Inner Join

After completing this module, you will be able to:

- **Project columns from many tables within the same projection.**
- **Distinguish between Subqueries and Inner Joins.**
- **Discuss differences in styles for coding join syntax.**
- **Contrast inner joins with cross joins.**
- **Join a table to itself (Self Join).**
- **Identify pitfalls associated with incorrect aliasing.**
- **Identify problems associated with many-to-many joins.**

Inner Join Concepts

- Inner joins project values based upon column values of one table matching corresponding column values of another table <u>based on equality</u>.
- To get a report of employee number, last name, and department name, you would need to join the employee table and the department table.
- Department number is the common column that determines the way data in these two tables will be matched.
- Note the one-to-many relationship for the join condition.

EMPLOYEE

EMP NUM	MGR EMP NUM	DEPT NUM	JOB CODE	LAST NAME	FIRST NAME	HIRE DATE	BIRTH DATE	SAL AMT
PK	FK	FK	FK					
1006	1019	301	312101	Stein	John	761015	531015	2945000
1008	1019	301	312102	Kanieski	Carol	770201	580517	2925000
1005	0801	403	431100	Ryan	Loretta	761015	550910	3120000
1004	1003	401	412101	Johnson	Darlene	761015	460423	3630000
1007	1005	403	432101	Villegas	Arnando	770102	370131	4970000
1003	0801	401	411100	Trader	James	760731	470619	3785000

employee_number	last_name	department_name
1004	Johnson	customer support
⋮	⋮	⋮

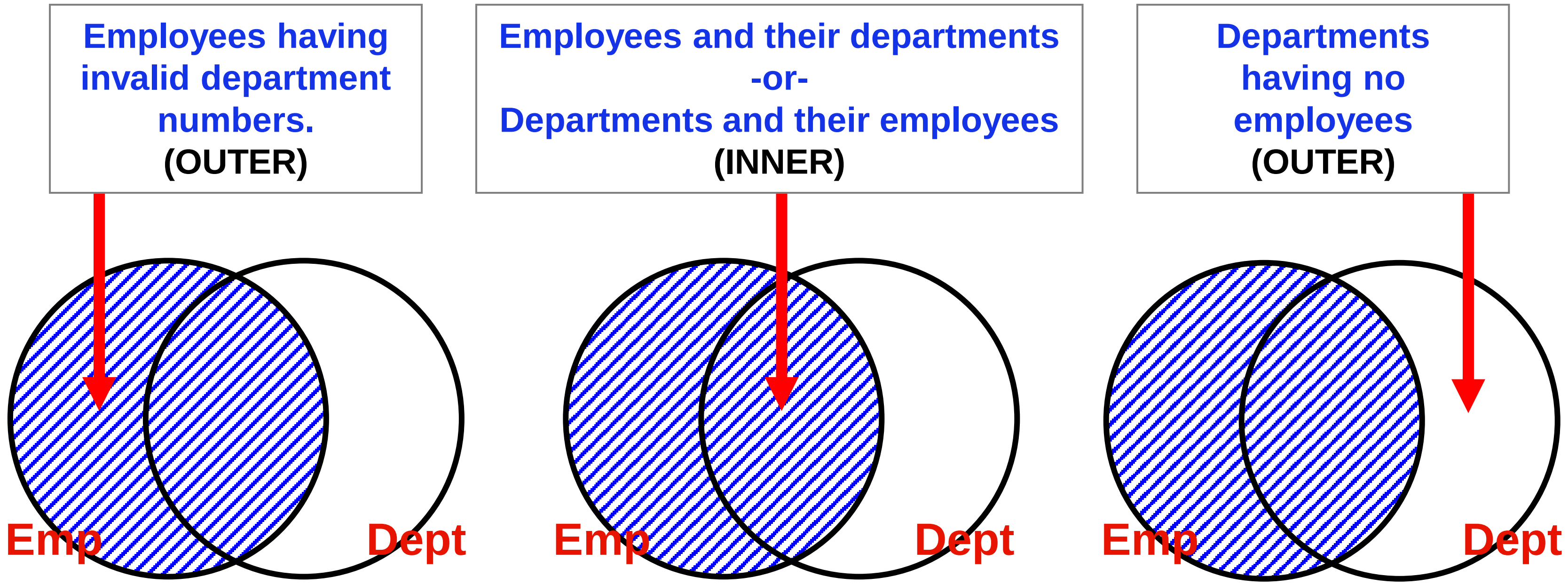
DEPARTMENT

DEPT NUM	DEPT NAME	BUDGET AMOUNT	MGR EMP NUM
PK			FK
501	marketing sales	80050000	1017
301	research and devel.	46560000	1019
302	product planning	22600000	1016
403	education	93200000	1005
402	software support	30800000	1011
401	customer support	98230000	1003
201	technical operations	29380000	1025

Inner Join vs. Subquery

Contrast the following bullets with what we know about subqueries.

- **Inner Joins return only inner result sets.**
- **Inner joins can be used to project from any joined table.**


Employees having invalid department numbers.
(OUTER)
Employees and their departments
-or-
Departments and their employees
(INNER)
Departments having no employees
(OUTER)
Emp
Dept
Emp
Dept
Emp
Dept

A Comparison

- Note the differences between the syntax used for a subquery and that for the join.
- The join condition must evaluate "True" in order to project column values.
- The SELECT *, in the case of the join, will project all columns from both tables for comparisons that evaluate "True."

Subquery:

```
SELECT Last_Name
FROM Employee
WHERE Department_Number IN
        (SELECT Department_Number
           FROM Department);
```

Join Equivalent:

```
SELECT Last_Name, Department_Name
FROM Employee, Department
WHERE    Employee.Department_Number =
         Department.Department_Number;
```

Recall that for a subquery:		However, for an Inner Join:
1. You can only project columns from the outer table.	⟷	1. You can project columns from any table.
2. A distinct list guarantees a one-to-many relationship between the inner and outer table.	⟷	2. Does not guarantee a one-to-many relationship between the tables.
3. Can return an inner result (using IN) or an outer result (using NOT IN)	⟷	3. Can only return an inner result.

Table Name Qualifications and Aliasing

Just as you can alias column names, you may also alias table names.
Without double-quotes, aliases:

- **May not contain non-standard characters.**
- **May not contain key-words.**

```
SELECT  employee.Last_Name,
        First_Name,
        Employee.Department_Number,
        d.Manager_Employee_Number
FROM Employee, Department  AS d
WHERE   Employee.Department_Number = d.Department_Number;
```

Qualification not required.

```
SELECT  e.Last_Name,
        First_Name,
        e.Department_Number,
        d.Manager_Employee_Number
FROM Employee  e, Department  d
WHERE   e.Department_Number = d.Department_Number;
```

Qualification required.

Varied Forms of INNER Join

Another form for doing an inner join is the ANSI 92 syntax.
Both return the same result.
Both are optimized equally.

(Implicit Form)

```
SELECT    e.Last_Name,
          e.First_Name,
          e.Department_Number,
          d.Manager_Employee_Number
FROM Employee  e, Department  d
WHERE     e.Department_Number = d.Department_Number
AND       e.Last_Name = 'Brown';
```

Equivalent Results

(Explicit Form)

```
SELECT    e.Last_Name,
          e.First_Name,
          e.Department_Number,
          d.Manager_Employee_Number
FROM Employee  AS e   INNER JOIN  Department  AS d
ON        e.Department_Number = d.Department_Number
WHERE     e.Last_Name = 'Brown';
```

Many-Table INNER Joins

You can join these 3 tables like this.
Notice the uniqueness involved.
If the tables have only the rows shown, what will this return?
How would you write this in explicit form?

```
SELECT  e.Last_Name, d.Department_Name, j.Description
FROM    Employee  e, Department  d, Job  j
WHERE   e.Department_Number = d.Department_Number
AND     e.Job_Code = j.Job_Code
```

Employee

Last Name	Department Number	Job Code
Jones	100	6666
Smith	200	7777
Brown	300	8888
Adams	400	9999

Department

Department Number *(Unique)*	Department Name
100	Sales
200	Marketing
600	Support

Job

Job Code *(Unique)*	Description
6666	Manager
5555	President
8888	Lead

Varied Forms of Many-Table Inner Joins

```
SELECT    Last_Name, d.Department_Name, j.Description
FROM Employee  e, Department  d, Job  j
WHERE     e.Department_Number = d.Department_Number
AND       e.Job_Code = j.Job_Code
AND       j.Description LIKE '%soft%'
AND       d.Budget_Amount > 350000;
```

There are many different forms one may use when writing inner joins.

State the business concern for these queries.

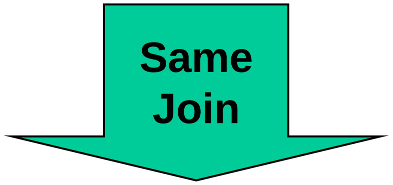
```
SELECT    e.Last_Name  AS "Ln", d.Department_Name  AS Dn,  j.Description AS "Desc"
FROM Employee  AS e   JOIN  Department  AS d
ON        e.Department_Number = d.Department_Number
                      JOIN   Job  AS j
ON        e.Job_Code = j.Job_Code
WHERE     j.Description LIKE '%soft%'
AND       d.Budget_Amount > 350000;
```

```
SELECT    e.Last_Name  AS "Ln", d.Department_Name  AS Dn, j.Description AS "Desc"
FROM       Department  d          JOIN
           Employee  e            JOIN
          Job  j
ON        e.Job_Code = j.Job_Code
ON        e.Department_Number = d.Department_Number
WHERE     j.Description LIKE '%soft%'
AND       d.Budget_Amount > 350000;
```

Using Parentheses to Understand Order

- Correct placement of parentheses can illustrate how to correctly place join conditions.
- Again, note that the key word INNER is optional.
- Also note that the number of join conditions is the number of tables minus 1
- Whether aliasing or not, is to always best to use column qualifiers to match columns to tables.

```
SELECT    e.Last_Name  AS "Ln", e.Department_Number  AS Dn,  j.Description AS "Desc"
FROM      Employee  AS e   JOIN  Department  AS d
ON        e.Department_Number = d.Department_Number
                      JOIN   Job  AS j
ON        e.Job_Code = j.Job_Code;
```



```
SELECT    e.Last_Name  AS "Ln", e.Department_Number  AS Dn,  j.Description AS "Desc"
FROM      ( ( Employee  AS e   JOIN  Department  AS d
ON        e.Department_Number = d.Department_Number )
                      JOIN   Job  AS j
ON        e.Job_Code = j.Job_Code );
```

Using Parentheses with Other Forms

Note in the example below that the key word INNER is optional. Also note that the number of join conditions is the number of tables minus 1 and that best practice, whether aliasing or not, is to always qualify, whether required or not, to match columns to tables.

```
SELECT    e.Last_Name  AS "Ln", e.Department_Number  AS Dn, j.Description AS "Desc"
FROM       Department  d           JOIN
           Employee  e             JOIN
          Job  j
ON        e.Job_Code = j.Job_Code
ON        e.Department_Number = d.Department_Number;
```

Same Join

```
SELECT    e.Last_Name  AS "Ln", e.Department_Number  AS Dn, j.Description AS "Desc"
FROM       ( Department  d         JOIN
           ( Employee  e           JOIN
             Job  j
ON        e.Job_Code = j.Job_Code )
ON        e.Department_Number = d.Department_Number );
```

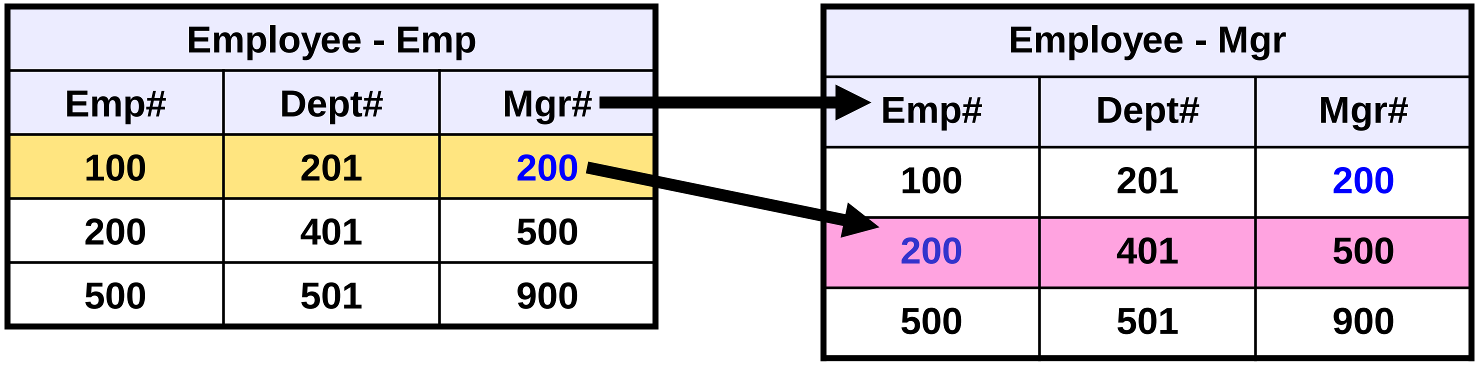
Self Joins

Sometimes it may be necessary to join a table to itself.

- **Aliasing of at least one version of the table is necessary.**
- **In the query below, we project the name of the employee and name of the manager -- *as different rows in the same table* -- onto the same result row.**

Display the last name and first names of employees along with the last name and first names of their managers for those working in departments 201 and 301.

```
SELECT      Emp.Last_Name, Emp.First_Name, Mgr.Last_Name, Mgr.First_Name
FROM        Employee Emp JOIN Employee Mgr
ON          Emp.Manager_Employee_Number = Mgr. Employee_Number
WHERE Emp.Department_Number IN (201, 301);
```



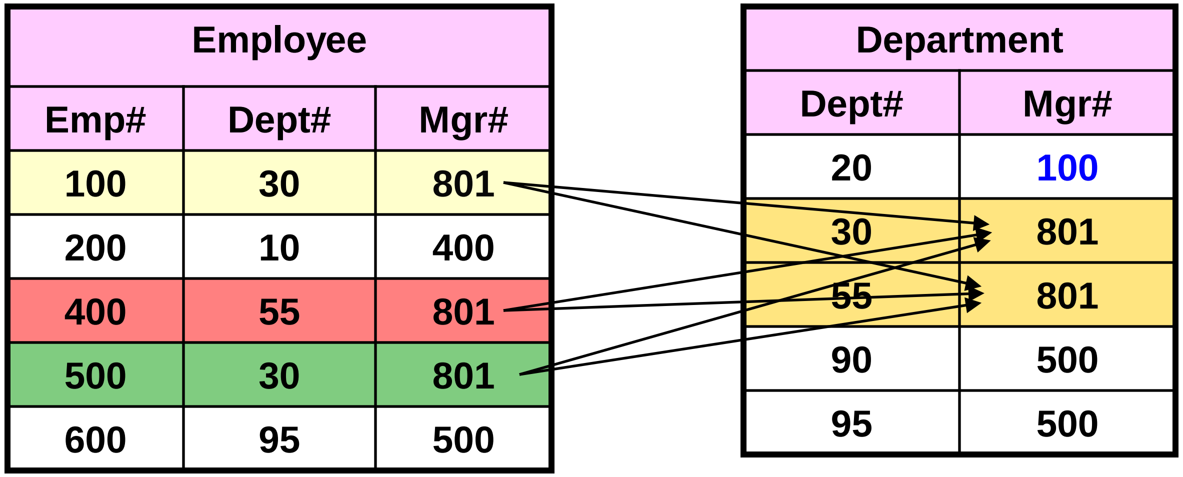
Employee - Emp		
Emp#	**Dept#**	**Mgr#**
100	**201**	**200**
200	**401**	**500**
500	**501**	**900**

Employee - Mgr		
Emp#	**Dept#**	**Mgr#**
100	**201**	**200**
200	**401**	**500**
500	**501**	**900**

Guaranteeing Uniqueness

When joining a many-to-many relationship, unintended result rows can be projected! The example below depicts 3 rows joining to 2, producing 6 result rows!

```
SELECT   e.Employee_Number  AS Emp#,
         d.Department_Number AS Dept#
FROM     Employee  e, Department  d
WHERE    e.Manager_Employee_Number = d.Manager_Employee_Number
AND      e.Manager_Employee_Number = 801;
```



Employee

Emp#	Dept#	Mgr#
100	30	801
200	10	400
400	55	801
500	30	801
600	95	500

Department

Dept#	Mgr#
20	100
30	801
55	801
90	500
95	500

Result

Emp#	Dept#
100	30
100	55
400	30
400	55
500	30
500	55

IN vs. Inner Join

Find employees have valid department numbers.

Subquery form:

```
SELECT    Employee_Number,
          First_name
FROM      Employee
WHERE Department_Number IN
(SELECT Department_Number FROM Department);
```

Inner Join form:

```
SELECT    Employee_Number,
          First_name
FROM      Employee  e JOIN Department  d
ON        e.Department_Number = d.Department_Number;
```

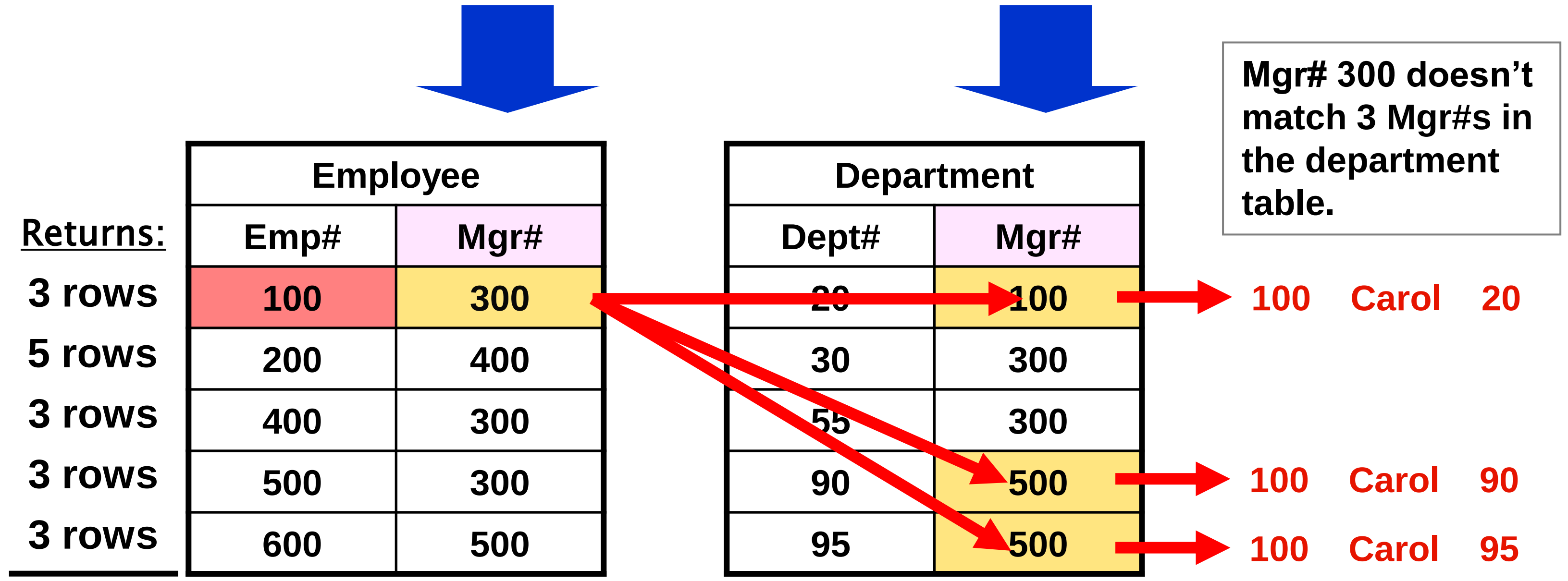
Note that you may only rewrite a join as a subquery if you are only projecting columns from one table!

NOT IN vs. Inner Join

The NOT IN subquery would have no issue with obtaining the result intended here.

Find employees whose managers are not department managers.

```
SELECT   Employee_Number,
         First_name
FROM     Employee  e JOIN Department  d
ON       e.Manager_Employee_Number <> d.Manager_Employee_Number;
```



Mgr# 300 doesn't match 3 Mgr#s in the department table.

Returns:	Emp#	Mgr#
3 rows	100	300
5 rows	200	400
3 rows	400	300
3 rows	500	300
3 rows	600	500

Employee

Dept#	Mgr#
20	100
30	300
55	300
90	500
95	500

Department

100	Carol	20
100	Carol	90
100	Carol	95

17 rows total

Cross Join

A CROSS join is a join where no join condition is specified.
Since no qualification exists, the database establishes a condition of “WHERE 1=1”.
Since this condition is true for each and every comparison, the following occurs.

```
SELECT Employee_Number,
       Last_Name
FROM   Employee  e, Department  d;
```

Equivalent

```
SELECT Employee_Number,
       Last_Name
FROM   Employee  e CROSS JOIN
       Department  d;
```

	Employee	
	Emp#	**Last_Name**
1	100	Smith
2	200	Jones
3	400	Adams

Department	
Dept#	**Mgr#**
20	100
30	300
55	300

Project the column values where 1=1 is true.

Result

	100 Smith
1	100 Smith
	100 Smith
	200 Jones
2	200 Jones
	200 Jones
	400 Adams
3	400 Adams
	400 Adams

Mistakes on Table Aliasing

- **Be careful! Do not alias a table and then use the name instead of the alias.**
- **In the examples below, the first one will fail due to a syntax error *(ANSI 92)*.**
- **The second will cause a 4-table join, one of which is a self join between Dept *(the aliased Department table)* and Department!**

```
SELECT  Last_Name, First_Name,
        Department_Name, Description
FROM    Employee AS Emp JOIN Department Dept
ON      Emp.Department_Number = Dept.Department_Number
                JOIN Job
ON      Emp.Job_Code = Job.Job_Code;
```

```
SELECT  Last_Name, First_name,
        Department_Name, Description
FROM    Employee Emp, Department AS Dept, Job
WHERE   Emp.Department_Number = Dept.Department_Number
AND     Emp.Job_Code = Job.Job_Code;
```

Mistakes on Column Aliasing

Both forms of joins cause bad self joins when referring to the table name in the select list instead of the alias!

```
SELECT  Emp.Last_Name, First_Name,
        Department_Name, Description
FROM    Employee AS Emp JOIN Department Dept
ON      Emp.Department_Number = Dept.Department_Number
                JOIN Job
ON      Emp.Job_Code = Job.Job_Code;
```

```
SELECT  Emp.Last_Name, First_name,
        Department_Name, Description
FROM    Employee Emp, Department Dept, Job
WHERE   Emp.Department_Number = Dept.Department_Number
AND     Emp.Job_Code = Job.Job_Code;
```

Module 3: Summary

- **Columns values may be projected from any table of a join.**
- **Subqueries and inner joins can both return inner result sets.**
- **Inner joins have both an implicit form and an explicit form.**
- **Inner joins typically involve one-to-many relationships based on equality.**
- **A table may be joined to itself.**
- **Incorrect table and column references can cause incorrect result sets.**
- **Inner joins can not return outer (NOT IN) result sets as can subqueries.**

Module 3: Review Questions

True or False:

1. For inner joins, each FROM clause requires an ON clause for join conditions.

2. Referencing a WHERE clause is invalid for the explicit form of inner join.

3. Many-to-many relationships are allowed with inner joins.

4. When performing a self join, table aliasing is required.

s

5. You can not write an inner join without qualifying at least some columns.

6. The explicit form of inner join can reject some uses of incorrect qualifications.

7. The implicit form of inner join is not ANSI standard.

[illegible]

Module 3: Lab Exercise

1) List all employees by name, the name of their department, their original salary, and salary again with a ten percent increase, for those working in departments with budgets > $400,000.00. Use the implicit form of inner join.

2) Find the department names and employee names for employees that have both an "i" and an "e" in their last name. Use the explicit form of inner join.

3) List department names that have people working in them whose job description has the word "manager" in it. List the employee names as well.

4) Write a cross join that lists all possible combinations of first names and last names from employee.