Database Design & Applications

The Database Language - SQL Join

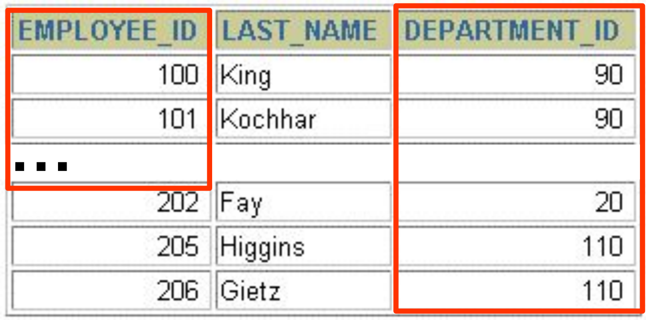
Displaying Data from Multiple Tables

Objectives

- Write SELECT statements to access data from more than one table using equality and nonequality joins
- View data that generally does not meet a join condition by using outer joins
- Join a table to itself by using a self join

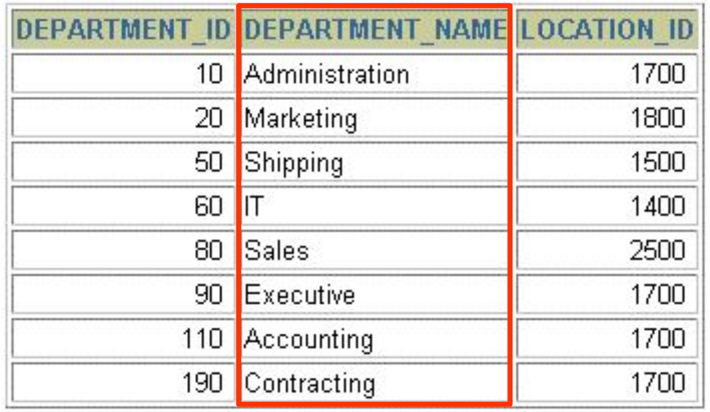
Obtaining Data from Multiple Tables

EMPLOYEES

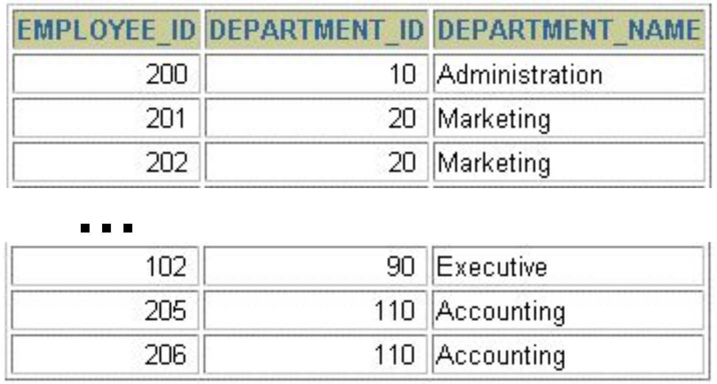


EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS



DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700



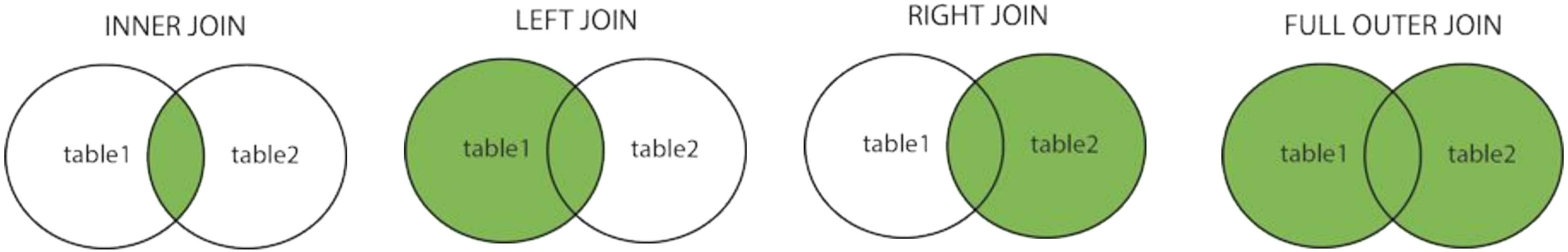
EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

SQL Joins

- A SQL join without a relationship called Cartesian Product
- All rows in the first table are joined to all rows in the second table
- Also known as CROSS JOIN
- A SQL join with a relationship may be one of two types:
  - A SQL equijoin (also known as a natural join) creates a relationship between two tables based on a comparison of values found in one or a set of columns in one table and one or an equal set of columns in another table.
  - A SQL non-equijoin (also known as a θ-join) effects a relation between two tables based on a filtered CROSS JOIN between two tables. This type of join can be a range comparison using the BETWEEN operator or a comparison of column values that uses an inequality operator.

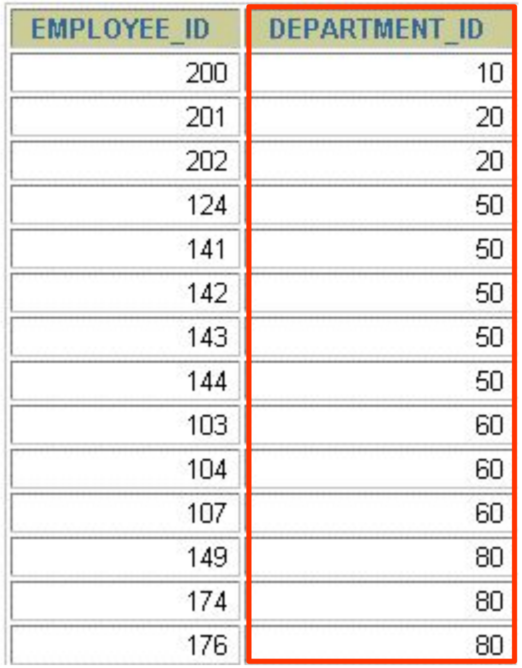
Different Types of SQL JOINs

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table



Inner Join

EMPLOYEES

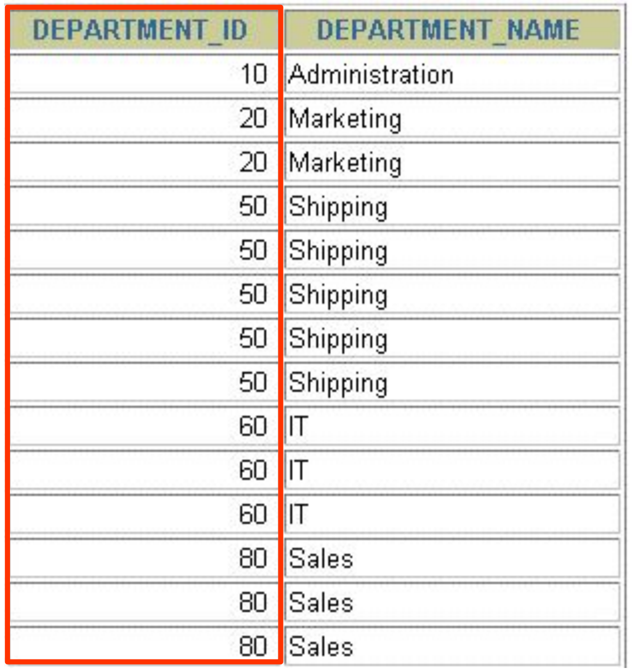


EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

…

Foreign key

DEPARTMENTS



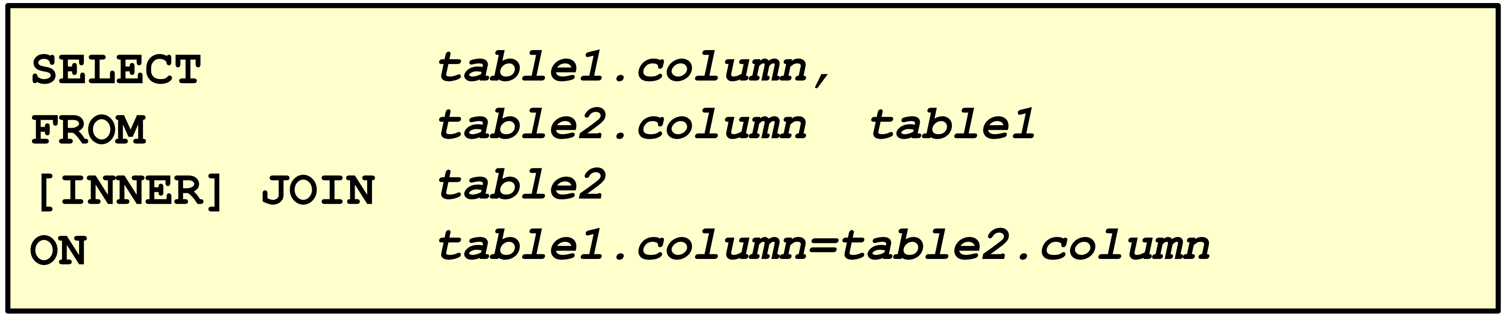
DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

…

Primary key

Inner Join

- The INNER JOIN keyword selects records that have matching values in both tables.



```
SELECT          table1.column,
FROM            table2.column  table1
[INNER] JOIN    table2
ON              table1.column=table2.column
```

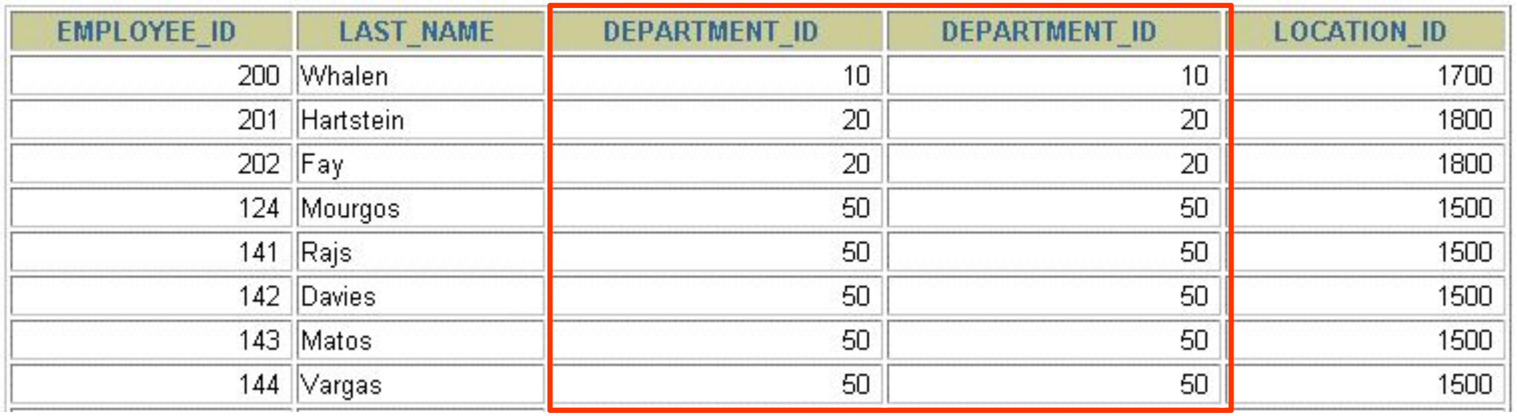
- In SQL, JOIN and INNER JOIN are syntactic equivalents (they can replace each other).

Inner Join

- The join condition for the natural join is basically an equijoin of all columns with the same name.
- To specify arbitrary conditions or specify columns to join, the ON clause is used.
- The join condition is separated from other *search* conditions.

Inner Join

```
SELECT employee.employee_id, employee.last_name,
       employee.department_id, department.department_id,
       department.location_id
FROM        employee
INNER JOIN department
ON employee.department_id = department.department_id
```



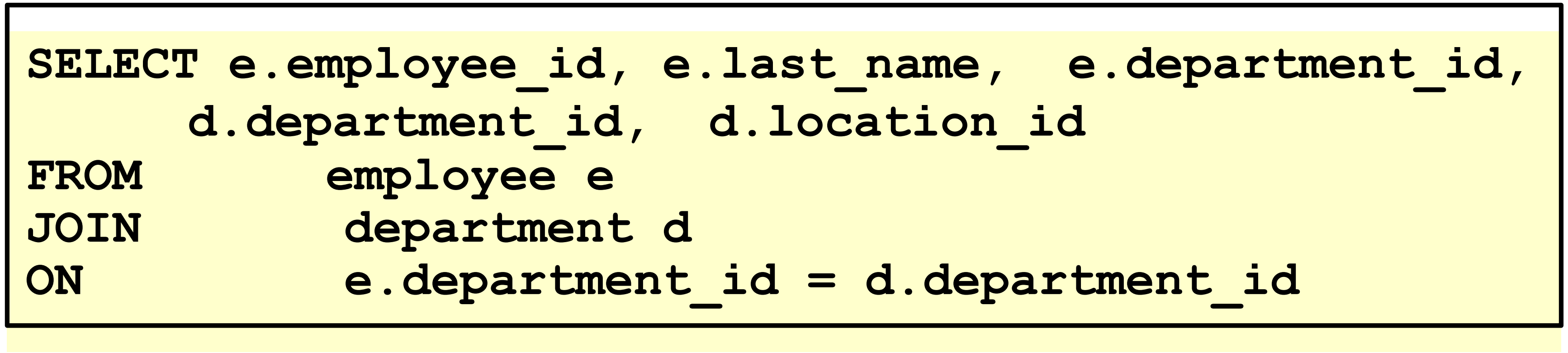
EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500
144	Vargas	50	50	1500

Qualifying Ambiguous Column Names

- Use table prefixes to qualify column names that are in multiple tables.
- Improve performance by using table prefixes.
- Distinguish columns that have identical names but reside in different tables by using column aliases.

Using Table Aliases

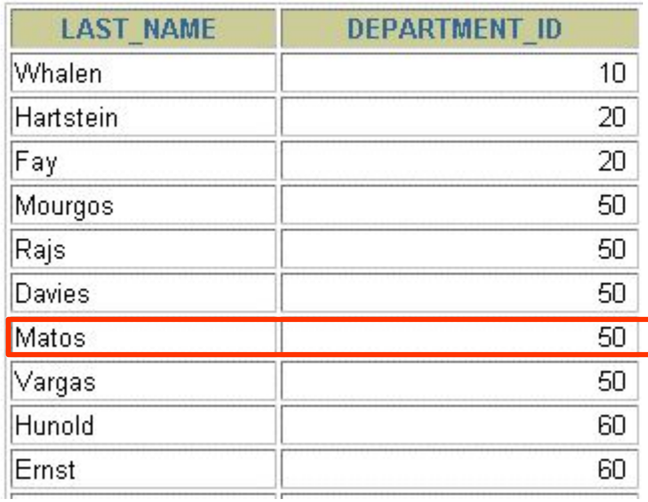
- Simplify queries by using table aliases.
- Improve performance by using table prefixes.



```
SELECT e.employee_id, e.last_name,  e.department_id,
      d.department_id,  d.location_id
FROM      employee e
JOIN       department d
ON         e.department_id = d.department_id
```

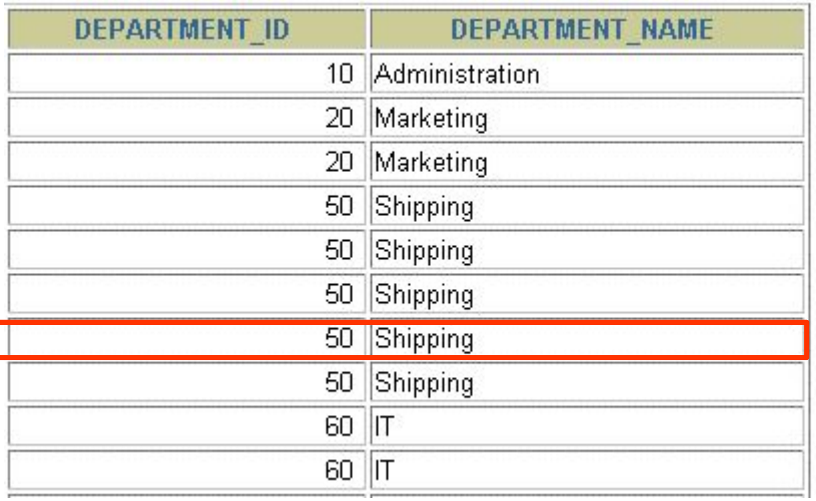
Additional Search Conditions using AND Operator

EMPLOYEES **DEPARTMENTS**



LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

...

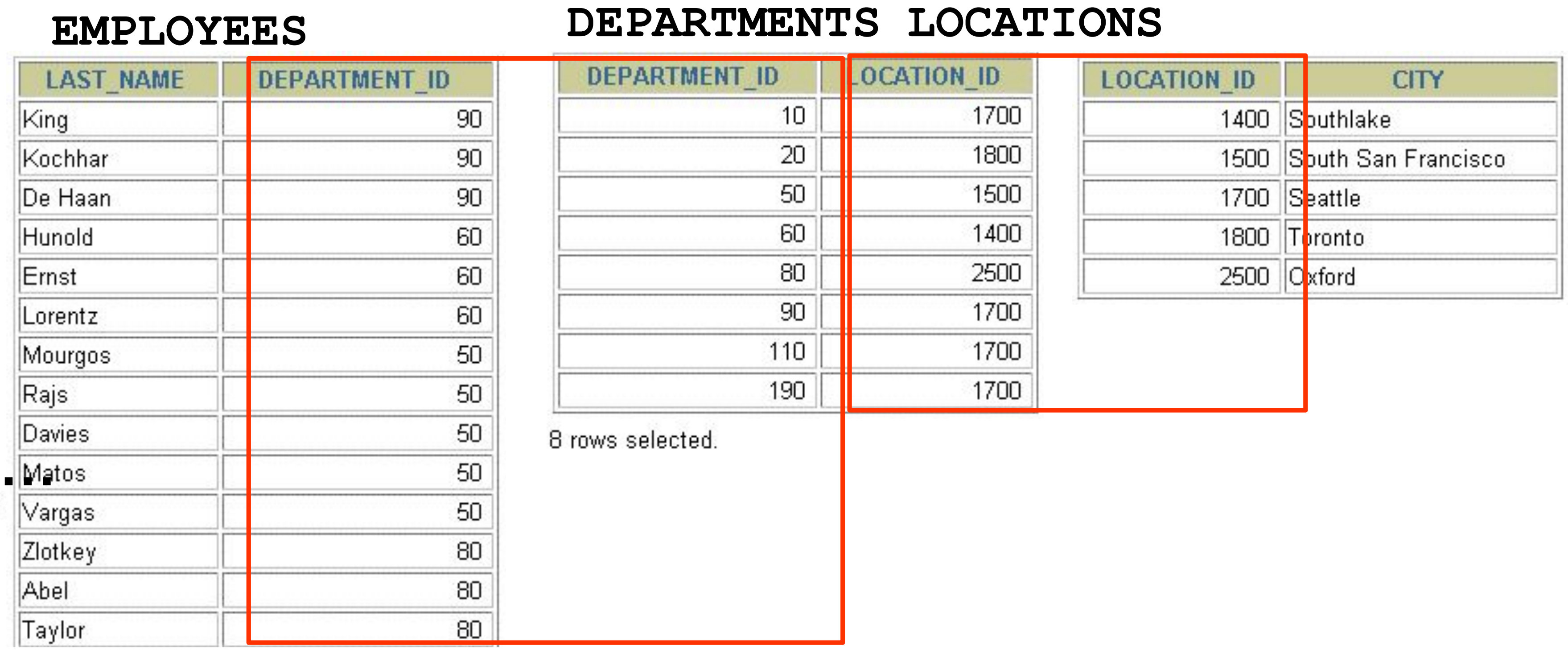


DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT

...

```
 SELECT e.employee_id, e.last_name,  e.department_id,
            d.department_id,  d.location_id
FROM             employee e
INNER JOIN   department d
ON               e.department_id = d.department_id
AND              d.department_id =50
```

Joining More than Two Tables



EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Zlotkey	80
Abel	80
Taylor	80

...

DEPARTMENTS

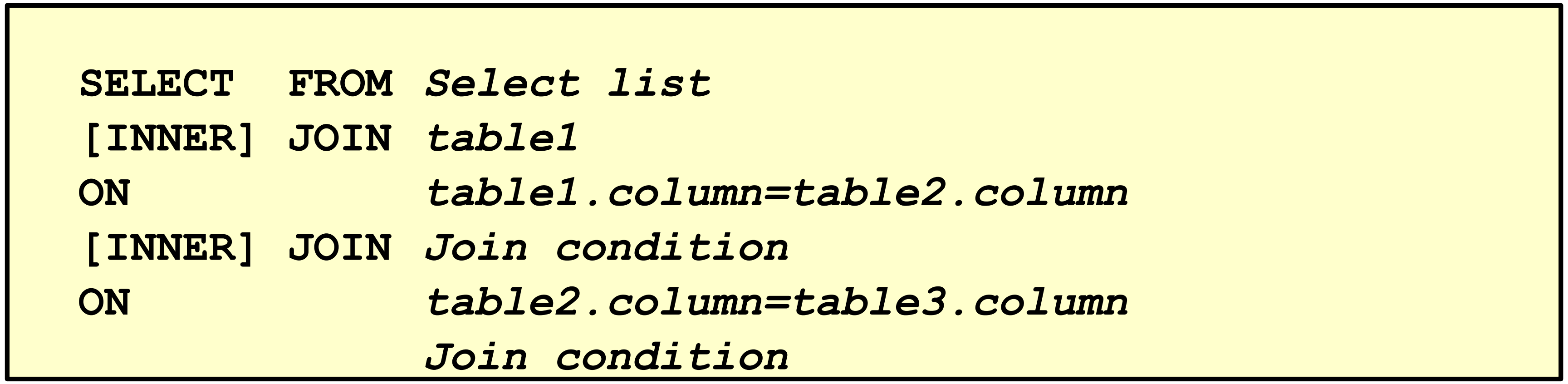
DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

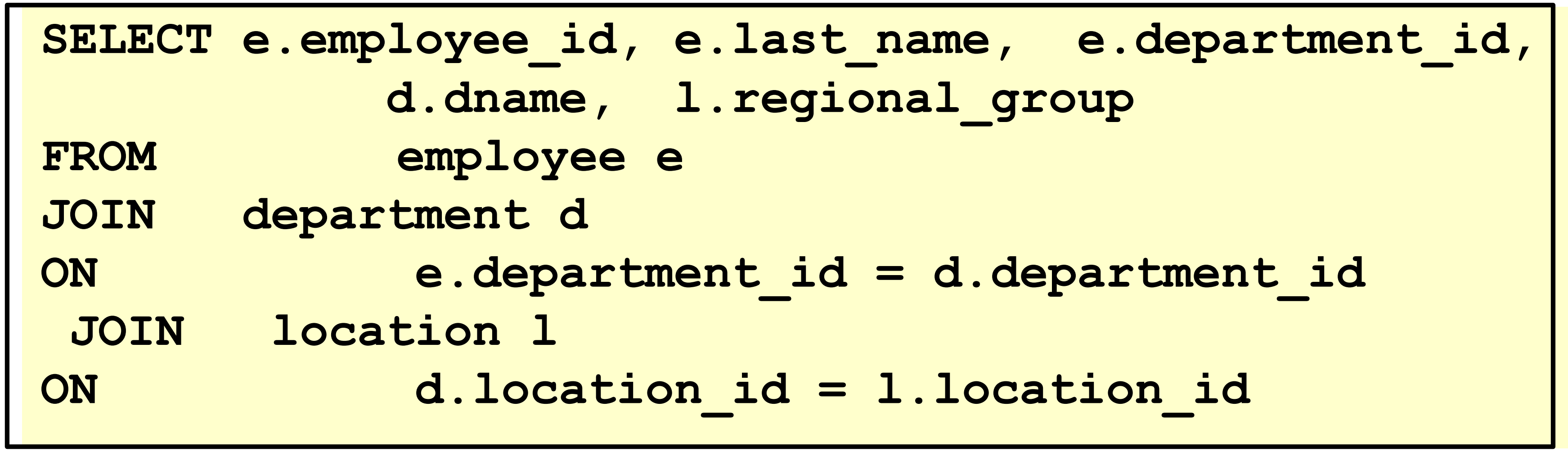
LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

Joining More than Two Tables



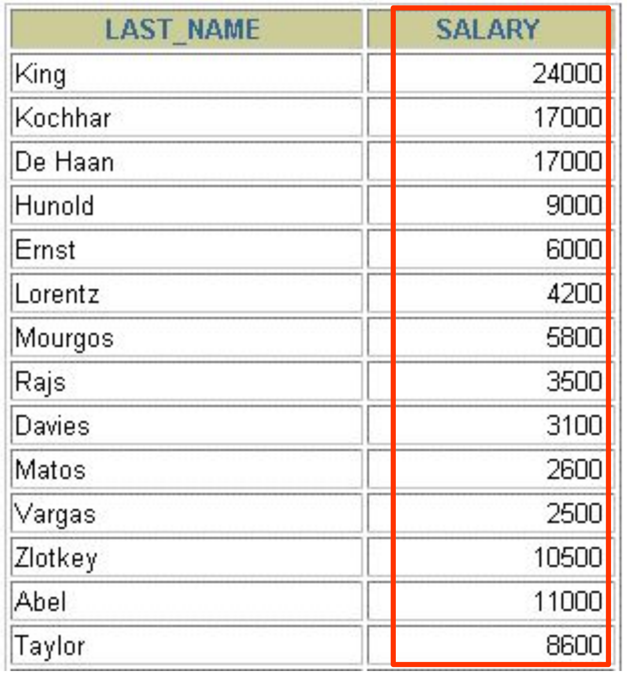
```
SELECT   FROM Select list
[INNER] JOIN table1
ON           table1.column=table2.column
[INNER] JOIN Join condition
ON           table2.column=table3.column
             Join condition
```



```
SELECT e.employee_id, e.last_name,  e.department_id,
            d.dname,  l.regional_group
FROM        employee e
JOIN   department d
ON          e.department_id = d.department_id
 JOIN   location l
ON          d.location_id = l.location_id
```

Non-Equijoins

EMPLOYEES

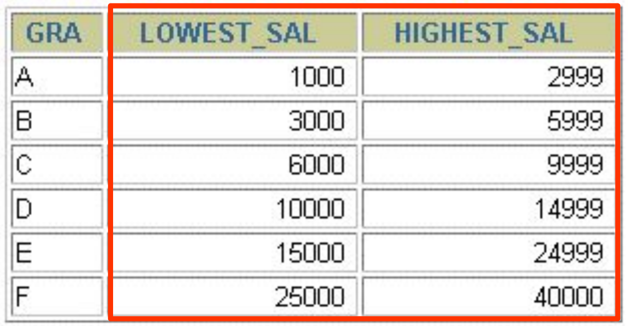


LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

…

20 rows selected.

SALARY_GRADES

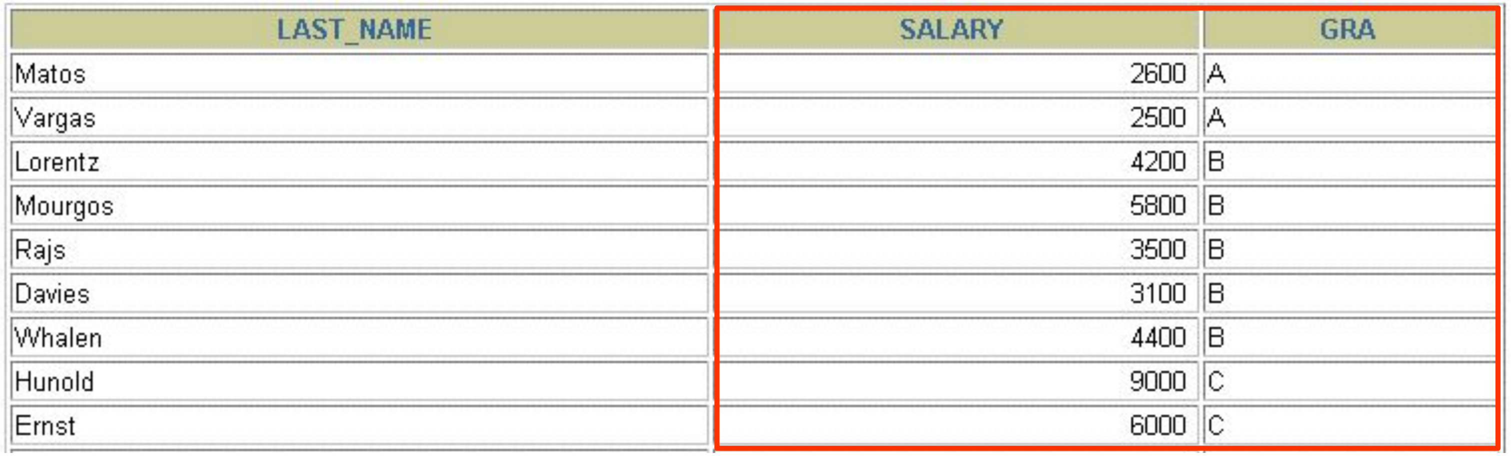


GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

Salary in the EMPLOYEES table must be between lowest salary and highest salary in the JOB_GRADES table.

Retrieving Records with Non-Equijoins

```
SELECT e.last_name, e.salary, s.grade
FROM   employeeS e, salary_grade s
WHERE  e.salary
         BETWEEN j.lowest_sal AND j.highest_sal;
```



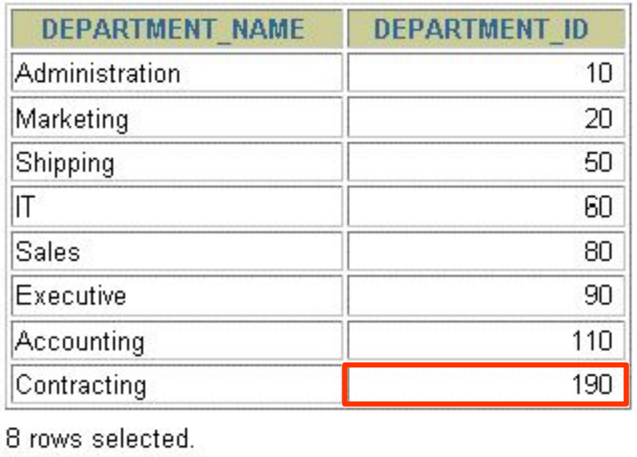
LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...

20 rows selected.

Outer Joins

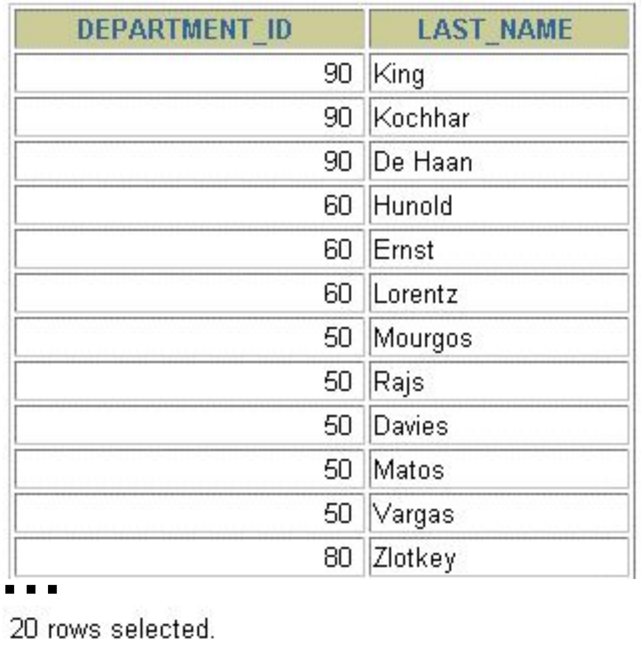
DEPARTMENTS



DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

EMPLOYEES



DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

…

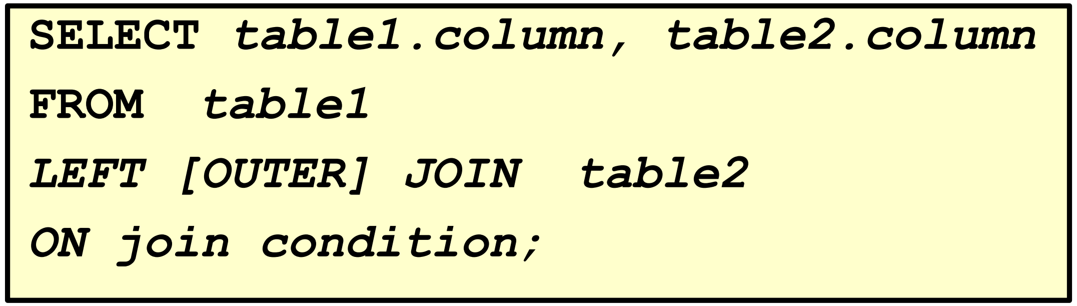
20 rows selected.

There are no employees in department 190.

Outer Joins Syntax

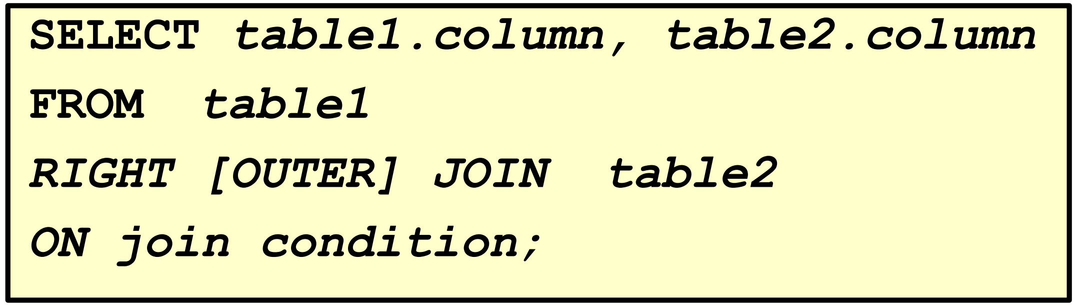
- You use an outer join to also see rows that do not meet the join condition.

- LEFT OUTER JOIN



```
SELECT table1.column, table2.column
FROM  table1
LEFT [OUTER] JOIN  table2
ON join condition;
```

- RIGHT OUTER JOIN

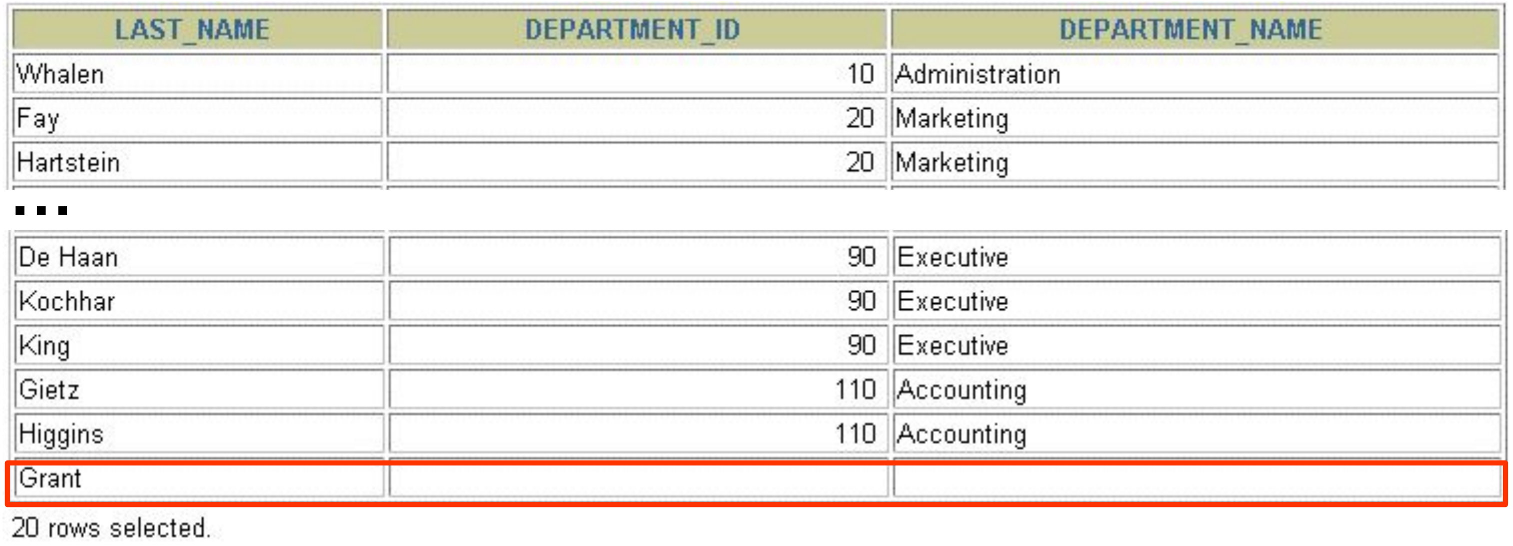


```
SELECT table1.column, table2.column
FROM  table1
RIGHT [OUTER] JOIN  table2
ON join condition;
```

Left Outer Join

```
SELECT e.last_name, e.department_id, d.department_name
FROM  Employee e
LEFT OUTER JOIN  Department d
ON d.department_id= e.department_id
```

- If there is no matching row for the right table in the ON part in a LEFT JOIN, a row with all columns set to NULL is used for the right table.



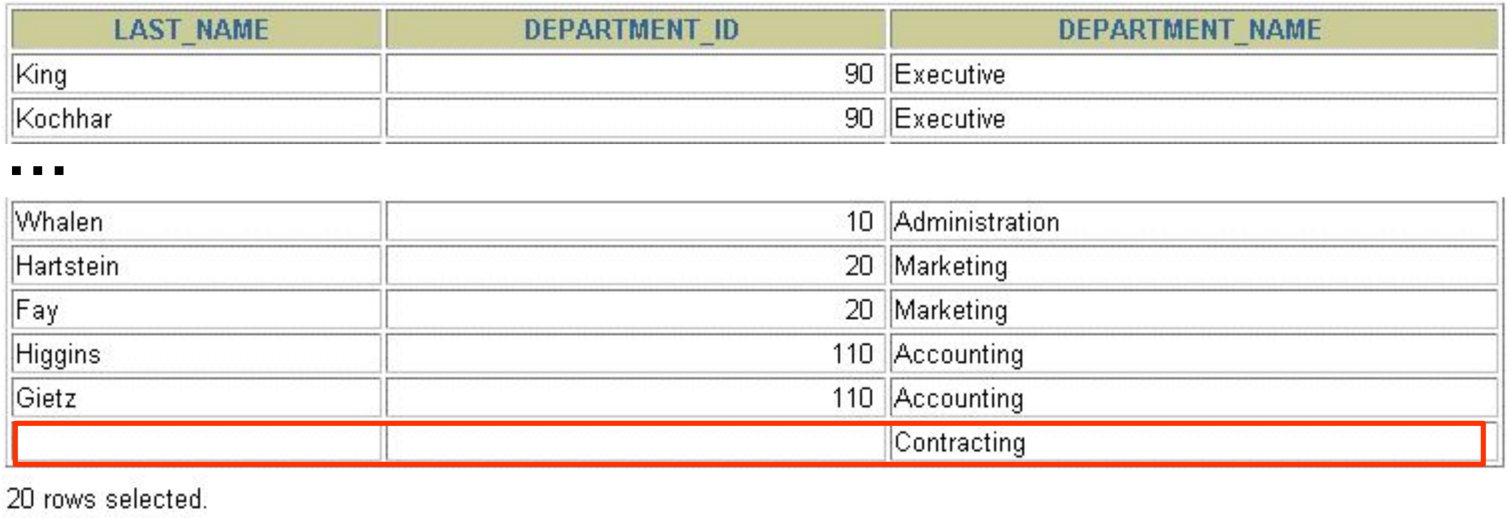
LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

Right Outer Join

```
SELECT e.last_name, e.department_id, d.department_name
FROM  Employee e
RIGHT JOIN  Department d
ON d.department_id= e.department_id
```

- If there is no matching row for the left table in the ON part in a RIGHT JOIN, a row with all columns set to NULL is used for the LEFT table.



LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive
...		
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Higgins	110	Accounting
Gietz	110	Accounting
		Contracting

20 rows selected.

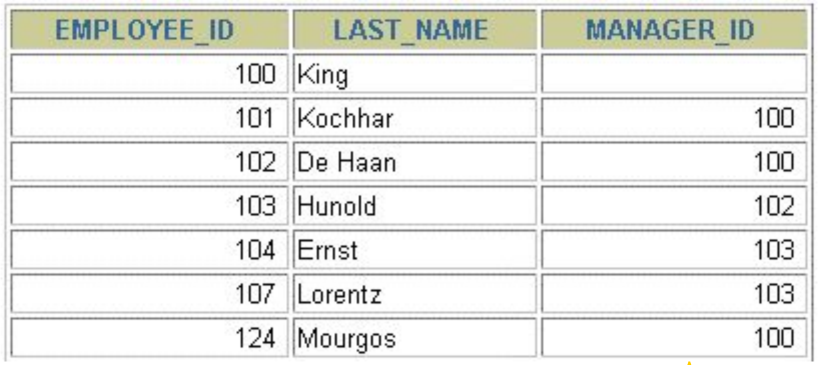
Full Outer Join

```
SELECT e.last_name, e.department_id, d.department_name
FROM  Employee e
FULL JOIN  Department d
ON d.department_id= e.department_id
```

- If there is no matching row for the left table in the ON part in a RIGHT JOIN, a row with all columns set to NULL is used for the LEFT table.
- If there is no matching row for the right table in the ON part in a LEFT JOIN, a row with all columns set to NULL is used for the right table.

Joining a Table to Itself Self Joins

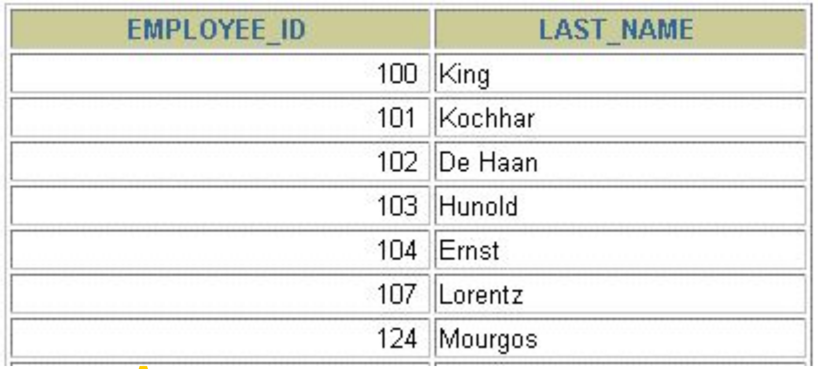
EMPLOYEES (WORKER)



EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

…

EMPLOYEES (MANAGER)



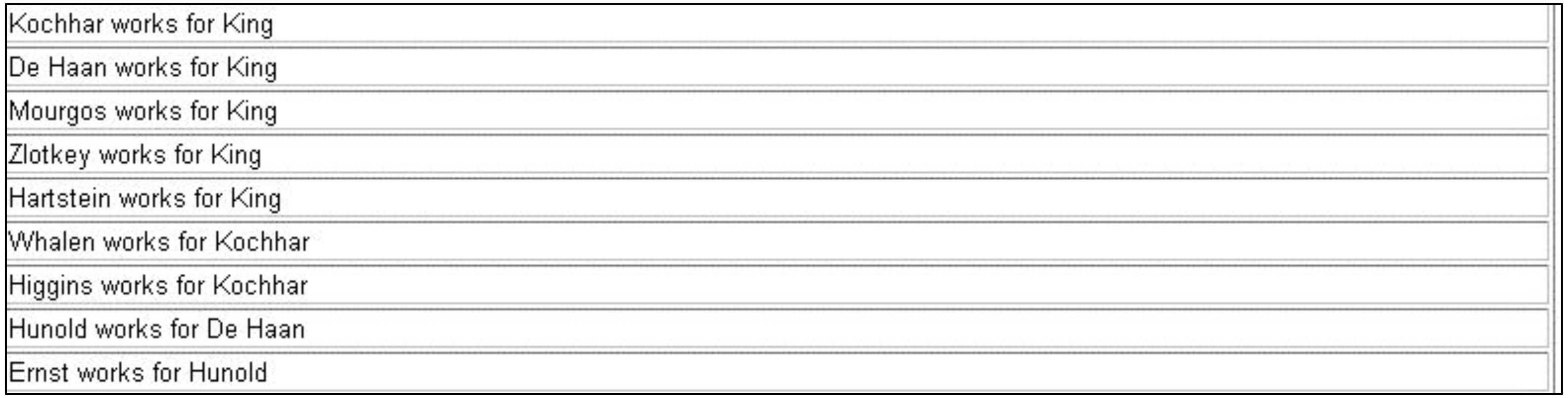
EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

…

MANAGER_ID in the WORKER table is equal to EMPLOYEE_ID in the MANAGER table.

Joining a Table to Itself Self Join

```
SELECT e.last_name as WORKER, m.last_name as MANAGER
FROM  Employee e
JOIN  Employee m
ON (e.manager_id = m.employee_id);
```

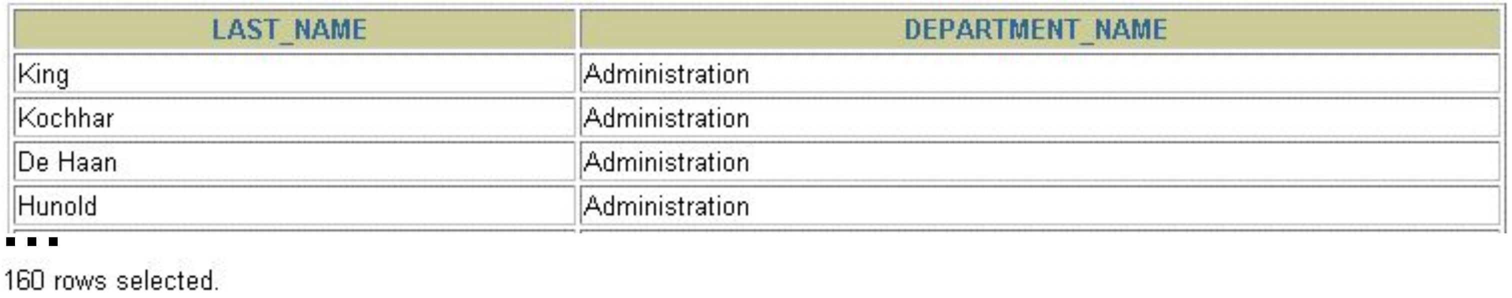


```
Kochhar works for King
De Haan works for King
Mourgos works for King
Zlotkey works for King
Hartstein works for King
Whalen works for Kochhar
Higgins works for Kochhar
Hunold works for De Haan
Ernst works for Hunold
```

Creating Cross Joins

- Cartesian product between the specified tables: If each and every row in the first table is joined to each and every row in the second table.
- The CROSS JOIN clause produces the Cartesian product of two tables.

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



LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

...

160 rows selected.

THANK YOU!