



JOIN | DISPLAYING DATA FROM MULTIPLE TABLES

Dr. Mohammed A. Mohammed

University of Sulaimani | College of science | Computer dept.

Objectives

- After completing this lesson, you should be able to do the following:
 - ✓ Write SELECT statements to access data from more than one table using equijoins and non-equijoins
 - ✓ Join a table to itself by using a self-join
 - ✓ View data that generally does not meet a join condition by using outer joins
 - ✓ Generate a Cartesian product of all rows from two or more tables

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



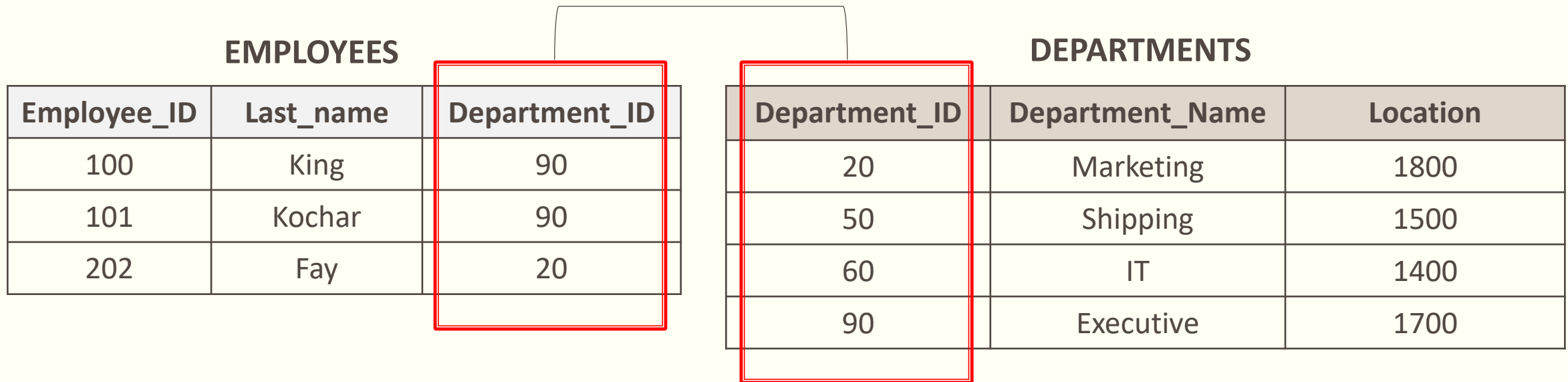
EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
102	90	Executive
205	110	Accounting
206	110	Accounting

Types of Joins

- ✓ Equijoin
- ✓ Cross JOIN
- ✓ Natural JOIN
- ✓ JOIN (Inner Join)
- ✓ Non-Equijoin
- ✓ Self JOIN
- ✓ Left | Right | Full Outer JOIN
- ✓ Set Operator

Equijoin

An **equijoin** is a join with a join condition containing an equality operator(=). An equijoin combines rows that have equivalent values for the specified columns



Example on Equijoin

- ✓ returns the last name and job_id of each employee and department_id and department_name of the department in which the employee works:

```
SELECT e.last_name, e.job_id, d.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id
ORDER BY e.last_name;
```

Cartesian Products

- ✓ combines each row of one table with each row of the other.
- ✓ A Cartesian product is formed when:
 - A join condition is omitted
 - A join condition is invalid
 - All rows in the first table are joined to all rows in the second table
- ✓ To avoid a Cartesian product, always include a valid join condition.
- ✓ Use (**Cross JOIN**)

Example on Cartesian Products

EMPLOYEES

Employee_ID	Last_name
100	King
101	Kochar
202	Fay

20 rows

DEPARTMENTS

Department_ID	Department_Name	Location
20	Marketing	1800
50	Shipping	1500
60	IT	1400

8 rows

•
•
•

Cartesian Product

Result:
20 x 8 = 160 rows

Employee_ID	Last_name	Department_ID	Department_Name	Location
100	King	20	Marketing	1800
100	King	50	Shipping	1500
100	King	60	IT	1400
101	Kochar	20	Marketing	1800
101	Kochar	50	Shipping	1500
101	Kochar	60	IT	1400
202	Fay	20	Marketing	1800
202	Fay	50	Shipping	1500
202	Fay	60	IT	1400

Example on Cartesian Products

```
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. . . .

Natural Joins

- ✓ The NATURAL JOIN clause is based on all columns in the two tables that have the same name.
- ✓ It selects rows from the two tables that have equal values in all matched columns.
- ✓ If the columns having the same names have different data types, an error is returned.

Example on Natural Joins

```
SELECT department_id, department_name, location_id,  
city FROM departments NATURAL JOIN locations ;
```

departments

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400

locations

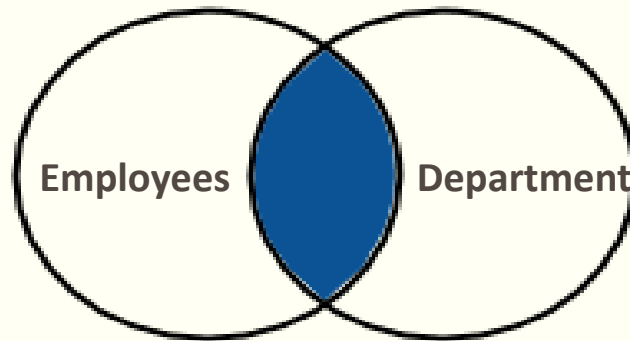
LOCATION_ID	CITY	COUNTRY_ID
1000	Roma	IT
1100	Venice	IT
1200	Tokyo	JP
1300	Hiroshima	JP
1400	Southlake	US
1500	South San Francisco	US

Output→

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
30	Purchasing	1700	Seattle

INNER JOIN || JOIN

An INNER JOIN combines data from two tables where there is a match on the joining column(s) in both tables.



EMPLOYEES

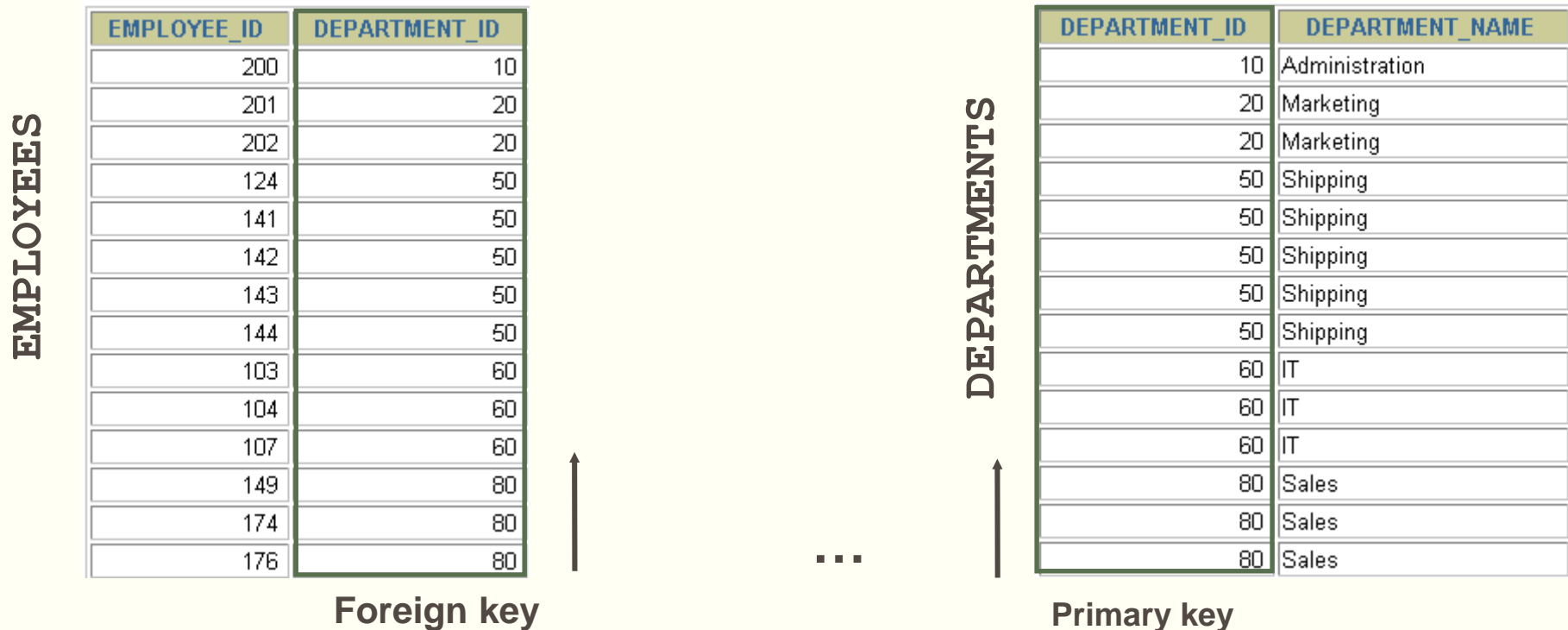
Employee_ID	Last_name	Department_ID
100	King	90
101	Kocher	90
202	Fay	20

DEPARTMENTS

Department_ID	Department_Name	Location
20	Marketing	1800
50	Shipping	1500
90	Executive	1700

Inner Join || Join with **USING**

- ✓ Use the **USING** clause to match only one column when more than one column matches.
- ✓ Do not use a table name or alias in the referenced columns.



Example JOIN with USING Clause

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

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

Using Table Aliases

- ✓ Use table aliases to simplify queries.
- ✓ Use table aliases to improve performance.

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

Inner Join || Join with ON

- ✓ The join condition for the natural join is basically an equijoin of all columns with the same name.
- ✓ Use the ON clause to specify arbitrary conditions or specify columns to join.
- ✓ The ON clause makes code easy to understand.

Example JOIN with ON Clause

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

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
124	Mourgos
141	Rajs
142	Davies
143	Matos

DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
10	10	1700
20	20	1800
20	20	1800
50	50	1500
50	50	1500
50	50	1500
50	50	1500

...

19 rows selected.

Applying Additional Conditions to a Join or Inner Join

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
AND    e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

Join Two or More Tables with ON Clause

```
SELECT e.employee_id, L.city, d.department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations L
ON     d.location_id = L.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

Non-Equijoins

- These operators are used in Non-Equijoin Query: \neq ,not in , between , $>$, $<$,,etc

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.

Dr. Mohammed Anwar

JOBS

JOB_ID	MIN_SALARY	MAX_SALARY
ST_CLERK	2008	5000
SH_CLERK	2500	5500
PU_CLERK	2500	5500
AD_ASST	3000	6000
IT_PROG	4000	10000
HR_REP	4000	9000
MK_REP	4000	9000
FI_ACCOUNT	4200	9000
AC_ACCOUNT	4200	9000
PR_REP	4500	10500
ST_MAN	5500	8500

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



Example on Non-Equijoins

```
SELECT e.last_name, e.salary, j.job_id
FROM   employees e JOIN jobs j
ON     e.salary BETWEEN ( j.min_salary AND j.max_salary );
```

LAST_NAME	SALARY	JOB_ID
Olson	2100	ST_CLERK
Markle	2200	ST_CLERK
Philtanker	2200	ST_CLERK
Landry	2400	ST_CLERK
Gee	2400	ST_CLERK

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 Employees table.**

Self-Joins Using the ON Clause

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     e.manager_id = m.employee_id;
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

...

19 rows selected.

LEFT | RIGHT | FULL 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

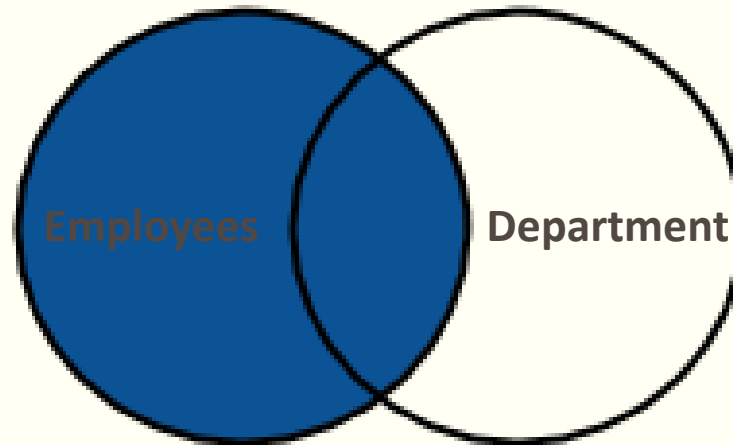
...

20 rows selected.

There are no employees in department 190.

LEFT [OUTER] JOIN

A LEFT JOIN statement returns all rows from the left table along with the rows from the right table for which the join condition is met.



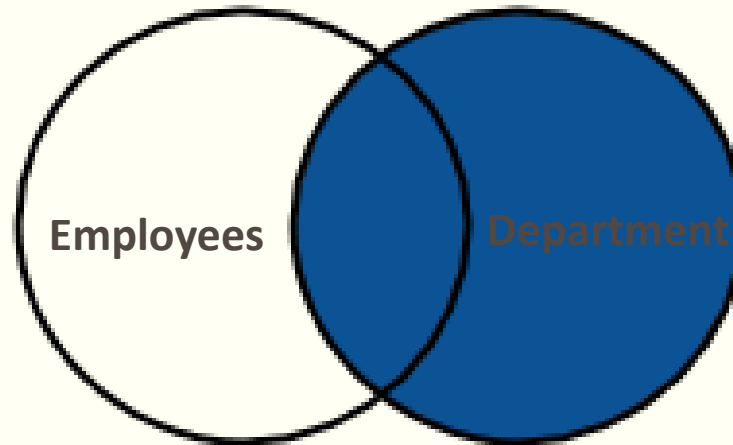
Example on **LEFT [OUTER] JOIN**

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

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	

RIGHT [OUTER] JOIN

A RIGHT JOIN statement returns all rows from the right table along with the rows from the left table for which the join condition is met.



Example on **RIGHT OUTER JOIN**

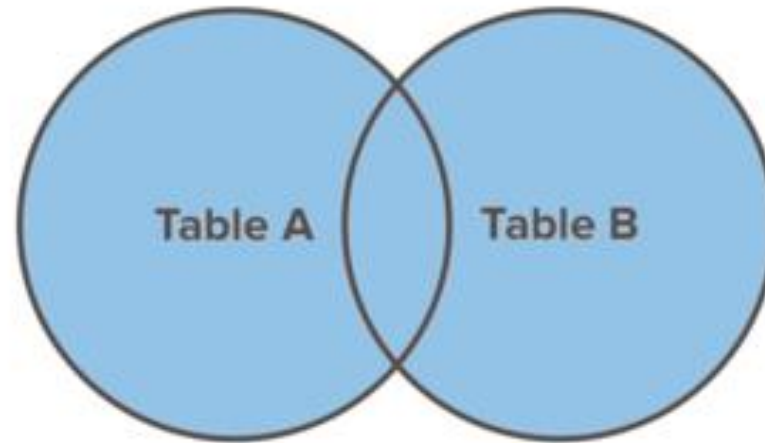
```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Davies	50	Shipping
...		
Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
	190	Contracting

20 rows selected.

FULL [OUTER] JOIN

A FULL JOIN returns all rows from the joined tables, whether they are matched or not i.e. you can say a full join combines the functions of a LEFT JOIN and a RIGHT JOIN.



SQL FULL JOIN

Example on FULL OUTER JOIN

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

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

Set Operators

- ✓ **Union** (U)
- ✓ **Intersect** (n)
- ✓ **Minus** (-)

Example:

$A = \{1, 2, 3, 4, 5\}$

$B = \{3, 2, 5, 7, 1\}$

$A \cup B = \{1, 2, 3, 4, 5, 7\}$

$A \cap B = \{1, 2, 3, 5\}$

$A - B = \{4\}$

UNION Operator

```
SELECT employee_id , last_name FROM employees
```

UNION

```
SELECT department_id , department_name FROM Departments;
```

EMPLOYEE_ID	LAST_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance
100	King
101	Kochhar
102	De Haan
103	Hunold

Intersect Operator

```
SELECT department_id FROM Departments  
Intersect  
SELECT department_id FROM Employees ;
```

DEPARTMENT_ID
10
20
30
40
50
60
70
80

Minus Operator

```
SELECT department_id FROM Departments  
Minus  
SELECT department_id FROM Employees ;
```

DEPARTMENT_ID
120
130
140
150
160
170
180

