

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```

-- Web Development --
(1, 'Data Science'),
(2, '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', 3, 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', 99, 4),
(3, 203, 103, '2023-04-20', 98, 3),
(4, 201, 102, '2023-04-22', 99, 3),
(5, 202, 103, '2023-04-25', 79, 4),
(6, 203, 104, '2023-04-28', 79, 2),
(7, 201, 104, '2023-05-01', 68, 3);

-- Aggregation Functions-----
-- part 1 -----
-- 1. Display all courses with prices.
--SELECT Title, Price
FROM Courses

```

Query executed successfully.

Results Messages

Title	Price
HTML & CSS Basics	29.99
Python for Data Analysis	49.99
Excel for Business	19.99
JavaScript Advanced	39.99

LN 62 Col 1 Ch 1 INS

DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\fatih ... online\_courses 00:00:00 4 rows

Ready 29°C Sunny

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```

-- Web Development --
(1, 'Data Science'),
(2, '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', 3, 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', 99, 4),
(3, 203, 101, '2023-04-20', 99, 4),
(4, 201, 102, '2023-04-22', 99, 3),
(5, 202, 103, '2023-04-25', 79, 4),
(6, 203, 104, '2023-04-28', 79, 2),
(7, 201, 104, '2023-05-01', 68, 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

```

Query executed successfully.

Results Messages

FullName	JoinDate
Ali Salim	2023-04-01
Layla Nasser	2023-04-05
Ahmed Said	2023-04-10

LN 86 Col 1 Ch 1 INS

DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\fatih ... online\_courses 00:00:00 3 rows

Ready 29°C Sunny

SQLQuery1.sql - DESKTOP-BLAC20R.online\_courses (DESKTOP-BLAC20R\fatih [15]) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

online\_courses

Object Explorer

Connect to: DESKTOP-BLAC20R (SQL Server 15.0.0)

SQL Database Security Server Objects Replication Polybase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler

SQLQuery1 Log - DE\_BLA20R\fatih [15] > [184] [JavaScript Advanced], 1, 1, 59.99, 2023-05-02 ];

INSERT INTO Students VALUES  
(281, 'Ali Salim', 'ali@student.com', '2023-04-01'),  
(282, 'Layla Nasser', 'layla@student.com', '2023-04-05'),  
(283, 'Ahmed Said', 'ahmed@student.com', '2023-04-18');

INSERT INTO Enrollments VALUES  
(1, 281, 101, '2023-04-10', 100, 5),  
(2, 281, 102, '2023-04-10', 90, 4),  
(3, 282, 101, '2023-04-28', 80, 4),  
(4, 281, 102, '2023-04-22', 50, 3),  
(5, 282, 103, '2023-04-25', 70, 4),  
(6, 281, 104, '2023-05-01', 90, 2),  
(7, 281, 104, '2023-05-01', 60, 3);

=====

--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;

Results Messages

EnrollmentID	CompletionPercent	Rating
1	80	4
2	90	4
3	50	3
4	70	4
5	30	2
6	60	3
7		

Query executed successfully.

DESKTOP-BLAC20R (15.0 RTM) DESKTOP-BLAC20R\fatih ... online\_courses 00:00:00 7 rows

Ready 24C Sunny

Ln E7 Col 1 Ch 1 RS

12:47 PM 12/23/2023 ENG

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to DESKTOP-BLAC2R\online\_courses (Windows Authentication) on Microsoft SQL Server 15.0.2022.0. The main window has a 'File' menu with options like 'Edit', 'View', 'Query', 'Project', 'Tools', 'Window', and 'Help'. Below the menu is a toolbar with icons for 'New Query', 'New Item', 'Save', 'Execute', and 'Copy'. The 'Object Explorer' sidebar on the left shows the database structure, including 'DESKTOP-BLAC2R (SQL Server 15.0.2022.0)', 'Databases', 'Security', 'Server Objects', 'Replication', 'File and Filegroup', 'Always On High Availability', 'Management', and 'Integration Services Catalogs'. A 'SQL Server Agent (Agent XPs disabled)' node is expanded, and 'XEvent Profiler' is listed under it. The central pane displays a query window titled 'SQLQuery1.sql - DE\_BLA... (Blah Blah)'. The code in the window is as follows:

```
--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
```

The results pane at the bottom shows the output of the fourth query:

TotalInstructors_2023
2

A status bar at the bottom right shows the system date and time: '12/23/2023 00:00:00 1 rows'.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses [DESKTOP-BLAC28R\ahati (8)] \* Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

Object Explorer

SQLQuery1Log - DE\_BLA28R\ahati (8) \* x

```
--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
3

Query executed successfully.

DESKTOP-BLAC28R (15.0 ITM) DESKTOP-BLAC28R\ahati ... online\_courses 00:00:00 1 rows

Ready 24°C Sunny

Search

Ln 103 Col 1 Ch 1 INS

12/23/2023 ENG

1102 File

```
SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...] Quick Launch (Ctrl+Q) [...]
[...] online_courses [...]
Object Explorer
Connect - 4 24C
File Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
SQLQuery1.sql - DESKTOP-BLAC28R\fatih (S1) - X
-->--1. Enrollment count per month.
SELECT COUNT(*) AS EnrollmentCount
FROM Enrollments
WHERE YEAR(JoinDate) = 2023;
-->--2. Completion percentage per month.
SELECT COUNT(*) AS TotalInstructors_2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;
-->--3. 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.
Ln 107 Col 1 Ch 1 INS
DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\fatih ... | online_courses | 00:00:00 | 1 rows
24C Sunny Search 12:05 PM 12/23/2025
```

```
SQLQuery1.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...] Quick Launch (Ctrl+Q) [...]
[...] online_courses [...]
Object Explorer
Connect - 4 24C
File Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
SQLQuery1.sql - DESKTOP-BLAC28R\fatih (S1) - X
-->--1. Enrollment count per month.
SELECT COUNT(*) AS EnrollmentCount
FROM Enrollments
WHERE YEAR(JoinDate) = 2023;
-->--2. Completion percentage per month.
SELECT COUNT(*) AS TotalInstructors_2023
FROM Instructors
WHERE YEAR(JoinDate) = 2023;
-->--3. 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 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 Java Script Advanced 2
4 Python for Data Analysis 3
Query executed successfully.
Ln 111 Col 1 Ch 1 INS
DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\fatih ... | online_courses | 00:00:00 | 4 rows
24C Sunny Search 12:09 PM 12/23/2025
```

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```
-->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	2
2	2

Query executed successfully.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```
-->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;

```

CategoryID	total_course
1	2
2	1
3	1

Query executed successfully.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - 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 enrolments.
SELECT COUNT(*) AS No_of_Enrolments
FROM Enrolments;

-->3. Find average rating per course.
SELECT c.Title, AVG(e.Rating) AS AverageRating
FROM Courses c
JOIN Enrolments 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
JOIN Enrolments 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_price
FROM Courses c, Categories cat
WHERE c.CategoryID = cat.CategoryID
GROUP BY cat.CategoryName;

```

Query executed successfully.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```

-->1. Count total number of students.
SELECT COUNT(*) AS no_of_students
FROM Students;

-->2. Count total number of enrolments.
SELECT COUNT(*) AS No_of_Enrolments
FROM Enrolments;

-->3. Find average rating per course.
SELECT c.Title, AVG(e.Rating) AS AverageRating
FROM Courses c
JOIN Enrolments e
WHERE 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
JOIN Enrolments 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_price
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.

```

Query executed successfully.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```

--1. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;

--2. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;

--3. 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;

--4. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_price
FROM Courses c, Categories cat
where c.CategoryID = cat.CategoryID
GROUP BY cat.CategoryName;

--5. Find maximum course price.
SELECT MAX(c.Price) AS max_course_price
FROM Courses c;

```

Results

max_course_price	1 49.99

Query executed successfully.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (S1)) - Microsoft SQL Server Management Studio (Administrator)

```

--1. Count courses per instructor.
SELECT c.InstructorID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.InstructorID;

--2. Count courses per category.
SELECT c.CategoryID, COUNT(c.CourseID) AS total_course
FROM Courses c
GROUP BY c.CategoryID;

--3. 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;

--4. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_price
FROM Courses c, Categories cat
where c.CategoryID = cat.CategoryID
GROUP BY cat.CategoryName;

--5. Find maximum course price.
SELECT MAX(c.Price) AS max_course_price
FROM Courses c;

--6. 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 e.CourseID = c.CourseID
GROUP BY c.Title;

```

Results

	min_Rating	max_Rating	avg_Rating
1 Excel for Business	4	4	4
2 HTML & CSS Basics	4	5	4
3 Java Script Advanced	2	3	2
4 Python for Data Analysis	3	4	3

Query executed successfully.

SQLQuery1.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\athif (8)) - Microsoft SQL Server Management Studio (Administrator)

```
--7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_price
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;
```

Results

	FiveStarRatings
1	1

Query executed successfully.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\athif (79)) - Microsoft SQL Server Management Studio (Administrator)

```
--7. Find average course price per category.
SELECT cat.CategoryName, AVG(c.Price) AS avg_course_price
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(EnrollmentDate) AS Year,
    MONTH(EnrollmentDate) AS Month,
    COUNT(*) AS TotalEnrollments
FROM Enrollments
GROUP BY YEAR(EnrollmentDate), MONTH(EnrollmentDate)
ORDER BY Year, Month;
```

Results

	Year	Month	TotalEnrollments
1	2023	4	6
2	2023	5	1

Query executed successfully.

online\_courses\_aggregation.sql - DESKTOP-BLAC20R.online\_courses (DESKTOP-BLAC20R\fathi (79)) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

online\_courses

online\_courses\_aggregation.BLA20R\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 e.CourseID = c.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
34.990000

Query executed successfully.

DESKTOP-BLAC20R (15.0 ITM) DESKTOP-BLAC20R\fathi ... online\_courses 00:00:00 1 rows

Ready

17°C Mostly sunny

Search

LN 165 Col 1 Ch 1 RS

8:13 AM 12/24/2023 ENG

online\_courses\_aggregation.sql - DESKTOP-BLAC20R.blac20r.online\_courses (DESKTOP-BLAC20R\fatih (%)) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

online\_courses

Object Explorer

Connect to: DESKTOP-BLAC20R (SQL Server 15.0.2600.6) online\_courses

online\_courses\_aggregation.BLA20R(fatih (%))

--> 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 Courses c
INNER JOIN Enrollments e
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 TotalInrollments
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-BLAC20R (15.0 ITM) DESKTOP-BLAC20R\fatih (%)) online\_courses 00:00:00 1 rows

Ready

17°C Mostly sunny

Search

Ln 169 Col 1 Ch 1 INS

8:34 AM 12/24/2023

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\ath(79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...]
[...] online_courses
Object Explorer
[...] DESKTOP-BLAC28R (SQL Server 15.0.2000.6)
[...] Databases
[...] Security
[...] Server Objects
[...] Replication
[...] PolyBase
[...] Always On High Availability
[...] Management
[...] Integration Services Catalogs
[...] SQL Server Agent (Agent XPs disabled)
[...] XEvent Profiler
online_courses_aggregation.sql - DESKTOP-BLAC28R\ath(79) - x
--11. 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 TotalEnrollment
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;

100 %
Results Messages
Rating RatingCount
1 2 1
2 3 2
3 4 3
4 5 1

Query executed successfully.

```

DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\ath(79) | online\_courses | 00:00:00 | 4 rows

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\ath(79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...]
[...] online_courses
Object Explorer
[...] DESKTOP-BLAC28R (SQL Server 15.0.2000.6)
[...] Databases
[...] Security
[...] Server Objects
[...] Replication
[...] PolyBase
[...] Always On High Availability
[...] Management
[...] Integration Services Catalogs
[...] SQL Server Agent (Agent XPs disabled)
[...] XEvent Profiler
online_courses_aggregation.sql - DESKTOP-BLAC28R\ath(79) - x
--11. Count how many students gave rating = 5.
SELECT 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;

100 %
Results Messages
CourseID Title
102 Python for Data Analysis
103 Excel for Business
104 Java Script Advanced

Query executed successfully.

```

DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\ath(79) | online\_courses | 00:00:00 | 3 rows

Ready 77°C Mostly sunny 8:16 AM 12/24/2025

```
online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\ath (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query Execute Find All Go To Object Explorer Properties Task List Results Messages
online_courses
Object Explorer
Connect - Help
File Desktop-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses_aggregation.BLA... (79)*
-->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.

```
online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\ath (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query Execute Find All Go To Object Explorer Properties Task List Results Messages
online_courses
Object Explorer
Connect - Help
File Desktop-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses_aggregation.BLA... (79)*
-->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;
```

Query executed successfully.

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)". The main area contains a query window with the following SQL code:

```

--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;

--18. Find courses with lowest average rating.
SELECT TOP 1 c.CourseID, c.Title,
    AVG(e.Rating) AS AvgRating
FROM Courses c, Enrollments e
WHERE c.CourseID = e.CourseID AND e.Rating IS NOT NULL
GROUP BY c.CourseID, c.Title
ORDER BY AvgRating ASC;

```

The results pane shows a single row of data from the last query:

	CourseID	Title	AvgRating
1	104	JavaScript Advanced	2

At the bottom of the screen, a status bar indicates "Query executed successfully." and shows system information like "DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\lathi ... online\_courses | 00:00:00 | 1 rows".

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**.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

```

Object Explorer  online_courses_aggregation.sql - [x]  online_courses
Connect  File  Edit  View  Query  Project  Tools  Window  Help
[...]  New Query  Execute  [...]
[...]  online_courses  Quick Launch (Ctrl+Q)  x

```

```

online_courses_aggregation.sql (79)  + x
FINDN Courses
WHERE Price > 30;
-->17. Find average completion percentage.
SELECT
AVG(CompletionPercent) AS AverageCompletionPercent
FROM Enrollments;
-->18. Find course with lowest average rating.
SELECT TOP 1 c.CourseID, c.Title,
AVG(e.Rating) AS AvgRating
FROM Courses c
JOIN Enrollments e
WHERE c.CourseID = e.CourseID AND e.Rating IS NOT NULL
GROUP BY c.CourseID, c.Title
ORDER BY AvgRating ASC;
-->19. Course Performance Snapshot Show!
• Course title
• Total enrolments
• Average rating
• Average completion %
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;

```

Results Messages

CourseTitle	TotalEnrollments	AverageRating	AverageCompletion
Excel for Business	1	4	70
HTML & CSS Basics	2	4	95
JavaScript Advanced	2	2	45
Python for Data Analysis	2	3	65

Query executed successfully.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

```

Object Explorer  online_courses_aggregation.sql - [x]  online_courses
Connect  File  Edit  View  Query  Project  Tools  Window  Help
[...]  New Query  Execute  [...]
[...]  online_courses  Quick Launch (Ctrl+Q)  x

```

```

online_courses_aggregation.sql (79)  + x
Total enrolments
• Average rating
• Average completion %
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
-->19. Courses with average rating >= 4
• Use JOIN with aggregation
• Apply HAVING correctly
• Think like a data analyst!
SELECT
c.Title AS 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;

```

Results Messages

Title	AvgRating
Excel for Business	4
HTML & CSS Basics	4

Query executed successfully.

Courses with **average rating  $\geq 4$**

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...] Quick Launch (Ctrl+Q) [...]
[...] online_courses [...]
Object Explorer - 0 X
online_courses_aggregation.sql - [ ] SELECT c.Title AS CourseTitle
--SELECT c.Title AS CourseTitle
--COUNT(e.EnrollmentID) AS TotalEnrollments,
--AVG(e.Rating) AS AverageRating,
--AVG(e.Completion) 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 AS CourseTitle
--JOIN Courses c, Enrollments e
--WHERE c.CourseID = e.CourseID
--GROUP BY c.Title
--HAVING COUNT(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;
--Query executed successfully.
Ln 238 Col 1 Ch 1 INS
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\lathi ... online_courses 00:00:00 4 rows
Ready 17°C Mostly sunny Search 8:47 AM 12/24/2025

```

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...] Quick Launch (Ctrl+Q) [...]
[...] online_courses [...]
Object Explorer - 0 X
online_courses_aggregation.sql - [ ] * Use JOIN with aggregation
-- * Apply HAVING correctly
-- * Think like a data analyst!
--SELECT c.Title AS CourseTitle
--AVG(e.Rating) AS AvgRating
--FROM Courses c
--JOIN 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
--FROM Categories cat
--LEFT JOIN Courses c ON cat.CategoryID = c.CategoryID
--GROUP BY cat.CategoryName;
--Query executed successfully.
Ln 245 Col 1 Ch 1 INS
DESKTOP-BLAC28R (15.0 RTM) DESKTOP-BLAC28R\lathi ... online_courses 00:00:00 3 rows
Ready 17°C Mostly sunny Search 8:48 AM 12/24/2025

```

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...]
[...] Object Explorer [...]
[...] online_courses [...]
[...] online_courses_aggregation.sql (79) * - [ ] ...
[...] 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 i.InstructorID = c.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
1 InstructorName AverageRating
1 Mohammed Al-Busaidi 3
2 Sarah Ahmed 3
[...]
[...] Query executed successfully.

```

```

DESKTOP-BLAC28R (15.0 RTM) | DESKTOP-BLAC28R\lathi ... | online_courses | 00:00:00 | 2 rows
Ready 77°C Sunny
[...] Results Messages
1 StudentName TotalCoursesEnrolled
1 Rania Abd 1
2 Ali Salim 3
3 Layla Nasser 2
[...]
[...] Query executed successfully.

```

online\_courses\_aggregation.sql - DESKTOP-BLAC20R.online\_courses (DESKTOP-BLAC20R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

online\_courses

Object Explorer

Connect to: DESKTOP-BLAC20R (SQL Server 15.0.2600.6)

- Databases
- Security
- Server Objects
- Replication
- Polybase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

online\_courses\_aggregation.BLA20R\lathi (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(c.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,
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%.

Results Messages

CategoryName	TotalEnrollments
Business	1
Data Science	2
Web Development	4

Query executed successfully.

Ready Rainy days ahead 17°C

Search

LN 269 Col 1 Ch 1 RS

DESKTOP-BLAC20R (15.0 RTM) DESKTOP-BLAC20R\lathi ... online\_courses 00:00:00 3 rows

8:51 AM 12/24/2023 ENG

The screenshot shows a Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih (79)) - Microsoft SQL Server Management Studio (Administrator)". The left pane is the Object Explorer, showing the database structure. The right pane displays a query results window for a stored procedure named "online\_courses\_aggregation".

The query results show two rows of data:

Fullname	TotalRevenue
Mohammed Al-Busaid	119.97
Sarah Ahmed	139.96

At the bottom of the results window, a message states "Query executed successfully."

System tray icons include: Rainy days ahead, 12%, ENG, 12/24/2023, and 853 AM.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
Connect New Query Execute
Object Explorer
File DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses
onlin... online_courses_aggre... DESKTOP-BLAC28R\lathi (79) + x
--4. Student name + total courses enrolled.
SELECT s.FullName AS StudentName
CROSS JOIN 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 c.CategoryName,
       COUNT(e.CourseID) 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,
       CASE WHEN COUNT(e.EnrollmentID) = 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
Excel for Business	0.0000000000
HTML & CSS Basics	50.0000000000
JavaScript Advanced	0.0000000000
Python for Data Analysis	0.0000000000

Query executed successfully.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
Connect New Query Execute
Object Explorer
File DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses
onlin... online_courses_aggre... DESKTOP-BLAC28R\lathi (79) + x
--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,
       CASE WHEN COUNT(e.EnrollmentID) = 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
JOIN 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.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...] Results [...] Messages
Object Explorer
[...] online_courses
[...] online_courses_aggregation
[...] online_courses_aggregation.BLA2C8Rlathi(79) - X
--> 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,
(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
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
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.

```

Query executed successfully.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
[...] New Query [...]
[...] Execute [...] Results [...] Messages
Object Explorer
[...] online_courses
[...] online_courses_aggregation
[...] online_courses_aggregation.BLA2C8Rlathi(79) - X
--> 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,
(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
JOIN 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.

```

Query executed successfully.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\ath(79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer
Connect Object Explorer
DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses
online_courses_aggre_BLAIC28R\ath(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.

```

Query executed successfully.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\ath(79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer
Connect Object Explorer
DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses
online_courses_aggre_BLAIC28R\ath(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.

```

Query executed successfully.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

```

--1. 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;

--2. Courses with average rating above 4.
SELECT c.Title
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
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.
SELECT s.FullName
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName
HAVING COUNT(e.CourseID) >= 2;

```

Query executed successfully.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

```

--1. 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;

--2. Courses with average rating above 4.
SELECT c.Title
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
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.
SELECT s.FullName
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName
HAVING COUNT(e.CourseID) >= 2;

-->-part 6-----
--1. Most popular course.
SELECT c.Title AS AvgRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.Title
ORDER BY AvgRating DESC;

--2. Instructor to promote.
--3. Highest revenue category.
--4. Do expensive courses have better ratings?
--5. Do cheaper courses have higher completion?

```

Query executed successfully.

**Explanation:** Highest average rating.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

Object Explorer
File Edit View Query Project Tools Window Help
Connect New Query Execute
online_courses
online_courses_aggre... (79) * x

--> 5. Students enrolled in at least 2 courses.
--> SELECT s.FullName
--> FROM Students s
--> JOIN Enrollments e ON s.StudentID = e.StudentID
--> GROUP BY s.FullName
--> HAVING COUNT(e.CourseID) >= 2;

-->--part 6-----
-->--1. Best performing course.
-->SELECT c.Title, AVG(e.Rating) AS AvgRating
-->FROM Courses c
-->JOIN Enrollments e ON c.CourseID = e.CourseID
-->GROUP BY c.Title
-->ORDER BY AvgRating DESC;

-->--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;

-->--3. Highest revenue category.
-->--4. Do expensive courses have better ratings?
-->--5. Do cheaper courses have higher completion?

```

Results

FullName	AvgRating
Mohammed Al-Busaidi	3
Sarah Ahmed	3

Query executed successfully.

**Explanation:** Instructor with best student feedback.

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

Object Explorer
File Edit View Query Project Tools Window Help
Connect New Query Execute
online_courses
online_courses_aggre... (79) * x

--> 5. Students enrolled in at least 2 courses.
--> SELECT s.FullName
--> FROM Students s
--> JOIN Enrollments e ON s.StudentID = e.StudentID
--> GROUP BY s.FullName
--> HAVING COUNT(e.CourseID) >= 2;

-->--part 6-----
-->--1. Best performing course.
-->SELECT c.Title, AVG(e.Rating) AS AvgRating
-->FROM Courses c
-->JOIN Enrollments e ON c.CourseID = e.CourseID
-->GROUP BY c.Title
-->ORDER BY AvgRating DESC;

-->--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;

-->--3. Highest revenue category.
-->SELECT cat.CategoryName, SUM(c.Price) AS Revenue
-->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 Revenue DESC;

-->--4. Do expensive courses have better ratings?
-->--5. Do cheaper courses have higher completion?

```

Results

CategoryName	Revenue
Web Development	393.95
Data Science	59.95
Business	19.95

Query executed successfully.

**Explanation:** Category earning the most money.

online\_courses\_aggregation.sql - DESKTOP-BLAC20R.blac20r.online\_courses (DESKTOP-BLAC20R\blac20r) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

online\_courses

Execute ✓

online\_courses\_aggregation

Object Explorer

Connect to: DESKTOP-BLAC20R (SQL Server 15.0.2600.6) [online\_courses]

Object Explorer Tree:

- Server
- Database
- Security
- Server Objects
- Replication
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPS disabled)
- XEvent Profiler

Toolbox

Query Editor

Results

Messages

online\_courses\_aggregation

GO

```
SELECT e.CourseID, COUNT(e.CourseID) AS EnrollmentCount, AVG(e.Rating) AS AvgRating
FROM Enrollments e
GROUP BY e.CourseID
ORDER BY EnrollmentCount DESC;
```

--1. Who is performing course?

```
SELECT c.CourseName, COUNT(e.CourseID) AS EnrollmentCount, AVG(e.Rating) AS AvgRating
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.CourseName
ORDER BY EnrollmentCount DESC;
```

--2. Instructor to promote.

```
SELECT i.InstructorName, COUNT(e.CourseID) AS EnrollmentCount, 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.InstructorName
ORDER BY EnrollmentCount DESC;
```

--3. What is revenue category.

```
SELECT cat.CategoryName, SUM(c.Price) AS Revenue
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 Revenue DESC;
```

--4. Do expensive courses have better ratings?

```
SELECT CASE WHEN c.Price > 30 THEN 'Expensive' ELSE 'Cheap' END AS PriceGroup,
       COUNT(e.CourseID) AS EnrollmentCount, 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;
```

100 %

Results

PriceGroup	AvgRating
Cheap	4
Expensive	3

Messages

Query executed successfully.

DESKTOP-BLAC20R (15.0 ITM) DESKTOP-BLAC20R\blac20r... online\_courses 00:00:00 2 rows

Ready

20°C Sunny

Search

Ln 159 Col 67 Ch 67 RS

9:47 AM 12/24/2023 ENG

**Explanation:** Compares ratings by price group.

online\_courses\_aggregation.sql - DESKTOP-BLAC2B8.online\_courses (DESKTOP-BLAC2B8\fahti (79)) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

online\_courses

Execute ✓

online\_courses\_aggregation.sql

```
SELECT * FROM online_courses
```

Object Explorer

Connect to: DESKTOP-BLAC2B8 (SQL Server 15.0.2600.6)

- atabases
- Security
- Server Objects
- Replication
- Polybase
- Always On High Availability
- Management
- Integration Services Catalogs
- SQL Server Agent (Agent XPs disabled)
- XEvent Profiler

```
--> 1. Enrollment count per course
SELECT c.CourseID, COUNT(e.CourseID) AS EnrollmentCount
FROM Courses c
JOIN Enrollments e ON c.CourseID = e.CourseID
GROUP BY c.CourseID
ORDER BY EnrollmentCount DESC;
```

```
--> 2. Instructor to promote.
--SELECT l.FullName, AVG(e.Rating) AS AvgRating
--FROM Instructors l
--JOIN Courses c ON l.InstructorID = c.InstructorID
--JOIN Enrollments e ON c.CourseID = e.CourseID
--GROUP BY l.FullName
--ORDER BY AvgRating DESC;
```

```
--> 3. Highest revenue category.
--SELECT cat.CategoryName, SUM(c.Price) AS Revenue
--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 Revenue DESC;
```

```
--> 4. Do expensive courses have better ratings?
--SELECT
--CASE WHEN c.Price > 30 THEN 'Expensive' ELSE 'Cheap' END AS PriceGroup,
--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;
```

```
--> 5. Do cheaper courses have higher completion?
--SELECT
--CASE WHEN c.Price < 30 THEN 'Cheap' ELSE 'Expensive' END AS PriceGroup,
--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;
```

100 %

Results Messages

PriceGroup	AvgCompletion
Cheap	86
Expensive	55

Query executed successfully.

**Explanation:** Compares completion based on price.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih) - 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.
```

Results

Title	Revenue
Python for Data Analysis	59.98
JavaScript Advanced	79.98
HTML & CSS Basics	59.98

Query executed successfully.

online\_courses\_aggregation.sql - DESKTOP-BLAC28R.online\_courses (DESKTOP-BLAC28R\fatih) - 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.
```

Results

FullName	TotalEnrollments
Sarah Ahmed	4

Query executed successfully.

```
online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
Connect Object Explorer
DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses
online_courses_aggre_BLA28R(lathi (79))
--Final Challenge-----
--1. Top 3 courses by revenue.
SELECT TOP 3 c.Title
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
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.
```

```
online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
Connect Object Explorer
DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases Security Server Objects Replication Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
online_courses
online_courses_aggre_BLA28R(lathi (79))
--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.
```

```

online_courses_aggregation.sql - DESKTOP-BLAC28R.online_courses (DESKTOP-BLAC28R\lathi (79)) - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help
New Query New Results New Object Explorer New Object Explorer Details New Task List New Task List Details
Execute
Object Explorer
Connect - Help
DESKTOP-BLAC28R (SQL Server 15.0.2000.5)
Databases
Always On
Server Objects
Replication
PolyBase
Always On High Availability
Management
Connection Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler

online_courses_aggregation.sql (79) + X
online_courses_aggregation.sql (79) + X
--1. 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;

--2. 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;

--3. 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;

--4. Student enrolled in most courses.
SELECT TOP 1 s.FullName, COUNT(e.StudentID) AS TotalCourses
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
GROUP BY s.FullName
ORDER BY TotalCourses DESC;

```

Results

FullName	TotalCourses
Ali Salim	3

Query executed successfully.

### 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**.