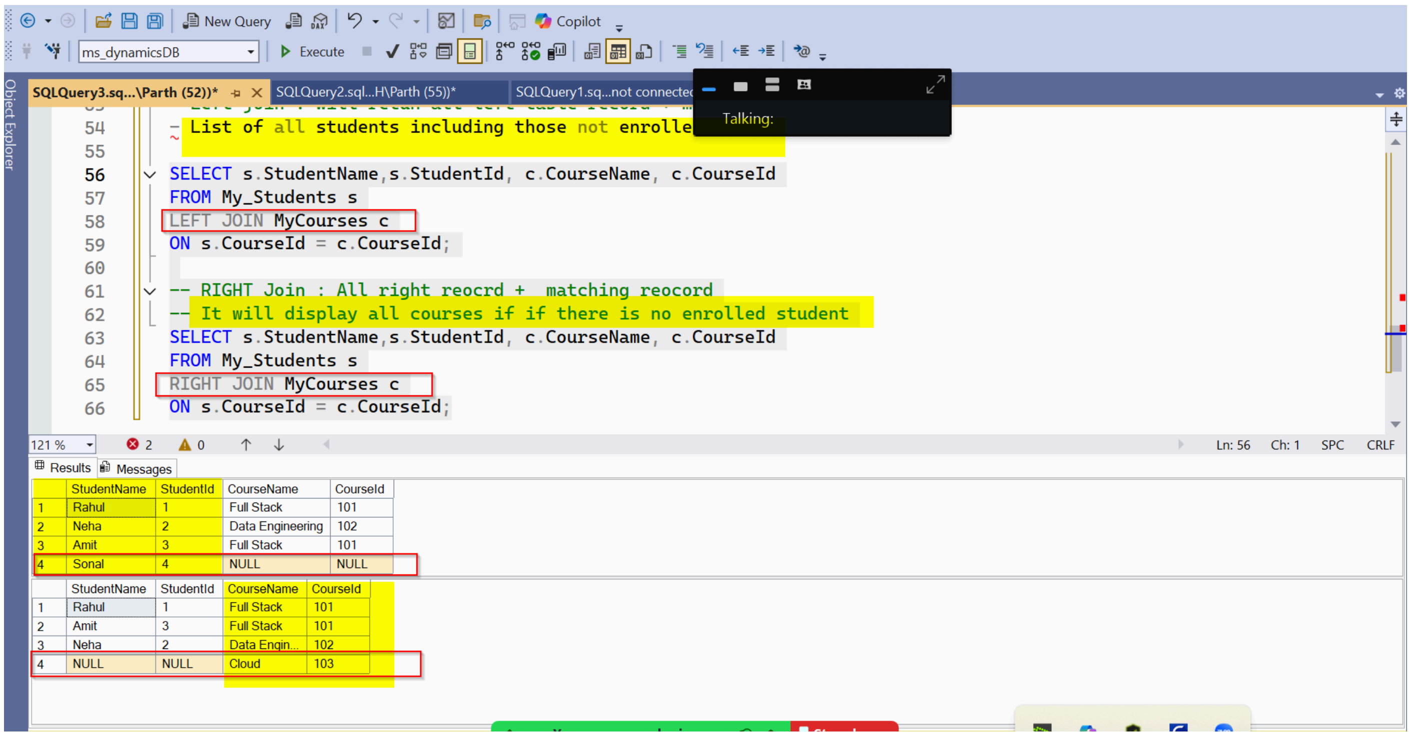
Joins
Inner join
Left join
Right join
Self-join
Full outer join
Cross join
Set operators
Union
Intersect
Minus



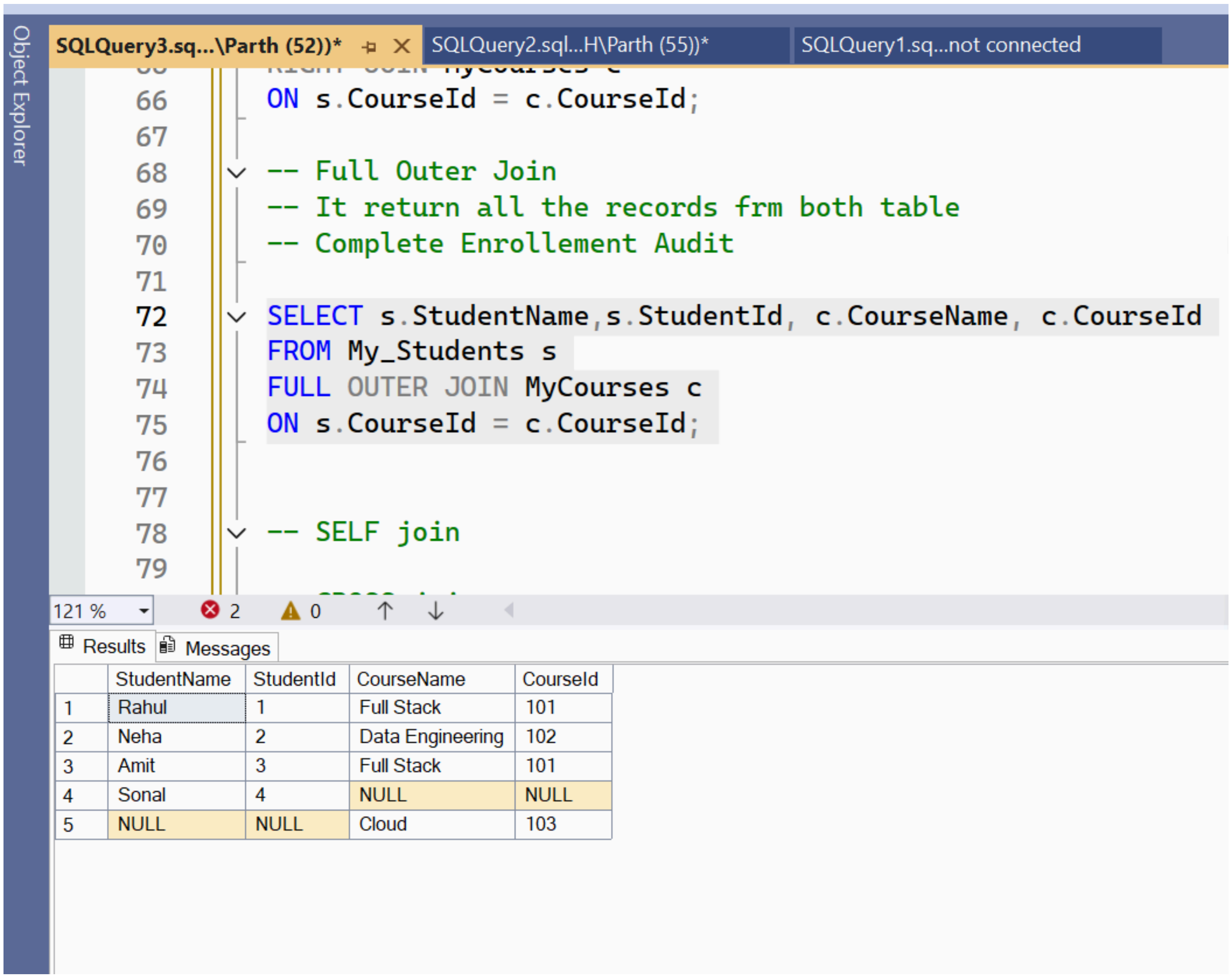
```sql
-- List of all students including those not enrolle

SELECT s.StudentName,s.StudentId, c.CourseName, c.CourseId
FROM My_Students s
LEFT JOIN MyCourses c
ON s.CourseId = c.CourseId;

-- RIGHT Join : All right reocrd +  matching reocord
-- It will display all courses if if there is no enrolled student
SELECT s.StudentName,s.StudentId, c.CourseName, c.CourseId
FROM My_Students s
RIGHT JOIN MyCourses c
ON s.CourseId = c.CourseId;
```

```
   StudentName  StudentId  CourseName        CourseId
1  Rahul        1          Full Stack        101
2  Neha         2          Data Engineering  102
3  Amit         3          Full Stack        101
4  Sonal        4          NULL              NULL
```

```
   StudentName  StudentId  CourseName     CourseId
1  Rahul        1          Full Stack     101
2  Amit         3          Full Stack     101
3  Neha         2          Data Engin...  102
4  NULL         NULL       Cloud          103
```


Object Explorer
SQLQuery3.sq...\Parth (52))*
SQLQuery2.sql...H\Parth (55))*
SQLQuery1.sq...not connected
121 %
2
0
Results
Messages

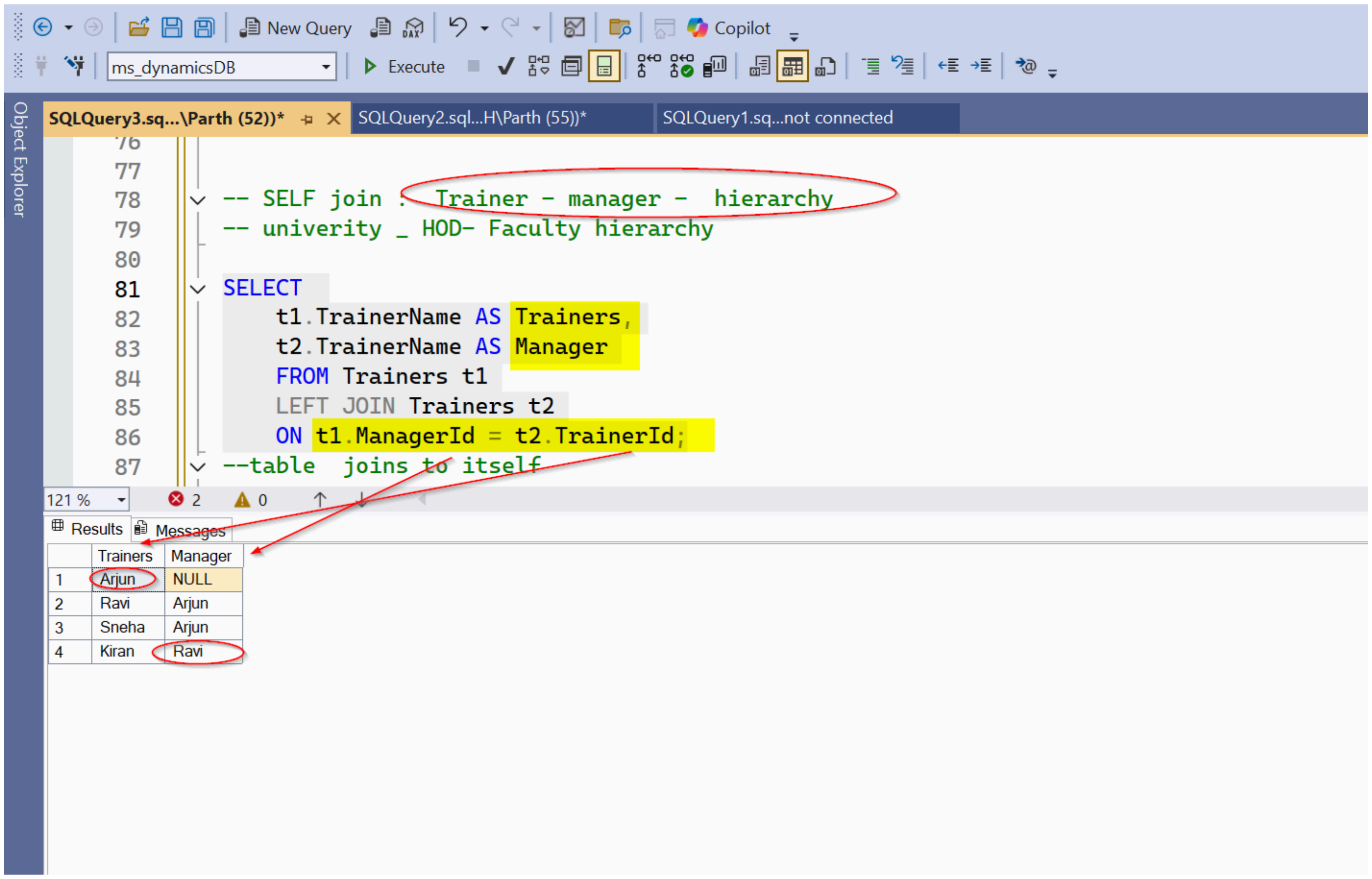
```sql
ON s.CourseId = c.CourseId;

-- Full Outer Join
-- It return all the records frm both table
-- Complete Enrollement Audit

SELECT s.StudentName,s.StudentId, c.CourseName, c.CourseId
FROM My_Students s
FULL OUTER JOIN MyCourses c
ON s.CourseId = c.CourseId;


-- SELF join
```

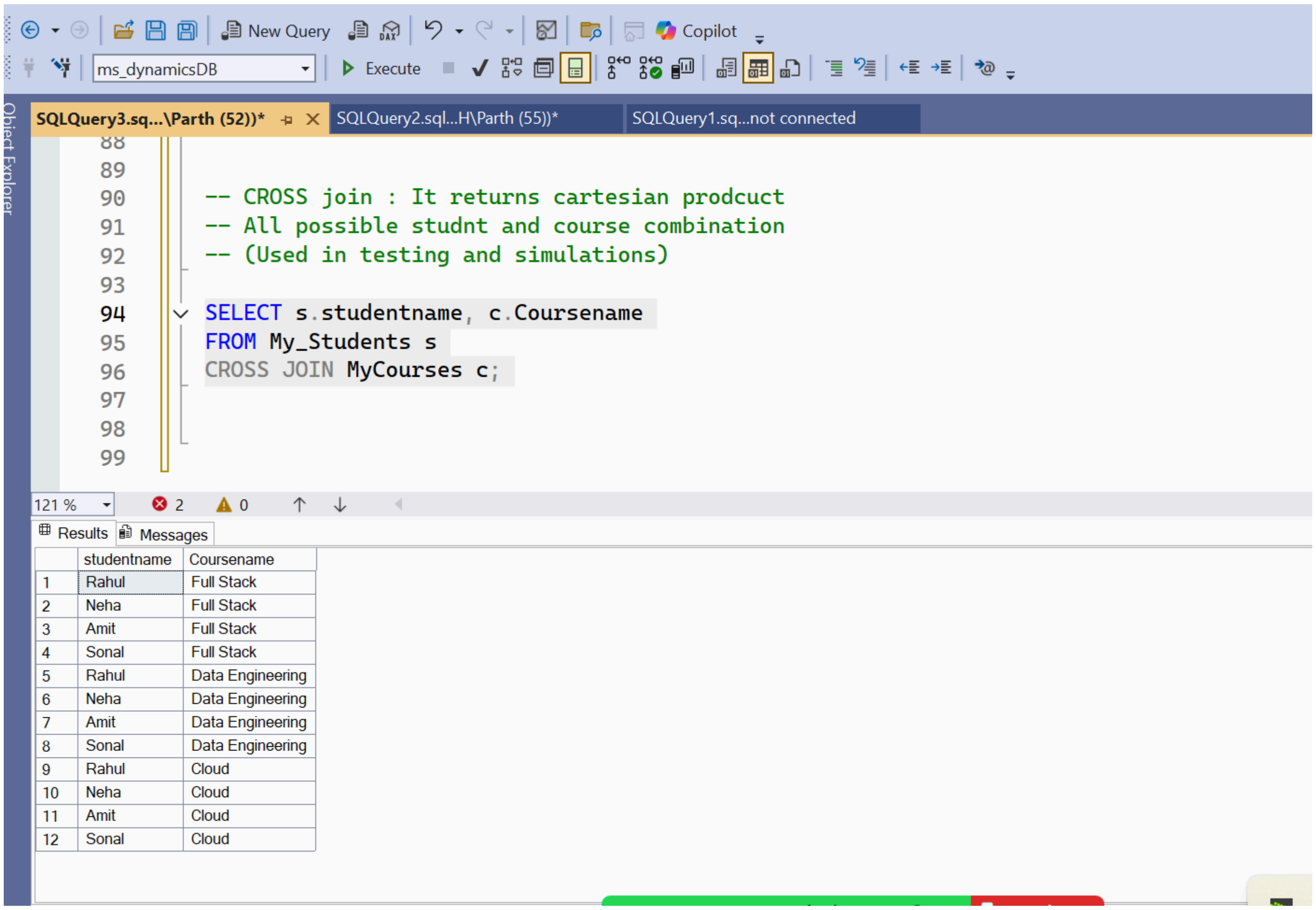
	StudentName	StudentId	CourseName	CourseId
1	Rahul	1	Full Stack	101
2	Neha	2	Data Engineering	102
3	Amit	3	Full Stack	101
4	Sonal	4	NULL	NULL
5	NULL	NULL	Cloud	103



```
-- SELF join : Trainer - manager -  hierarchy
-- univerity _ HOD- Faculty hierarchy

SELECT
    t1.TrainerName AS Trainers,
    t2.TrainerName AS Manager
    FROM Trainers t1
    LEFT JOIN Trainers t2
    ON t1.ManagerId = t2.TrainerId;
--table  joins to itself
```

	Trainers	Manager
1	Arjun	NULL
2	Ravi	Arjun
3	Sneha	Arjun
4	Kiran	Ravi



New Query
Execute
Copilot
ms_dynamicsDB
SQLQuery3.sq...\Parth (52))*
SQLQuery2.sql...H\Parth (55))*
SQLQuery1.sq...not connected
Object Explorer
121 %
Results
Messages

```sql
-- CROSS join : It returns cartesian prodcuct
-- All possible studnt and course combination
-- (Used in testing and simulations)

SELECT s.studentname, c.Coursename
FROM My_Students s
CROSS JOIN MyCourses c;
```

	studentname	Coursename
1	Rahul	Full Stack
2	Neha	Full Stack
3	Amit	Full Stack
4	Sonal	Full Stack
5	Rahul	Data Engineering
6	Neha	Data Engineering
7	Amit	Data Engineering
8	Sonal	Data Engineering
9	Rahul	Cloud
10	Neha	Cloud
11	Amit	Cloud
12	Sonal	Cloud

Case Study: Learning Management & Enrollment System (LMES)

Business Context

A training organization wants to build a **Learning Management & Enrollment System** to manage:

- Courses
- Students

- Trainers
- Enrollments
- Reporting & access control

The system must ensure **data integrity**, **secure access**, and **accurate reporting**.

120-Minute Session Breakdown

Time (mins)	Module
0–15	Database basics & SSMS setup
15–35	Table identification & relationships
35–60	DDL, constraints, CRUD
60–85	Queries, joins, functions
85–105	Stored procedures & transactions
105–120	Security (DCL), review & Q&A

---

1. Database Identification (User Story Driven)

User Story 1

As a system administrator, I want to store master data for courses so that enrollments can reference valid courses.

Identified Tables

Table	Type
Courses	Master
Students	Master

Trainers	Master
Enrollments	Transaction

Relationship summary

Parent Table	Child Table	Relationship
Students	**Enrollments**	**One-to-Many**
Courses	**Enrollments**	**One-to-Many**
Trainers	**Trainers**	**Self-Join**

Requirement	Achieved
Data Integrity	**Constraints & FK**
Security	**GRANT / REVOKE**
Reporting	**Joins & functions**
Reliability	**Transactions**
Scalability	**Normalized design**

Best Practices for Keys and constraints:

Practice	Recommendation
Primary Key	**Always use, preferably surrogate (INT/IDENTITY)**
Foreign Key	**Enforce referential integrity**
NOT NULL	**Use wherever possible**
UNIQUE	**Enforce business rules**
CHECK	**Validate data at database level**
Index FK columns	**Improves JOIN performance**

Data Types & Storage

Practice	Reason
Use appropriate data types	Saves space, improves speed
Avoid VARCHAR(MAX) unless required	Causes performance issues
Use DATE / DATETIME2	Better precision and storage
Avoid implicit conversions	Prevents query slowdown

DML & Query Writing

Practice	Benefit
Avoid SELECT *	Fetch only required columns
Use WHERE clause always	Prevent full table scans
Use parameterized queries	Prevent SQL injection(Website attack)
Batch inserts/updates	Better performance
Avoid cursors	Use set-based operations

Indexing Best Practices

Practice	Why
Index columns used in JOIN, WHERE	Faster reads
Avoid over-indexing	Slows inserts/updates
Use clustered index wisely	Affects physical data order
Monitor index fragmentation	Maintain performance

Transactions & Concurrency

Practice	Impact
Keep transactions short	Reduces locking
Use explicit transactions	Ensures data consistency
Handle deadlocks gracefully	Improves reliability
Choose correct isolation level	Balances consistency & performance

Stored Procedures & Functions

Practice	Recommendation
Use stored procedures for DML	Better control & security
Avoid business logic in DB	Keep logic minimal
Prefer inline table-valued functions	Best performance
Do not modify data in functions	SQL Server rule(Imp)

1. SQL JOINS – Interview Questions

Basic

1. **What is the difference between INNER JOIN and LEFT JOIN?**
2. **When will RIGHT JOIN return NULL values?**
3. **What is a SELF JOIN and where is it used?**
4. **What is a CROSS JOIN and why is it rarely used?**
5. **What is the difference between JOIN and SUBQUERY?**

Intermediate

6. How does FULL OUTER JOIN behave when there is no matching data?
7. Can INNER JOIN return duplicate rows? Why?
8. What happens if JOIN condition is missing?
9. How do JOINs impact query performance?
10. Difference between JOIN and UNION?

Advanced / Scenario

11. How would you find records present in one table but not in another?
12. Explain a real-world use case for SELF JOIN.
13. How do you optimize JOIN-heavy queries?
14. How do foreign keys affect JOIN performance?
15. When would you prefer EXISTS over JOIN?

---

2. STORED PROCEDURES – Interview Questions

Basic

16. What is a stored procedure?
17. Why are stored procedures preferred over inline SQL?
18. How do you pass parameters to a stored procedure?
19. What is the difference between input and output parameters?

20. How do you execute a stored procedure?

Intermediate

21. Can stored procedures return multiple result sets?
22. What is the use of TRY–CATCH in stored procedures?
23. How do transactions work inside stored procedures?
24. Difference between stored procedure and function?
25. What is parameter sniffing?

Advanced / Scenario

26. How do you handle errors inside stored procedures?
27. How do you prevent SQL injection in stored procedures?
28. When should you avoid using stored procedures?
29. How do you log errors from a stored procedure?
30. Can a stored procedure call a function? Explain.

---

3. FUNCTIONS – Interview Questions

Basic

31. What are SQL Server functions?
32. What is the difference between scalar and table-valued functions?
33. Can functions modify data in SQL Server?

34. **How do you call a function?**

35. **Name some built-in SQL Server functions.**

Intermediate

36. **Difference between inline and multi-statement table-valued functions?**

37. **Why are inline TVFs faster?**

38. **Can functions contain TRY–CATCH blocks?**

39. **Can a function call a stored procedure?**

40. **What are deterministic and non-deterministic functions?**

Advanced / Scenario

41. **When should you use a function instead of a stored procedure?**

42. **What are performance drawbacks of scalar functions?**

43. **Can functions be used in JOIN conditions?**

44. **Why are scalar UDFs discouraged in SELECT?**

45. **How do you refactor a slow function?**

---

4. DATA TYPES – Interview Questions (SQL Server)

Basic

46. **Difference between CHAR and VARCHAR?**

47. **Difference between VARCHAR and NVARCHAR?**

48. **What is the difference between INT and BIGINT?**

49. **When should you use DATETIME2 instead of DATETIME?**

50. **What is NULL and how is it different from empty string?**

Intermediate

51. **What happens when data types are mismatched in JOINs?**

52. **What is implicit vs explicit conversion?**

53. **Difference between FLOAT and DECIMAL?**

54. **What data type should be used for money?**

55. **What is the impact of using VARCHAR(MAX)?**

Advanced / Scenario

56. **How do wrong data types affect performance?**

57. **How do you store encrypted data in SQL Server?**

58. **How do you handle time zones in SQL Server?**

59. **What is collation and why is it important?**

60. **How do indexes behave with different data types?**

---

5. Rapid-Fire Scenario Questions

Scenario	Expected Answer
Find unmatched records	LEFT JOIN / EXCEPT
Count related rows	GROUP BY + JOIN
Secure data modification	Stored Procedure
Reusable query logic	Inline TVF
Large text storage	VARCHAR(MAX) (with caution)
Avoid duplicate records	UNIQUE constraint

6. Interview Tip (What Interviewers Look For)

Area	What They Expect
Joins	Correct type + performance awareness
Procedures	Error handling & transactions
Functions	Knowing limitations
Data Types	Storage + performance reasoning