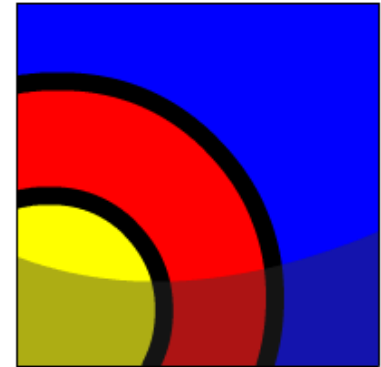


# **Review of Joins**

# What Will I Learn?

**In this lesson, you will learn to:**

- Determine the correct join syntax to use given a scenario requiring the join of data from two or more tables.






### Why Learn It?

Knowing when to use the correct join syntax to meet the needs stated in a business scenario requiring the join of data is very important to your success. This lesson will afford you the opportunity to review the join syntax.





# Try It / Solve It

## Classroom Activity

Try every example listed below. Confirm that your results match the expected result. If you need help, ask another student or the teacher. All example code is based on the Oracle database.

### 1. Cartesian Product

```
SELECT last_name, department_name  
FROM employees, departments;  
All rows will show.
```

### 2. Oracle Proprietary Joins (equivalent ANSI joins given in parenthesis)

#### a. Equijoin (Natural Join, Join .. Using, Join .. On)

```
SELECT e.employee_id, e.last_name, e.department_id, d.department_name  
FROM employees e, departments d  
WHERE e.department_id = d.department_id;
```

Selects rows from both tables where department IDs match.  
Sometimes referred to as an Inner Join

## Try It / Solve It

### Classroom Activity

#### b. Nonequijoin (Join .. On)

```
SELECT e.employee_id, e.last_name, e.salary, j.grade_level  
FROM employees e, job_grades j  
WHERE e.salary >= j.lowest_sal  
AND e.salary <= j.highest_sal;
```



This displays the grade level for each employee based on salary.

#### c. Outer Joins (Right Outer Join, Left Outer Join)

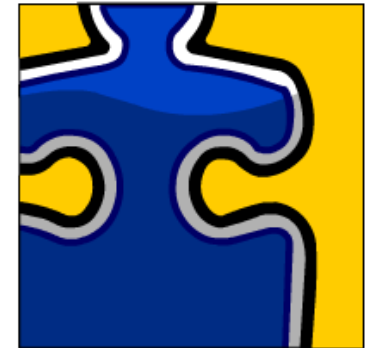
```
SELECT e.employee_id, e.last_name,  
e.department_id, d.department_name  
FROM employees e, departments d  
WHERE e.department_id (+) = d.department_id;
```

Retrieves all data in the departments table, including nulls. Returns departments with employees and departments without employees.

## Try It / Solve It

### Classroom Activity

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e, departments d  
WHERE e.department_id = d.department_id(+);
```



Retrieves all data in the employees table including nulls. Returns employees who are assigned to a department as well as employees who are not assigned to a department.

### d. Self-Joins (Join .. On)

```
SELECT e.employee_id, e.last_name, m.employee_id, m.last_name  
FROM employees e, employees m  
WHERE e.manager_id = m.employee_id;
```

All managers are employees, so you need to look to the same table (EMPLOYEES) to check for the managers.

# Try It / Solve It

## Classroom Activity

### 3. ANSI SQL Standard Syntax (equivalent Oracle specific joins given in parenthesis)

#### a. Cross Join (Cartesian Product)

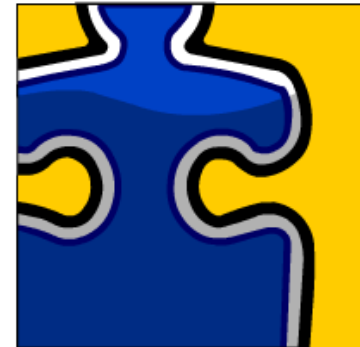
```
SELECT last_name, department_name  
FROM employees CROSS JOIN departments;
```

All rows will show.

#### b. Natural Join (Equijoin)

```
SELECT employee_id, last_name, department_name  
FROM employees NATURAL JOIN departments;
```

Joins by column names and data types that are identical in each table. Both the employees and departments tables have the columns department\_id and manager\_id. Therefore, the query will return the rows where the values in both columns match.



# Try It / Solve It

## Classroom Activity

### c. Joins .. Using (Equijoin)

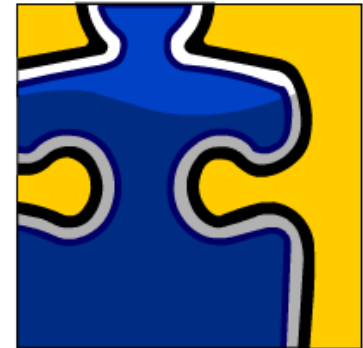
```
SELECT employee_id, last_name, department_name  
FROM employees JOIN departments  
USING (department_id);
```

Joins by column names and data types that are identical in each table but USING statement limits to one column.

### d. Join .. On

```
SELECT e.employee_id, e.last_name, d.department_id, d.location_id  
FROM employees e JOIN departments d  
ON (e.department_id = d.department_id);
```

All employees and their work locations.





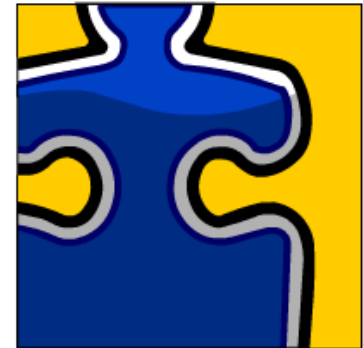
# Try It / Solve It

## Classroom Activity

### e. Join .. On (Non equijoin)

```
SELECT e.employee_id, e.last_name, e.salary, j.grade_id  
FROM employees e JOIN job_grades j  
ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

This displays the grade level for each employee based on salary.



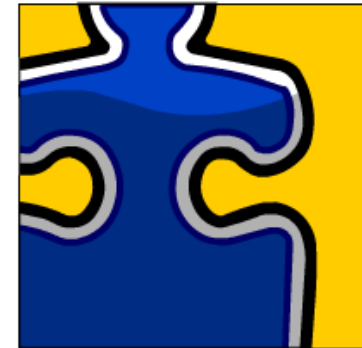
# Try It / Solve It

## Classroom Activity

### f. Outer Joins (+)

Right Outer Join

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e RIGHT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```



Retrieves all data in the right table (DEPARTMENTS), including nulls. Returns departments with employees and departments without employees.

Left Outer Join

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e LEFT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

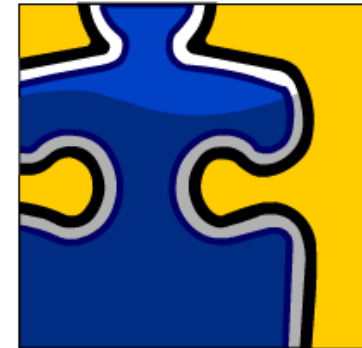
Retrieves all data in the left table (EMPLOYEES). Returns employees who are assigned to a department as well as employees who are not assigned to a department.

## Try It / Solve It

### Classroom Activity

#### 3. Full Outer Join (No comparable Oracle specific Joins)

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e FULL OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```



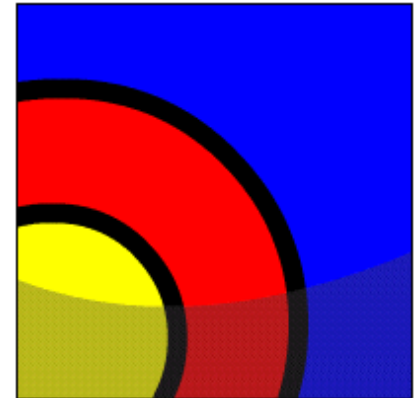
Retrieves all data in the left table and all data in the right table. This includes departments with employees and departments without employees. It also includes employees who are assigned to a department as well as employees who are not assigned to a department.



# Summary

**In this lesson you have learned to:**

- Determine the correct join syntax to use given a scenario requiring the join of data from two or more tables



# Summary

## Practice Guide

The link for the lesson practice guide can be found in the course outline.

