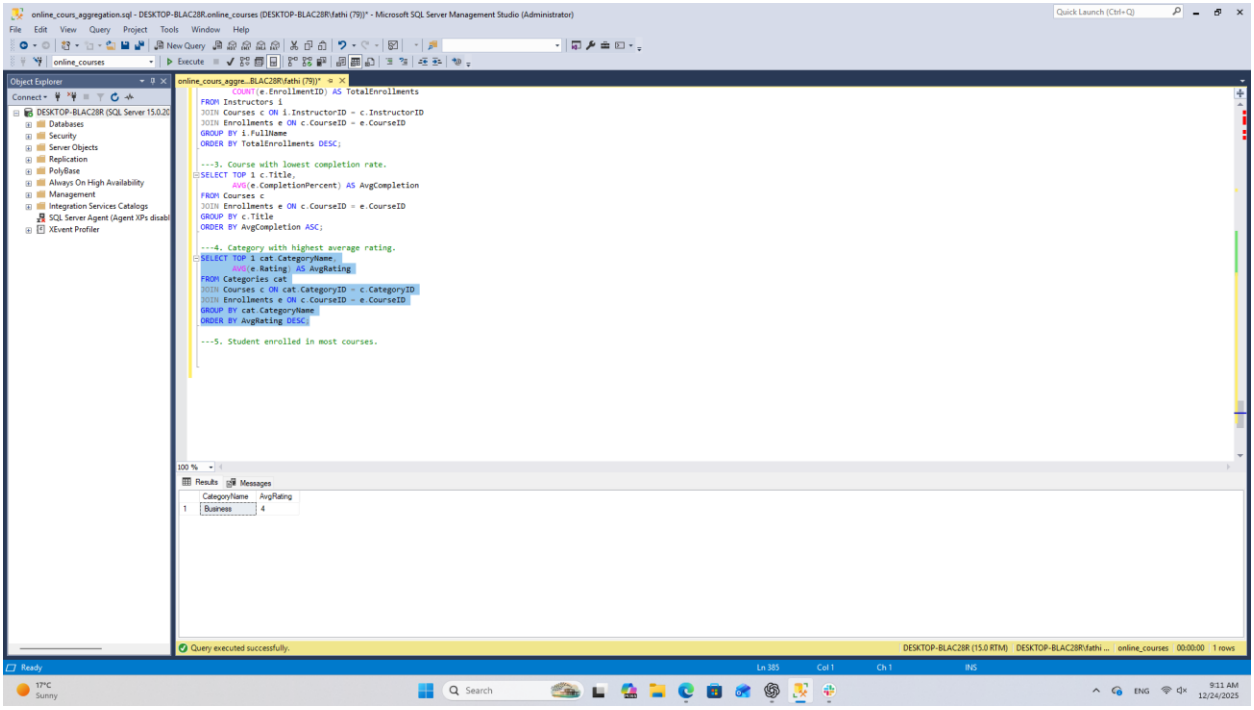


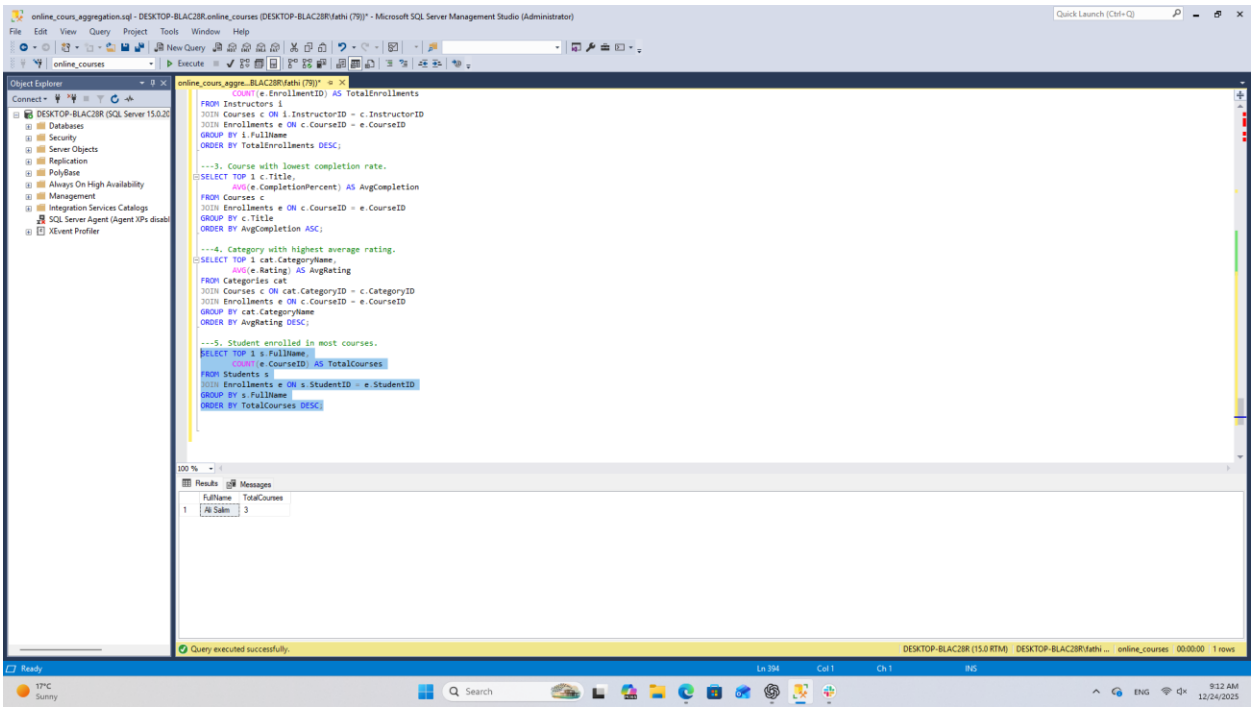
Explanation: Compares completion based on price.


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---Final Challenge-----------------------
---1. Top 3 courses by revenue.
SELECT TOP 3 c.Title,
SUM(c.Price) AS Revenue
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY Revenue DESC;
---2. Instructor with most enrollments.
---3. Course with lowest completion rate.
---4. Category with highest average rating.
---5. Student enrolled in most courses.
Title
Revenue
1 Python for Data Analysis 99.98
2 JavaScript Advanced 79.98
3 HTML & CSS Basics 59.98
Query executed successfully.


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---Final Challenge-----------------------
---1. Top 3 courses by revenue.
SELECT TOP 3 c.Title,
SUM(c.Price) AS Revenue
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY Revenue DESC;
---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 TotalEnrollments DESC;
---3. Course with lowest completion rate.
---4. Category with highest average rating.
---5. Student enrolled in most courses.
FullName
TotalEnrollments
1 Sarah Ahmed 4
Query executed successfully.


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
---Final Challenge----------
---1. Top 3 courses by revenue.
SELECT TOP 3 c.Title,
SUM(c.Price) AS Revenue
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY Revenue DESC;
---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 TotalEnrollments DESC;
---3. Course with lowest completion rate.
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 AvgCompletion ASC;
---4. Category with highest average rating.
---5. Student enrolled in most courses.
Results
Messages
Title
AvgCompletion
1
JavaScript Advanced
45
Query executed successfully.
Ready


online_cours_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fathi (79))* - Microsoft SQL Server Management Studio (Administrator)
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 TotalEnrollments DESC;
---3. Course with lowest completion rate.
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 AvgCompletion ASC;
---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 AvgRating DESC;
---5. Student enrolled in most courses.
Results
Messages
CategoryName
AvgRating
1
Business
4
Query executed successfully.
Ready



At conclusion:

The **online_courses** database models an e-learning platform, storing information about **instructors, students, courses, categories, and enrollments**. It allows analysis of course performance, student engagement, instructor activity, and revenue using SQL queries with **joins, aggregations, and filters**.