# **SQL Assignment #2 for Cohort #2**

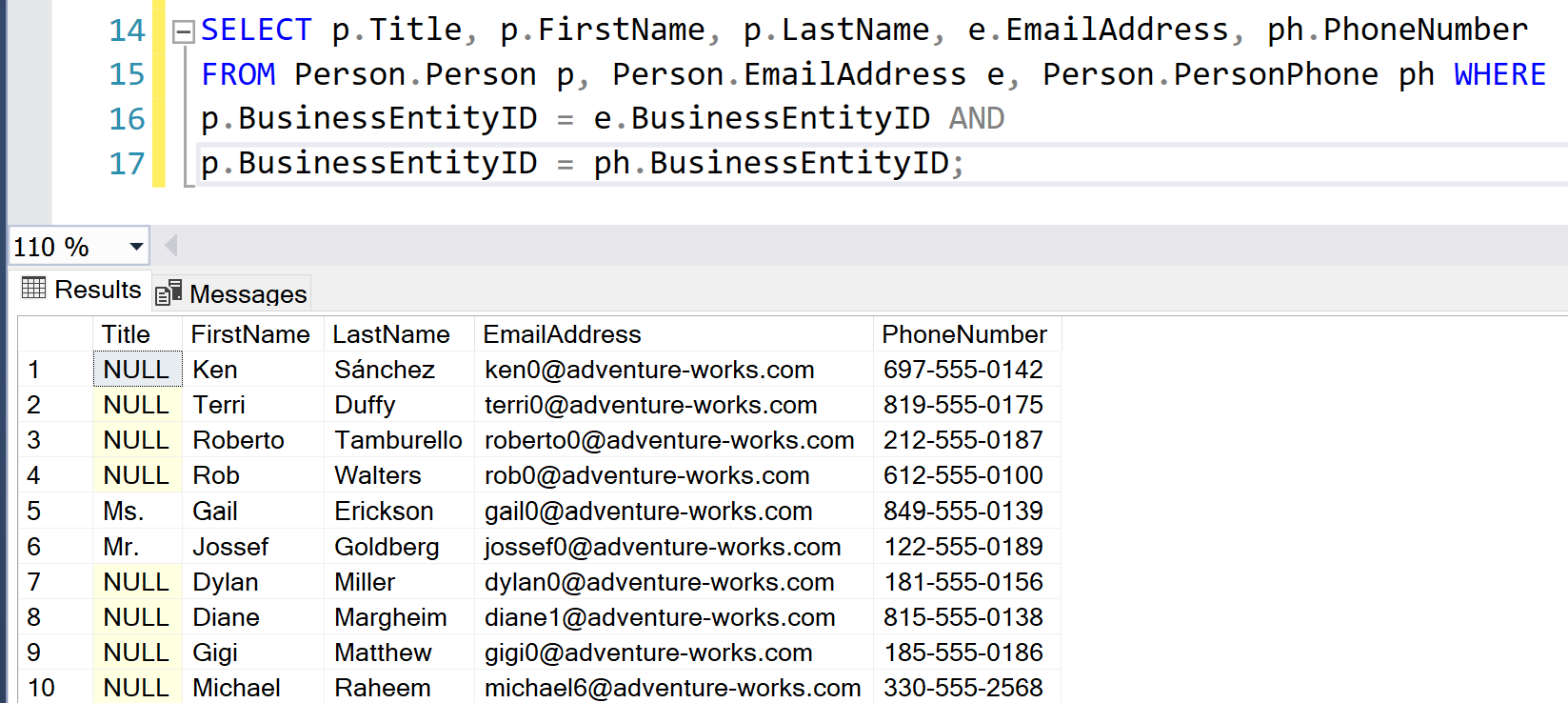
1. We need to send emails to our customers about a campaign and need Contact Details like PhoneNumber, EmailID and their names:

SELECT p.Title, p.FirstName, p.LastName, e.EmailAddress, ph.PhoneNumber

FROM Person.Person p, Person.EmailAddress e, Person.PersonPhone ph WHERE

p.BusinessEntityID = e.BusinessEntityID AND

p.BusinessEntityID = ph.BusinessEntityID;



Here we manually matched the BusinessEntityID column from Person, PersonPhone and EmailAddress tables to get 19,972 Rows.

1. Using PersonPhone table classify Phone-Number-Type as Cell, Landline or Fax:

SELECT \*,

PhoneType =

CASE PhoneNumberTypeID

WHEN '1' THEN 'Cellphone'

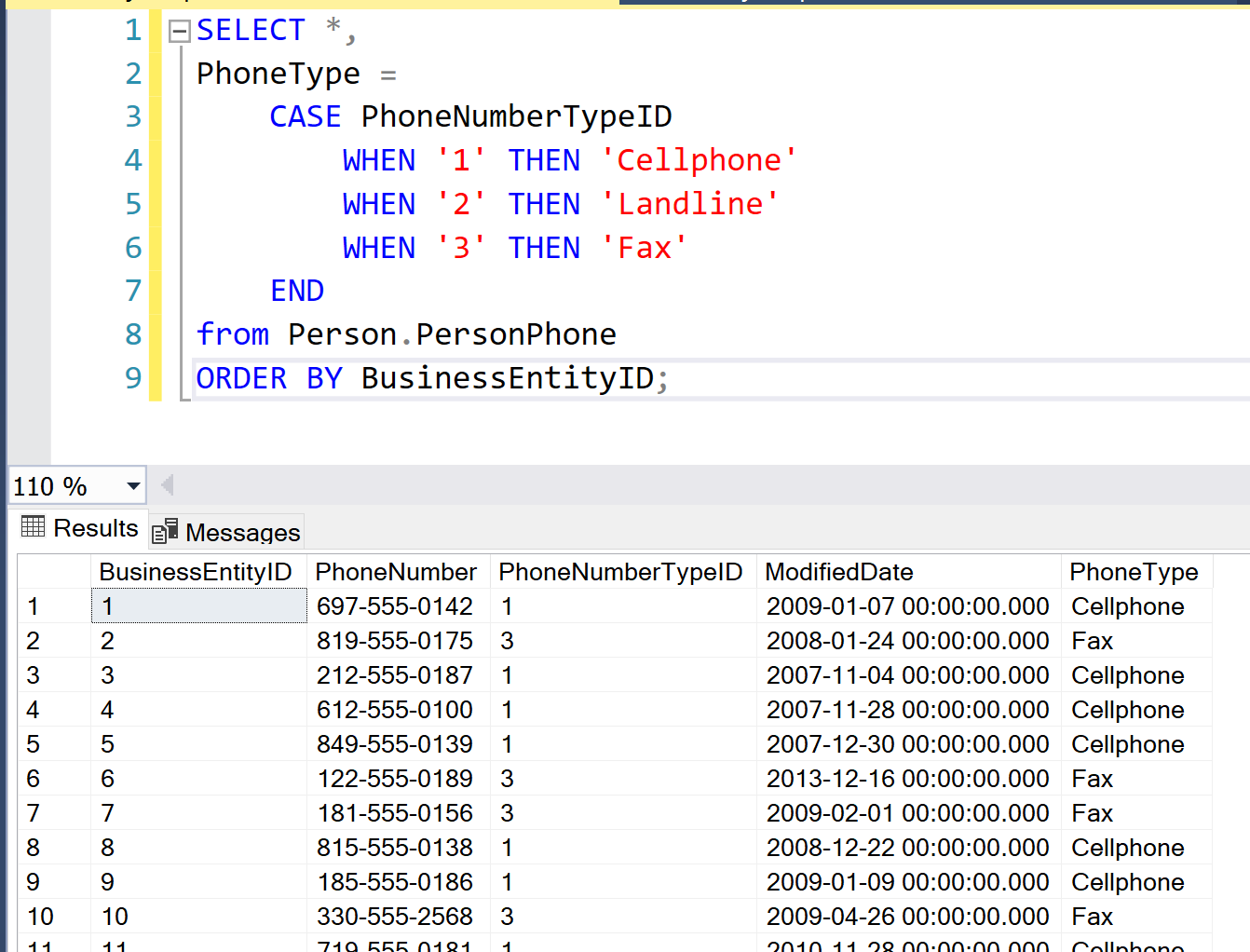
WHEN '2' THEN 'Landline'

WHEN '3' THEN 'Fax'

END

from Person.PersonPhone

ORDER BY BusinessEntityID;



Here we selected all columns and then Classified PhoneNumberTypeID as:

|  |  |
| --- | --- |
| **PhoneNumberTypeID** | **Value** |
| 1 | CellPhone |
| 2 | Landline |
| 3 | Fax |

1. What are Joins and explain the difference between them:

Joins are used to retrieve data from two or more related tables who are related to each other using a common column like a foreign key constraint or other matching column:

Syntax for Joins except Cross Joins:

SELECT columnList

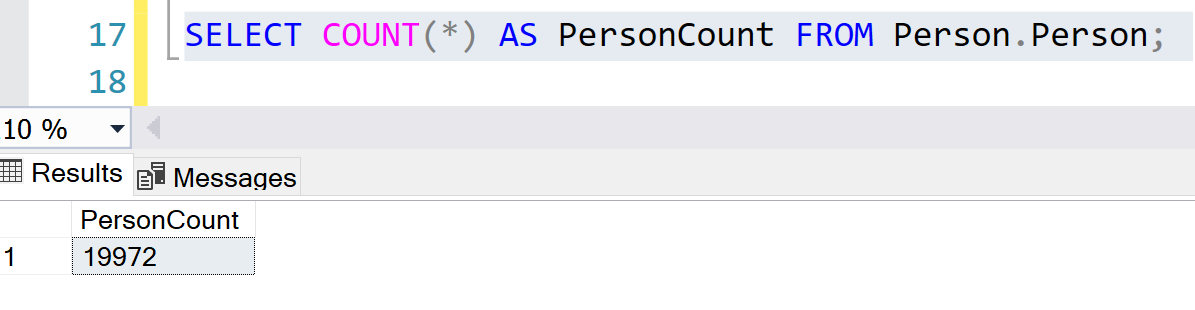
FROM table1

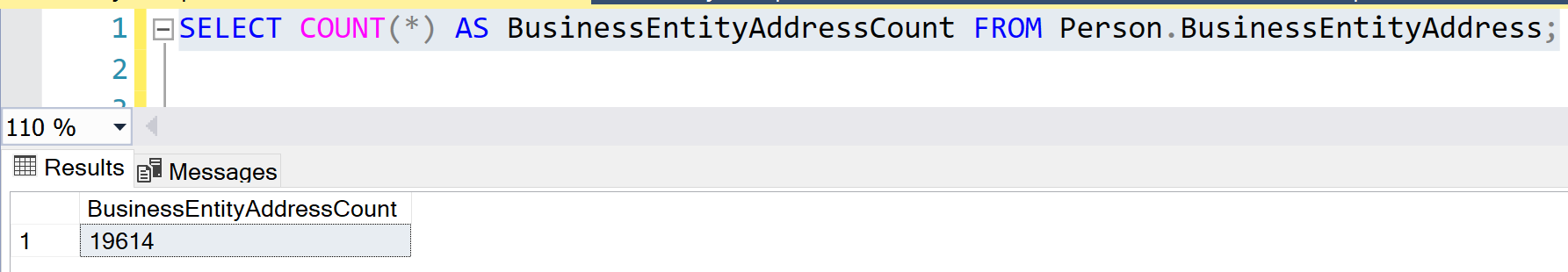
JOINTYPE table2

ON

JoinCondition

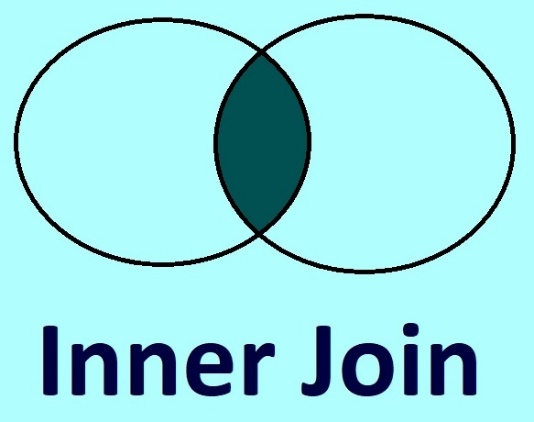
Total Count of Rows in Both our tables, Person and BusinessEntityAddress:





Inner Join or Join – This is similar to Intersection of two sets

Here, we get just the matching rows from both tables and eliminate others:



**Only Matching Rows Returned**

SELECT count(\*) AS innerJoinCnt FROM

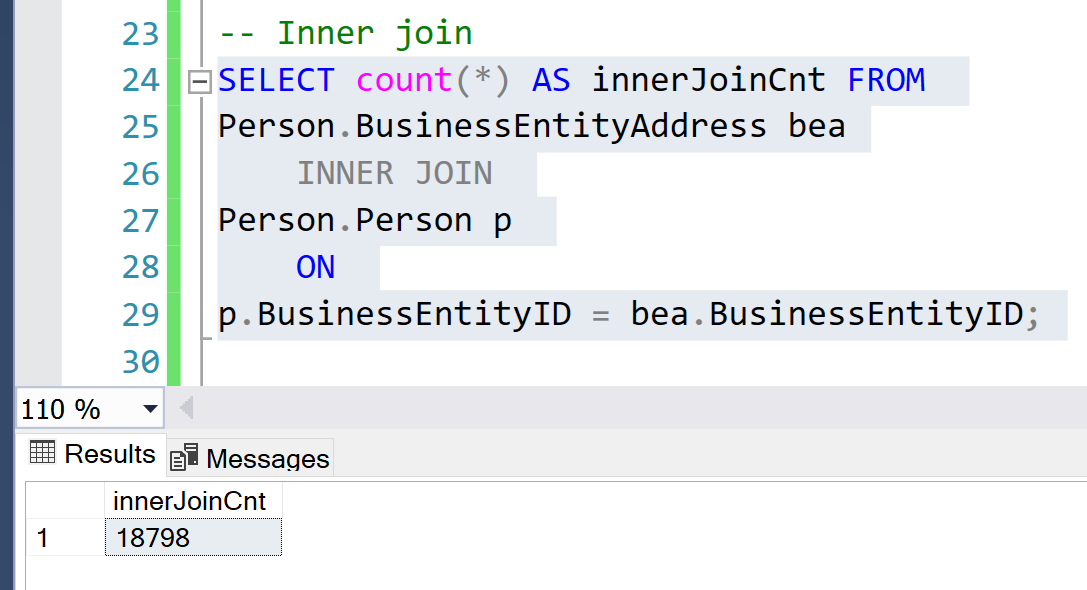
Person.BusinessEntityAddress bea

INNER JOIN

Person.Person p

ON

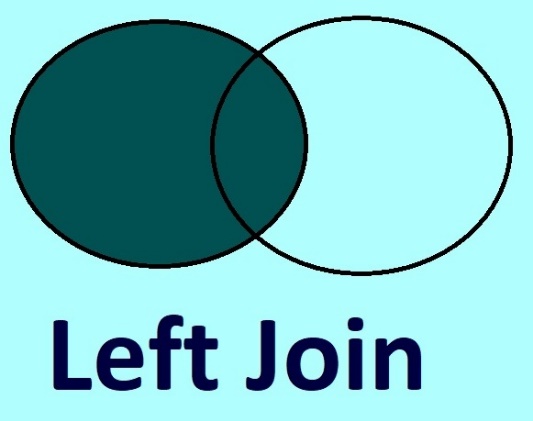
p.BusinessEntityID = bea.BusinessEntityID;



Inner Join it will return only the matching rows in both Person and BusinessEntityAddress table.

Left Join or Left Outer Join – This is similar to Set Difference.

Here we will get the matching rows in addition to the non-matching rows from the Left table.



**Get the matching rows + non-matching rows from the Left table**

SELECT count(\*) AS leftJoinCnt FROM

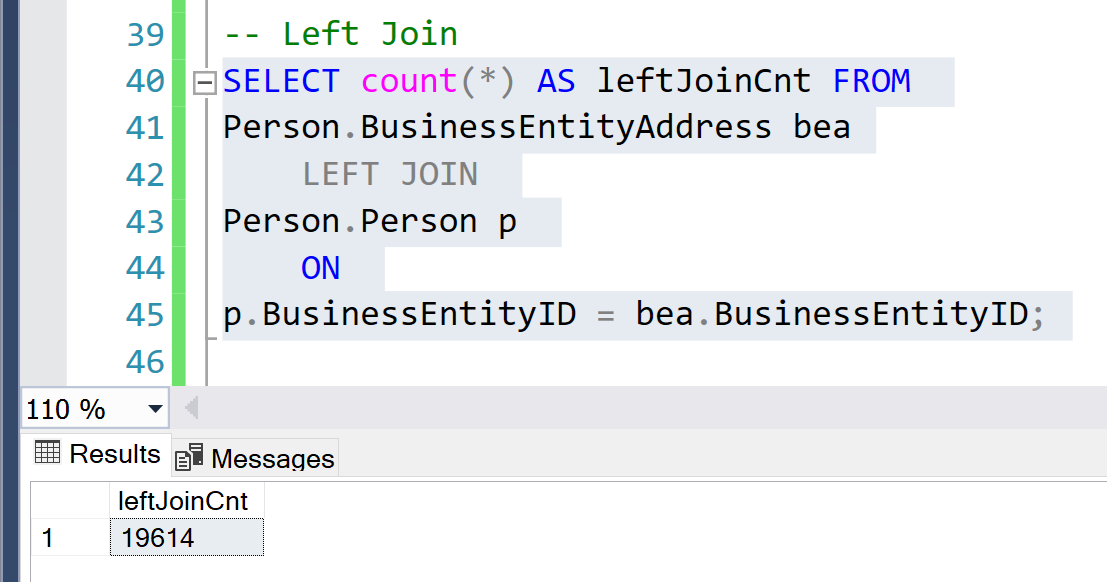
Person.BusinessEntityAddress bea

LEFT JOIN

Person.Person p

ON

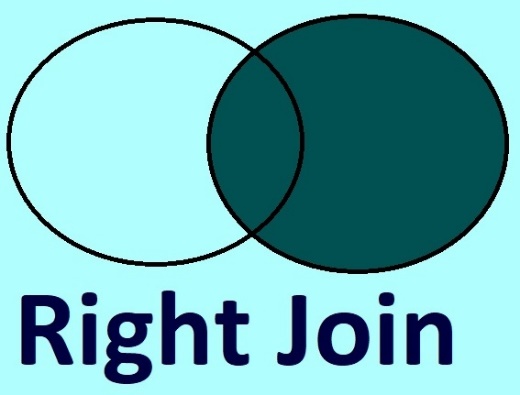
p.BusinessEntityID = bea.BusinessEntityID;



Here it will return all the rows from the BusinessEntityAddress table and only the matching rows from Person table.

Right Join or Right Outer Join – This is similar to Set Difference.

Here we get the matching rows in addition to the non-matching rows from the Right table.



**Get the matching rows + non-matching rows from the Right table**

SELECT count(\*) AS rightJoinCnt FROM

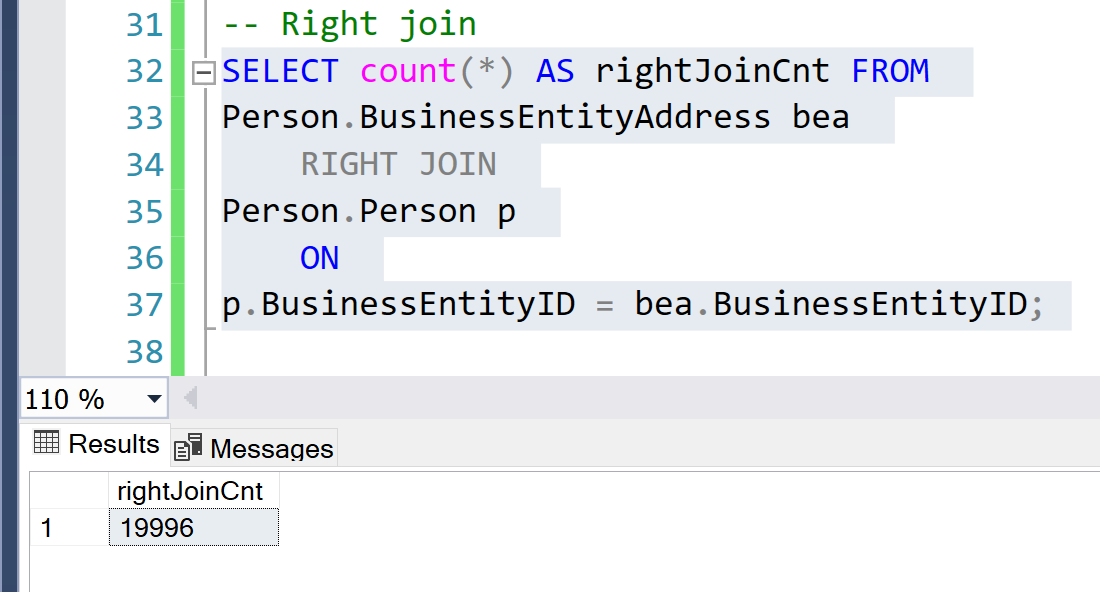
Person.BusinessEntityAddress bea

RIGHT JOIN

Person.Person p

ON

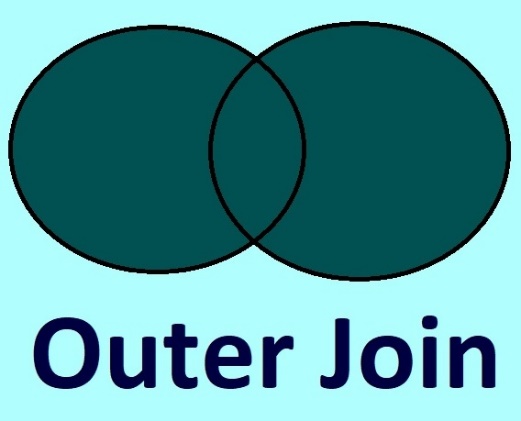
p.BusinessEntityID = bea.BusinessEntityID;



Here it will return all the rows from the Person table and only the matching rows from BusinessEntityAddress table.

Outer Join – This is similar to Union in Sets.

Here we get the matching as well as non-matching rows from the tables.



**Get the matching rows + non-matching rows from all tables**

SELECT count(\*) AS outerJoinCnt FROM

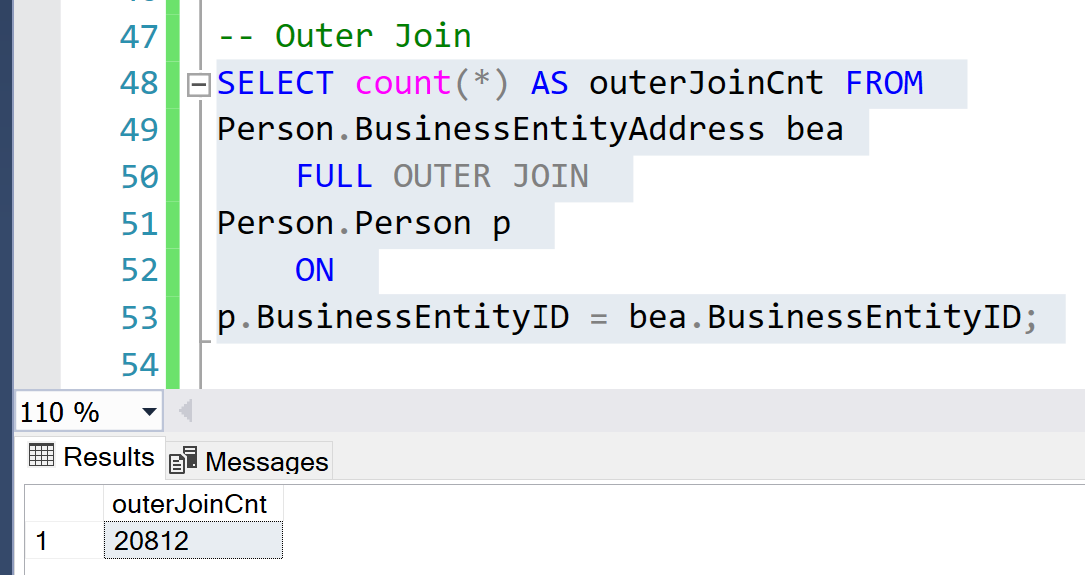
Person.BusinessEntityAddress bea

FULL OUTER JOIN

Person.Person p

ON

p.BusinessEntityID = bea.BusinessEntityID;



Here it will return All the rows from Person and BusinessEntityAddress table.

Cross Join – Here we get the cartesian product

**Cartesian Product = number of rows of table1 \* number of rows of table2**

Syntax for Cross Joins:

SELECT columnList

FROM table1

CROSS JOIN table2

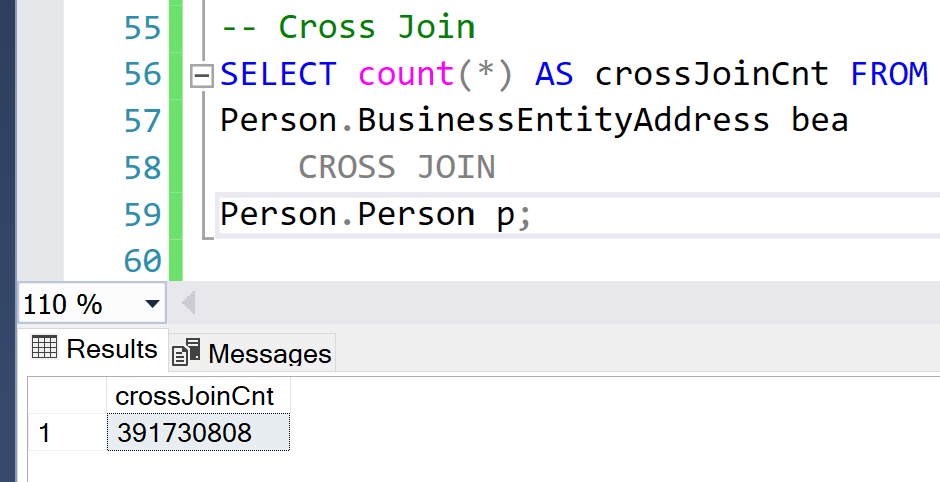
Since Cross Joins give us the Cartesian Product, we don’t need to specify a condition

SELECT count(\*) AS crossJoinCnt FROM

Person.BusinessEntityAddress bea

CROSS JOIN

Person.Person p;



Cartesian Product:

