**Assignment 4**

**Name** : Jeevan Sai Badana

**Mail** : jeevansai100@gmail.com

**Date** : 22-01-2024

**Fundamentals of SQL Manipulating Data by Using DML Statements**

1.Storing data in a table

We can store data by using insert statement



2.Update the data in a table

UPDATE animals SET color = 'Brown' WHERE id = 'a1e7a7fc-b429-41ec-9924-8bb39dd397c8';

* Here it updates the colour to brown for the respective id

3.Deleting the data

DELETE FROM animals WHERE id = '89354034-20d9-4c3d-8195-3294bfd9dbc5';

* Here it delete respective id from animals table

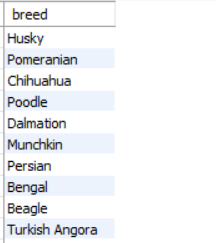
4.Retrieving data from table

SELECT \* FROM animals;



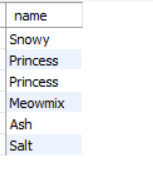
5.Retrieving selected row only

SELECT breed FROM animals;



6.Filetring data using Where clause

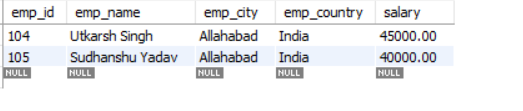
SELECT name FROM animals WHERE gender = 'Female';



7. Filtering Data:IN,DISTINCT,AND,OR,IN,BETWEEN,LIKE,Column & table aliases

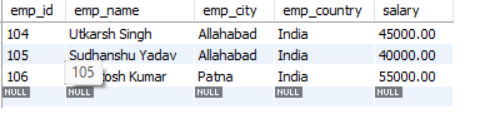
* AND operator

SELECT \* FROM employee WHERE emp\_city = 'Allahabad' AND emp\_country = 'India';



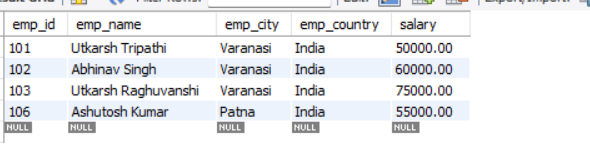
* IN operator

SELECT \* FROM employee WHERE emp\_city IN ('Allahabad', 'Patna');



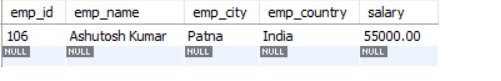
* NOT LIKE operator

SELECT \* FROM employee WHERE emp\_city NOT LIKE 'A%';



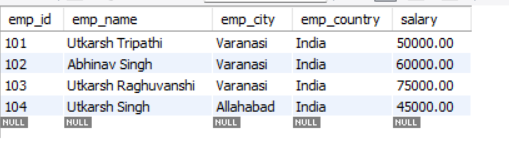
* LIKE operator

SELECT \* FROM employee WHERE emp\_city LIKE 'P%';



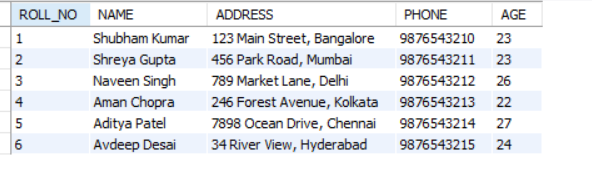
* BETWEEN operator

SELECT \* FROM employee WHERE emp\_id BETWEEN 101 AND 104;



* Distinct Operator

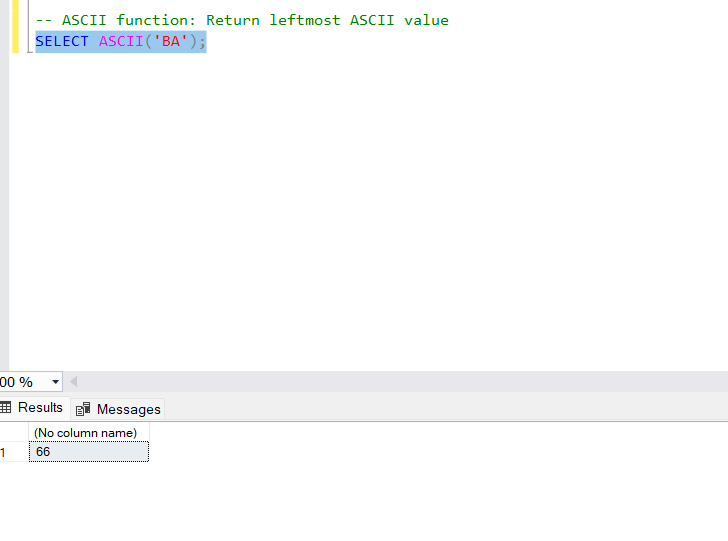
SELECT DISTINCT \* FROM students;



**Manipulating Data by using DML statements**

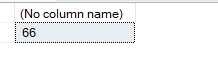
**STRING FUNCTIONS:**

1.ASCII function: Return leftmost ASCII value



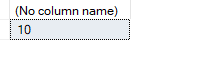
2.CHAR function: Return ASCII value to character

SELECT CHAR(67);



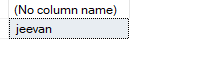
3. LEN function: Return length

SELECT LEN('Jeevan Sai');



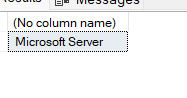
4. LOWER function: Convert to lowercase

SELECT LOWER('JEEVAN');



5.REPLACE function: Replace

SELECT REPLACE('Microsoft SQL', 'SQL', 'Server');



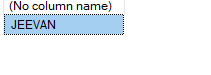
6. REVERSE function: Reverse the string

SELECT REVERSE('python');



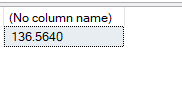
7. UPPER function: Converts to upper

SELECT UPPER('jeevan');



8.STR function: Convert number to string

SELECT STR(136.564, 8, 4);



**DATE FUNCTIONS:**

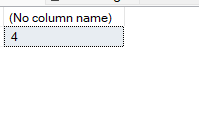
1.GETDATE function: Get current date and time

SELECT GETDATE();



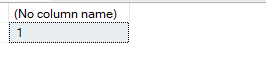
2.DATEDIFF function: Return the difference of date in years

SELECT DATEDIFF(YEAR, CONVERT(DATETIME, '2006-03-06'), CONVERT(DATETIME, '2010-09-01'));



3. -- DATEPART function: Return months value

SELECT DATEPART(MM, '2024-01-22');



4.DAY function: Return day value

SELECT DAY('2024-01-22');



5. MONTH function: Return month value

SELECT MONTH('2024-01-22');

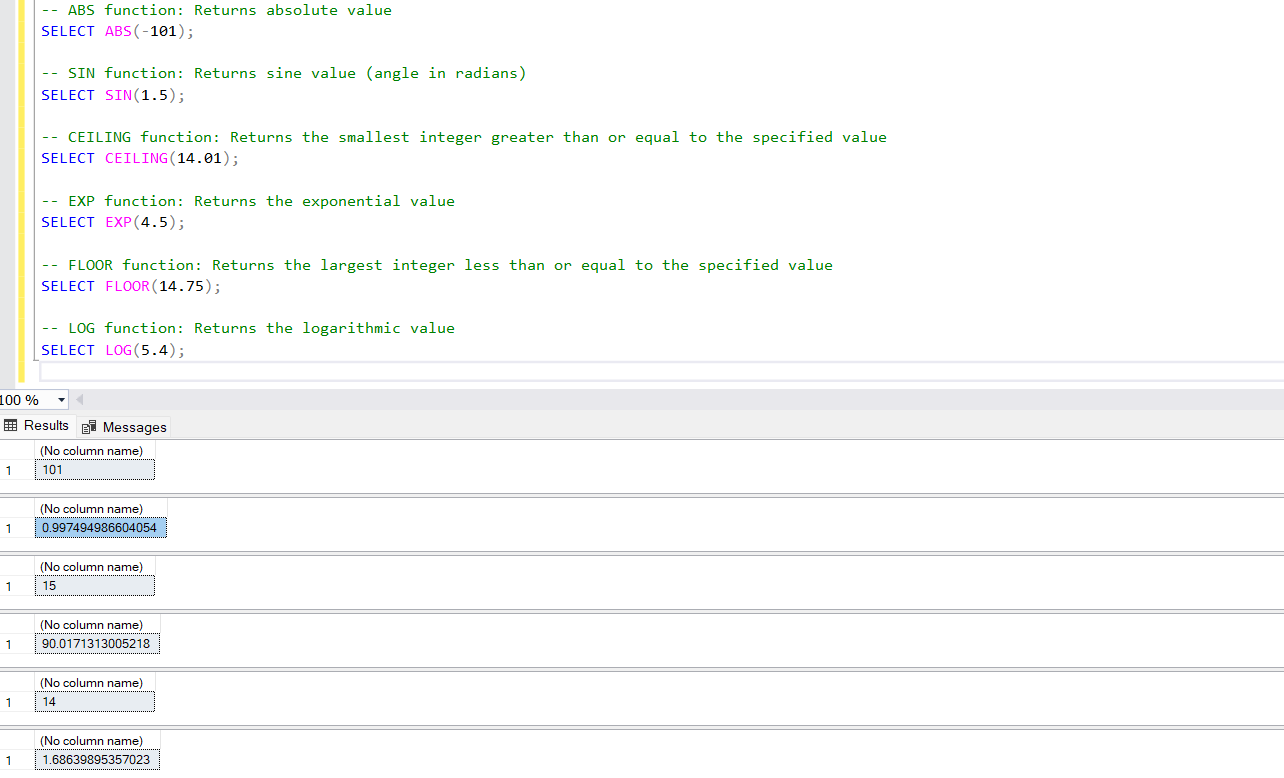


6.YEAR function: Return year value

SELECT YEAR('2024-01-22');

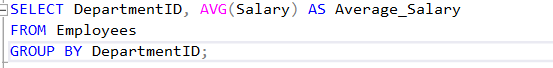


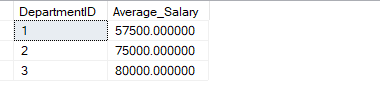
**MATHMETICAL FUNCTIONS:**

****

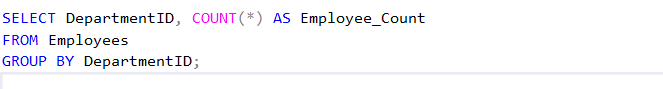
**HANDS ON AGGREGATE FUNCTIONS:**

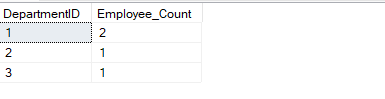
1.Avg()



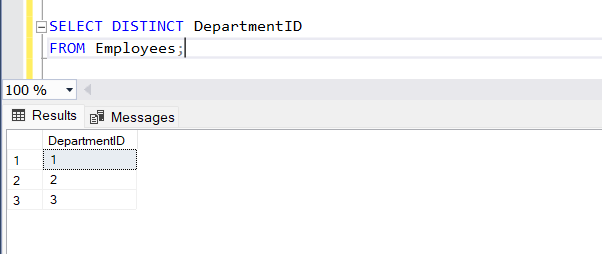


2.COUNT()

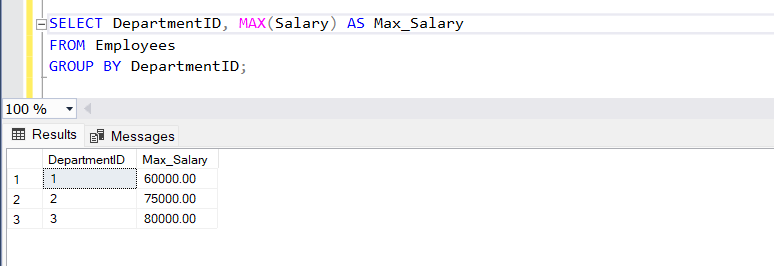




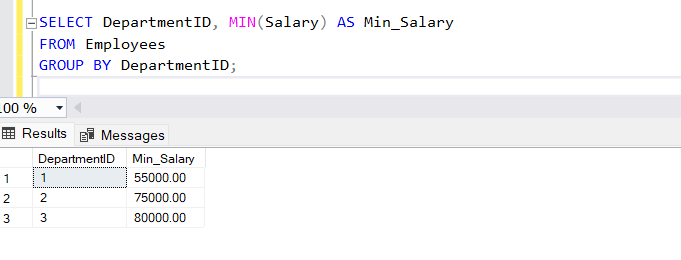
3.DISTINCT()



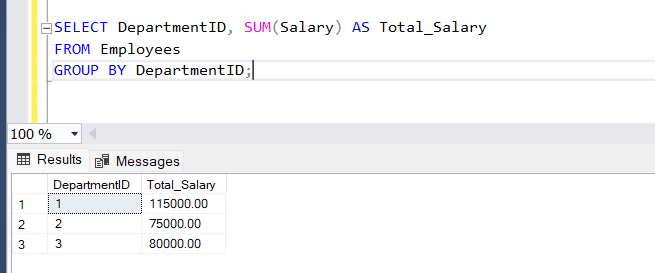
4.MAX()



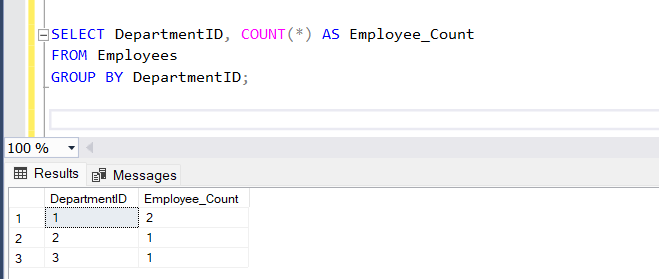
5.MIN()

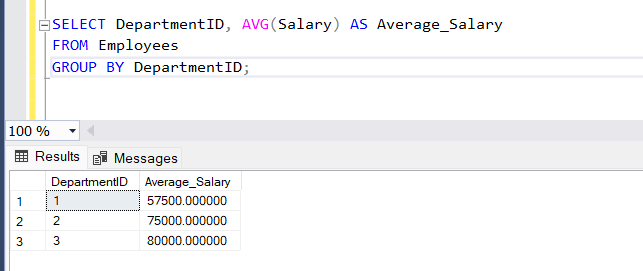


6.SUM()

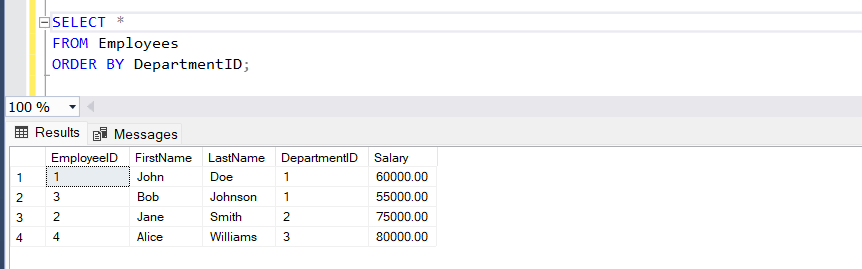


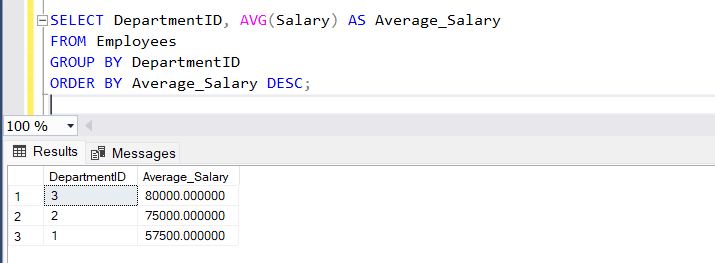
**GROUP BY:**

****

****

**ORDER BY:**

****

****

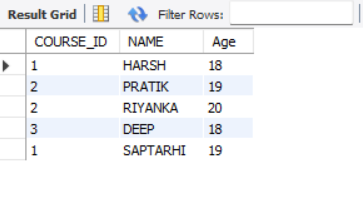
**SQL JOINS:**

INNER JOIN

* It joins both table until condition satisfies
* The result of an INNER JOIN is a new table that contains only the rows where there is a match in the specified columns from both tables. If there is no match, the row is excluded from the result set.

select studentcourse.COURSE\_ID,student.NAME,student.Age

from studentcourse join student on studentcourse.ROLL\_NO=student.ROLL\_NO;

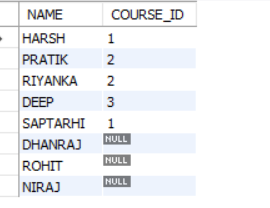


LEFT JOIN

* This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null.

select student.NAME,studentcourse.COURSE\_ID from student

left join studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO

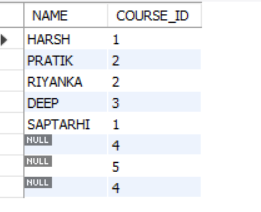


RIGHT JOIN

* RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null.

select student.NAME,studentcourse.COURSE\_ID from student right join

studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO;



FULL JOIN

* FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values

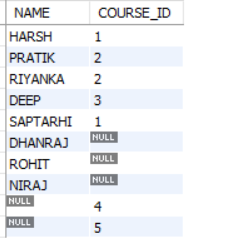
select student.NAME,studentcourse.COURSE\_ID from student

left join studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO

union

select student.NAME,studentcourse.COURSE\_ID from student right join

studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO;



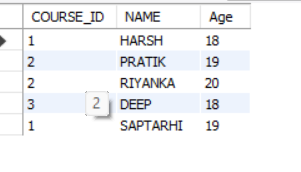
NATURAL JOIN

* In a NATURAL JOIN, the join is based on all columns with the same name in both tables. You don't need to explicitly specify the columns or conditions for the join.

SELECT studentcourse.COURSE\_ID, student.NAME, student.Age

FROM studentcourse

NATURAL JOIN student;



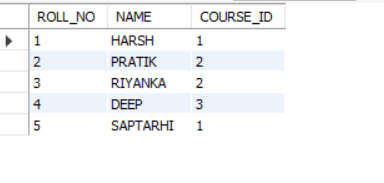
EQUI JOIN

* In an EQUI join, you typically join tables based on the equality of columns. In this example, I'll join the Student and StudentCourse tables based on the common column ROLL\_NO.

SELECT Student.ROLL\_NO, Student.NAME, StudentCourse.COURSE\_ID

FROM Student

JOIN StudentCourse ON Student.ROLL\_NO = StudentCourse.ROLL\_NO;



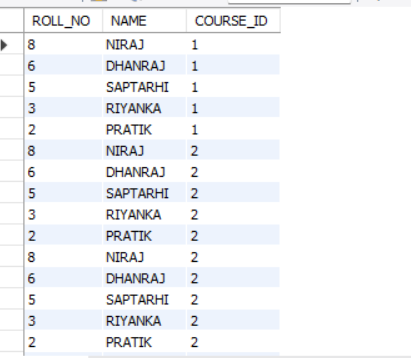
NON-EQUII JOIN

* In a NON-EQUI join, you join tables based on conditions other than equality. In this example, I'll use a condition where the Age of the student is greater than 18.

SELECT Student.ROLL\_NO, Student.NAME, StudentCourse.COURSE\_ID

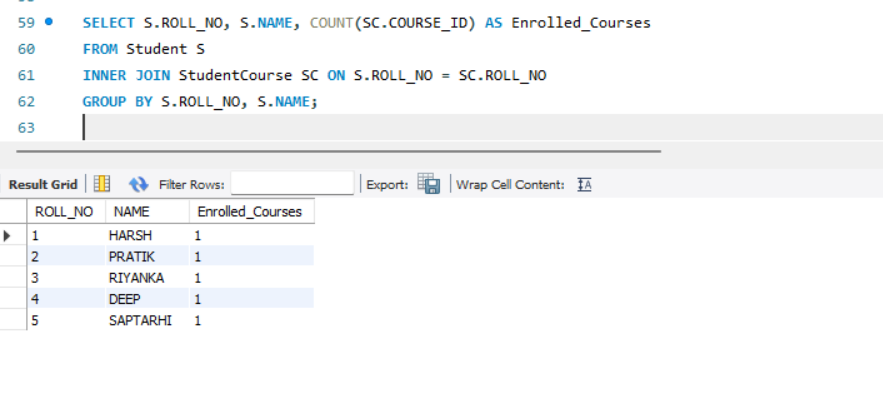
FROM Student

JOIN StudentCourse ON Student.Age > 18;

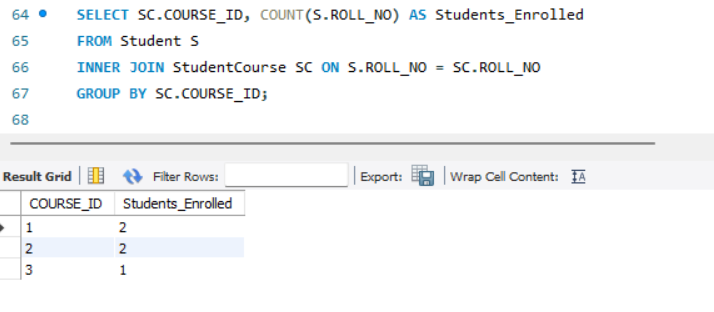


**JOINS USING AGGREGATE FUNCTIONS:**

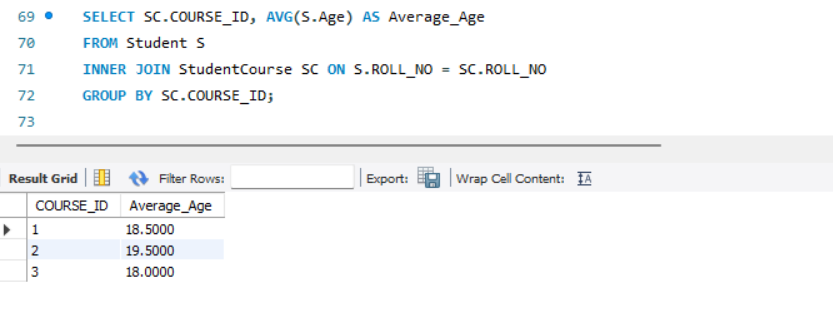
--get the Count of Courses Each Student is Enrolled In

****

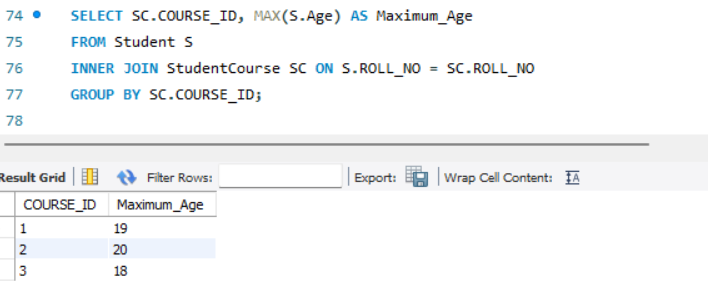
**--**get total number of students enrolled in each course

****

--get the Average Age of Students Enrolled in Each Course**:**

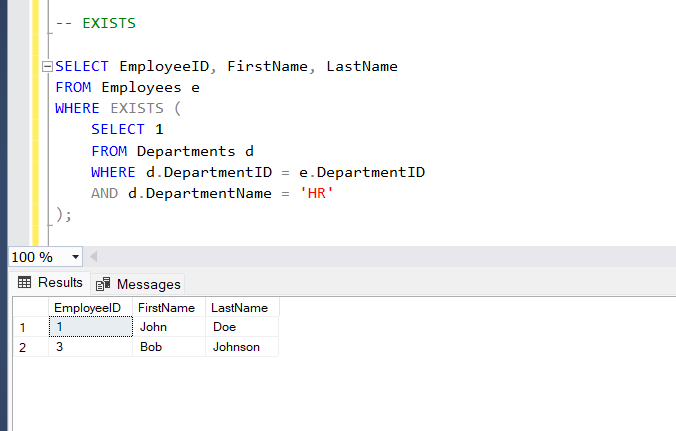
****

-- get the Maximum Age of Students Enrolled in Each Course

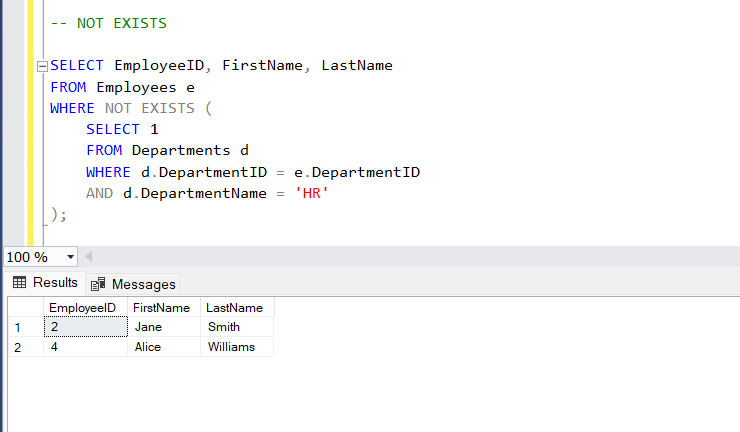


**QUERYING DATA USING JOINS AND SUBQUERIES:**

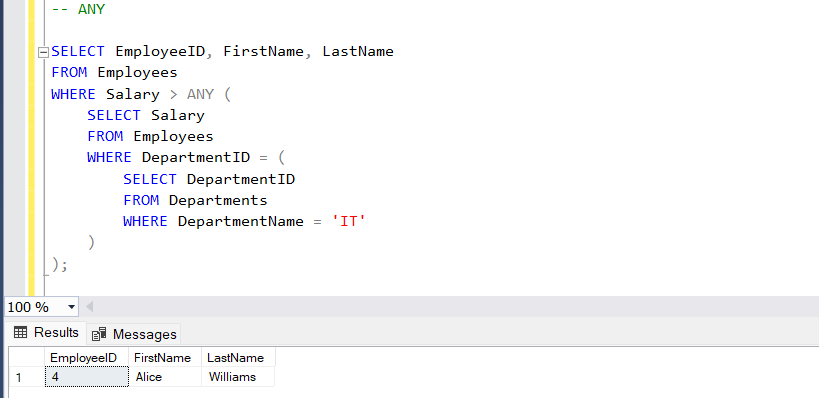
**EXISTS:**

****

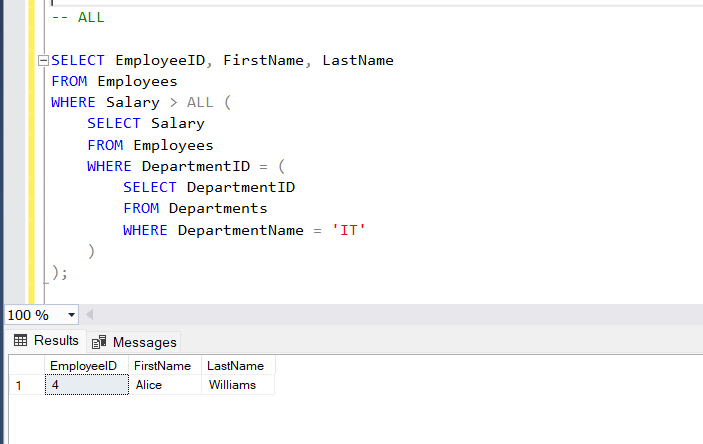
**NOT EXISTS:**

****

**ANY :**

****

**ALL :**

****