Database Systems(2)

Tutorial 4

DR. ALYAA HAMZA

ENG. ESRAA SHEHAB

- **Set (or Aggregate) Functions:**

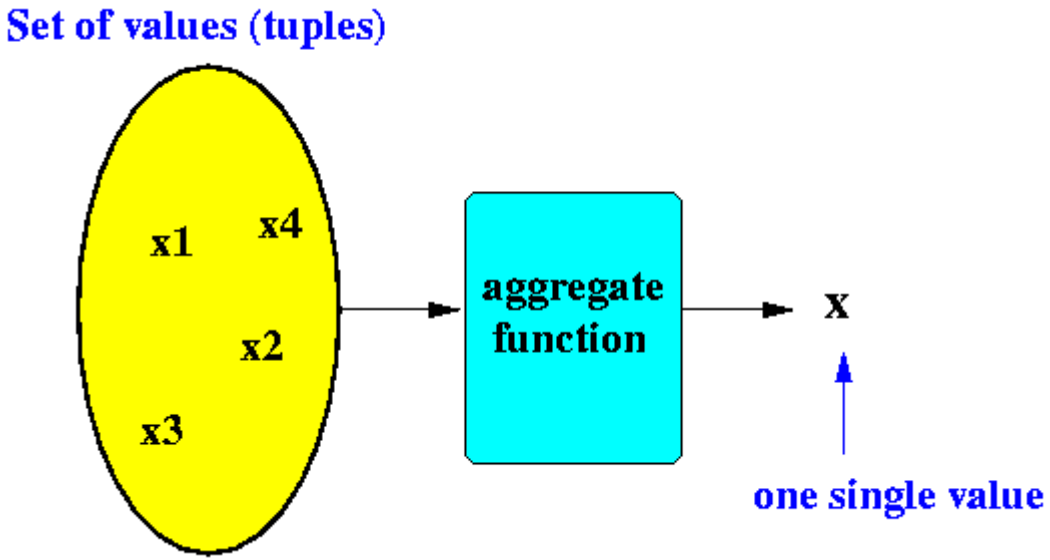
  Set functions operates on a set of values and computes one single output value.

  They are used to formulate "group conditions" - conditions on a set of tuples.

- **Aggregate or Set Functions:**

  Aggregate or Set functions are introduced to relational algebra to increase its expressive power.

  An aggregate function operates on a set of values (tuples) and computes one single value as output.

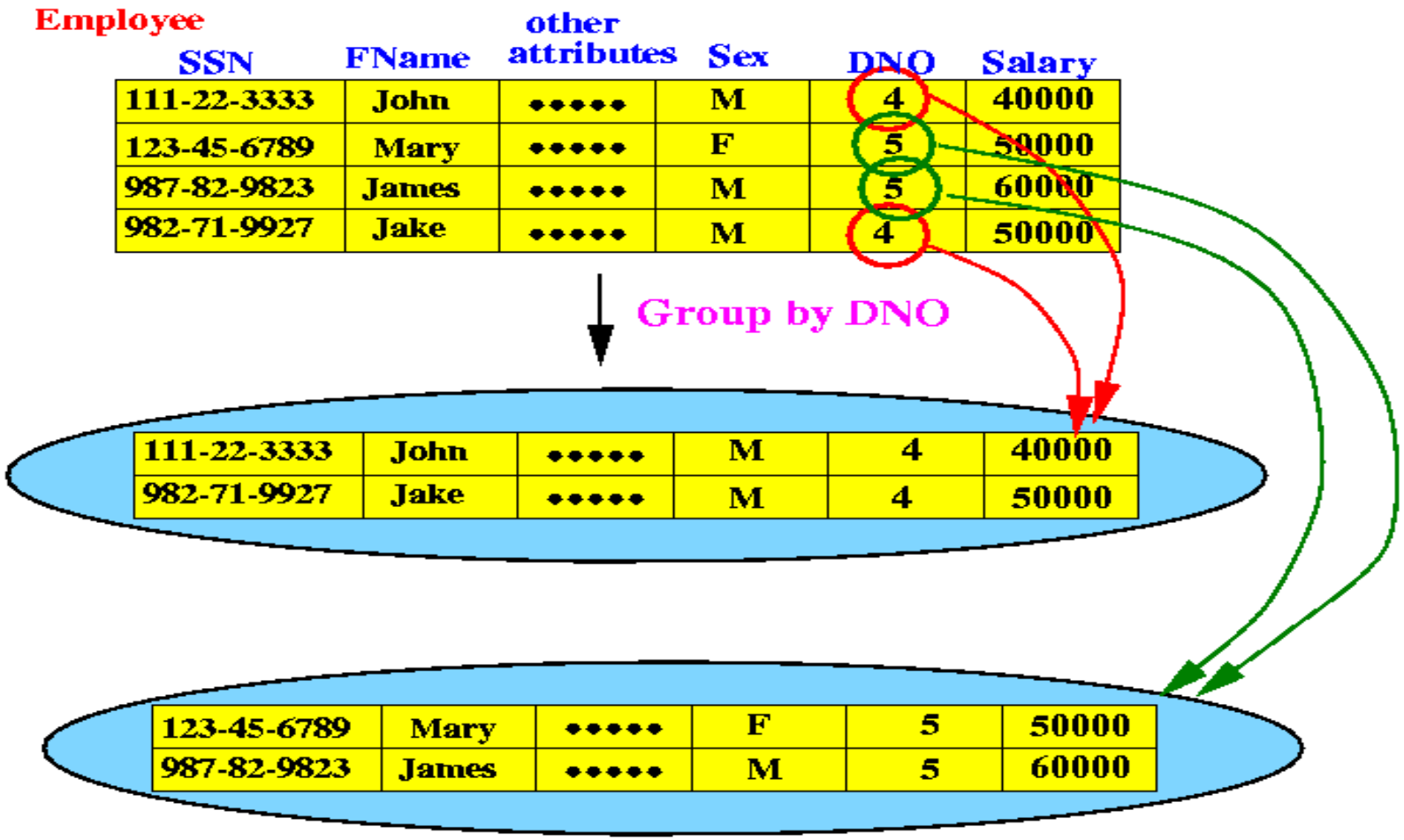


The Set Functions in Relational Algebra:

- **sum():** computes the sum of all values in the (numeric) set
- **avg():** computes the average of all values in the (numeric) set
- **max():** finds the maximum value of all values in the set
- **min():** finds the minimum value of all values in the set
- **any():** returns **TRUE** if set is not empty, otherwise (i.e., empty set), returns **FALSE**
- **count():** returns the cardinality (number of elements) in the set

Groups are formed based on **common values** in **one or more attributes**

Example: grouping **employee** tuples based on their ***dno*** **attribute:**


Employee
SSN
FName
other attributes
Sex
DNO
Salary
111-22-3333 John ••••• M 4 40000
123-45-6789 Mary ••••• F 5 50000
987-82-9823 James ••••• M 5 60000
982-71-9927 Jake ••••• M 4 50000
Group by DNO
111-22-3333 John ••••• M 4 40000
982-71-9927 Jake ••••• M 4 50000
123-45-6789 Mary ••••• F 5 50000
987-82-9823 James ••••• M 5 60000

Set functions used on groups formed on attribute values

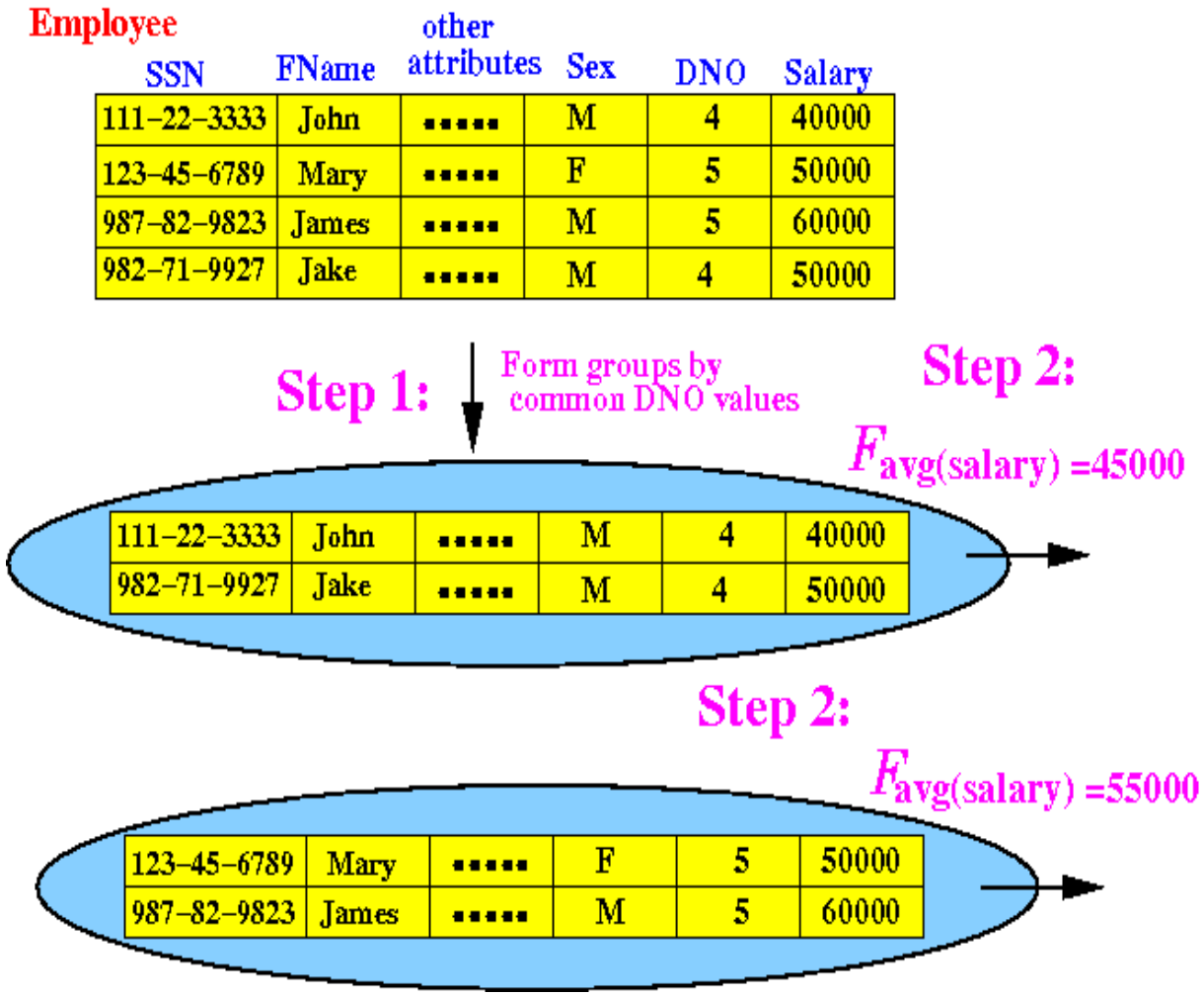
• We saw that **tuples** in a **relation** can be **grouped based on one or more attribute values**

• **One or more set functions** can now be **applied to these groups.**

Example:

Find the **average salary for each department**

We must first **form groups of employee tuples based on their DNO attribute** (grouped by department number) and then **compute the average Salary.**



The **Result SET** is

DNO	Avg
4	45000
5	55000

- **NOTE:**

1. The **result** is **always a RELATION** (i.e., a **set of tuples !!!**)

2. The **attributes** of the **result relation** consist of:

   - The **grouping attributes** !!!
   - The **set function**

   The **grouping attributes** are necessary, otherwise, you cannot tell much from the result, e.g.:

Avg
45000
55000

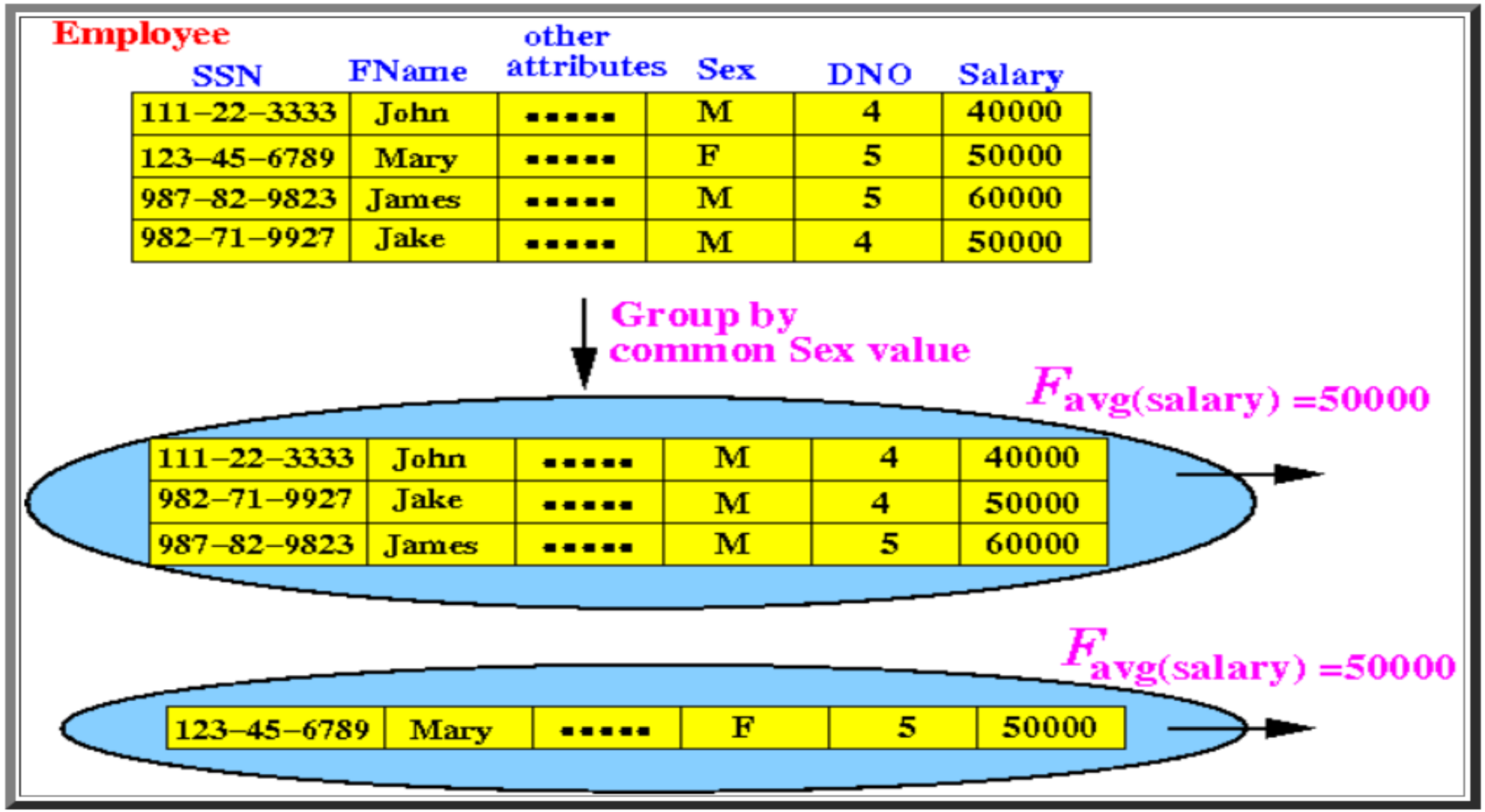
   You can't tell from this result what the average salary is of each department !

- **Example:**

  Compare (find) the **average salary for male and female employees**

We must first **form groups of employee tuples based on their "sex" attribute**

And then **compute the average Salary** :



- The **Result SET** is:

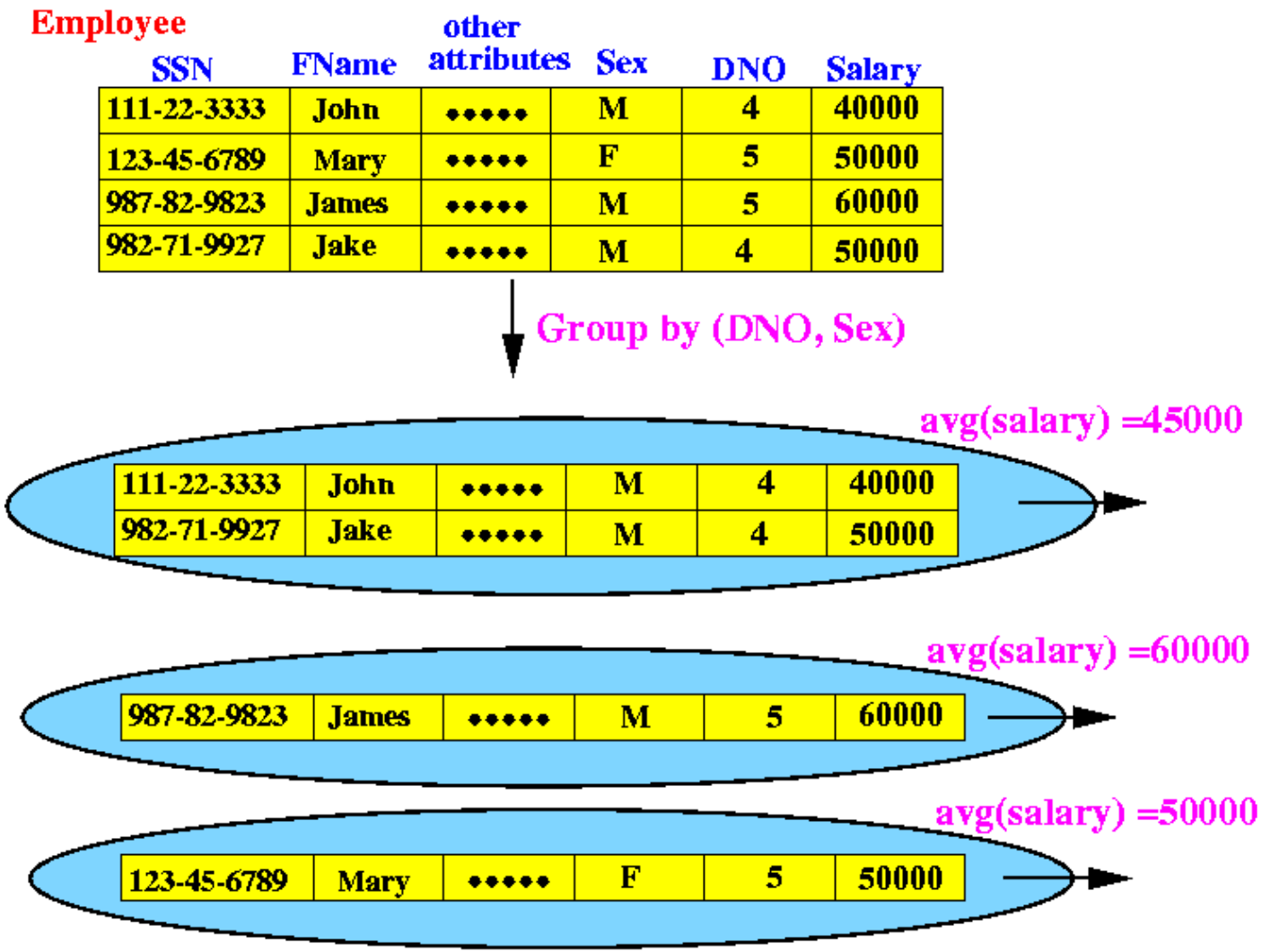
sex	Avg
M	50000
F	50000

Example:

Find the **average salary** of **males and females for each department.**

• We must first **form groups of employee tuples based on their DNO *and* sex attributes** (grouped by department number **and** sex).

• And then **compute the average Salary.**



- Group 1 is the male employees in department 4
- Group 2 is the male employees in department 5
- Group 3 is the female employees in department 5

  - The **Result SET** is:

DNO	sex	Avg
4	M	45000
5	M	60000
5	F	50000

- **Notice that:**

  1. the set/group of **female employees in department 4** is **empty**

  2. The **result set** is **NOT** equal to:

```
   DNO        sex        Avg
+--------+----------+------------+
|   4    |     M    |   45000    |
+--------+----------+------------+
|   4    |     F    |       0    |  <---- NOT the result !!!
+--------+----------+------------+
|   5    |     M    |   60000    |
+--------+----------+------------+
|   5    |     F    |   50000    |
+--------+----------+------------+
```

Keep this in mind when you do queries !!!

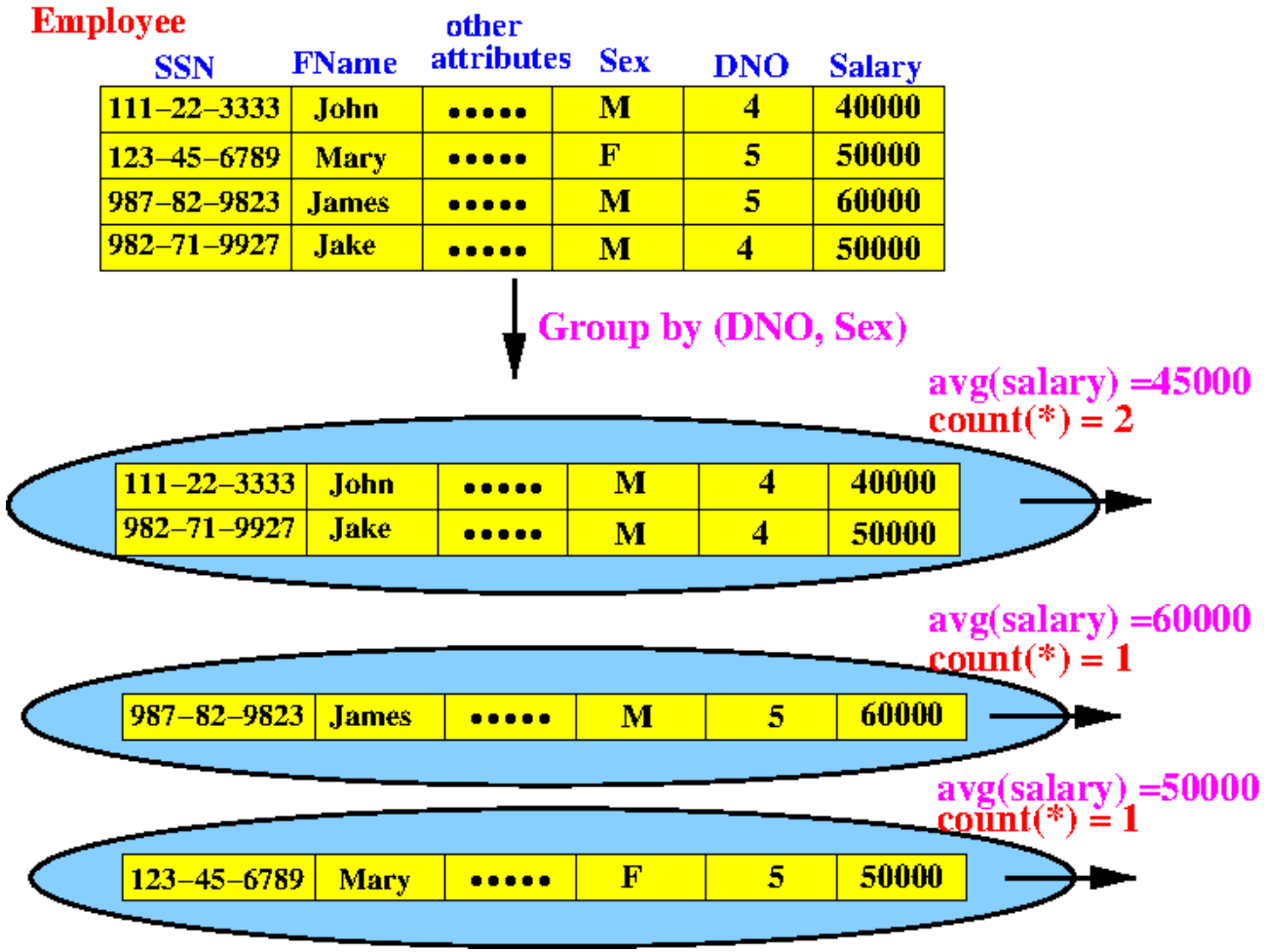
Applying *multiple* (set) functions

- You can apply **more than one (set) function** on the groups
- **Each set function** will produce **one value for each group**
- So, you will get a "vector" (or array) of function values

Example:

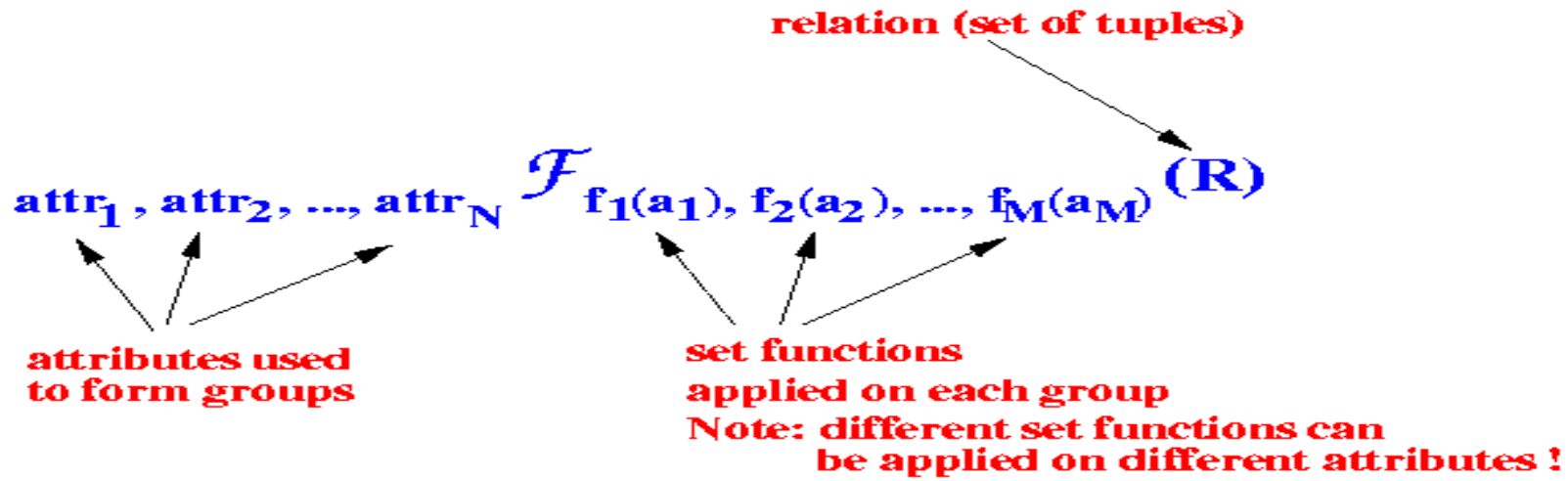
Find the **average salary *and* number of employees** for **males and females for each department.**

- We must first **form groups of employee tuples based on their DNO *and* sex attributes** (grouped by department number **and** sex)
- And then compute **average Salary *and* count the SSN of employees**.



- The **Result SET** is:

DNO	sex	Avg	Count
4	M	45000	2
5	M	60000	1
5	F	50000	1



- The **output** of the **set function** is a **relation** containing the following **attributes**:

- The **content** (i.e., **tuples**) of the **relation** consists of:
  - all **distinct values** of the **grouping attributes** $attr_1$, $attr_2$, ..., $attr_N$.
  - and the **function values** $f_1(a_1)$, $f_2(a_2)$, ..., $f_M(a_M)$ on the **corresponding group**.

Employee

SSN	FName	other attributes	Sex	DNO	Salary
111-22-3333	John	•••••	M	4	40000
123-45-6789	Mary	•••••	F	5	50000
987-82-9823	James	•••••	M	5	60000
982-71-9927	Jake	•••••	M	4	50000

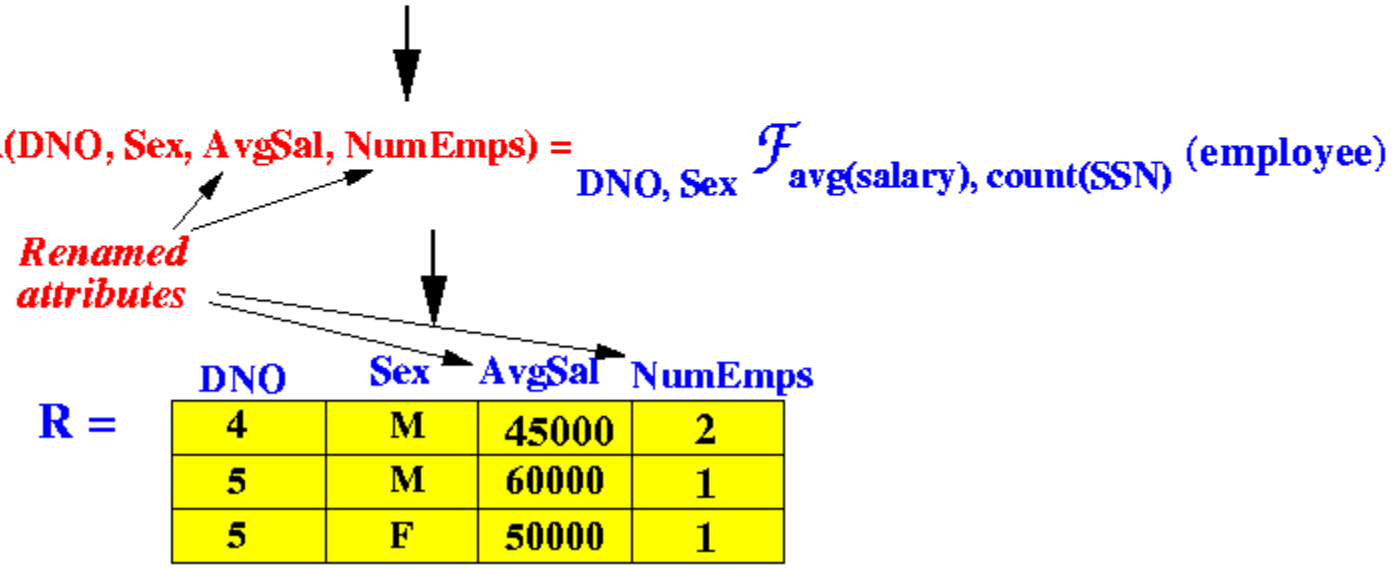
$$R = {}_{DNO,\ Sex}\mathcal{F}_{avg(salary),\ count(SSN)}\ (employee)$$

R =

DNO	Sex	avg	count
4	M	45000	2
5	M	60000	1
5	F	50000	1

Employee

SSN	FName	other attributes	Sex	DNO	Salary
111-22-3333	John	•••••	M	4	40000
123-45-6789	Mary	•••••	F	5	50000
987-82-9823	James	•••••	M	5	60000
982-71-9927	Jake	•••••	M	4	50000



Applying (set) functions *without any* grouping attributes

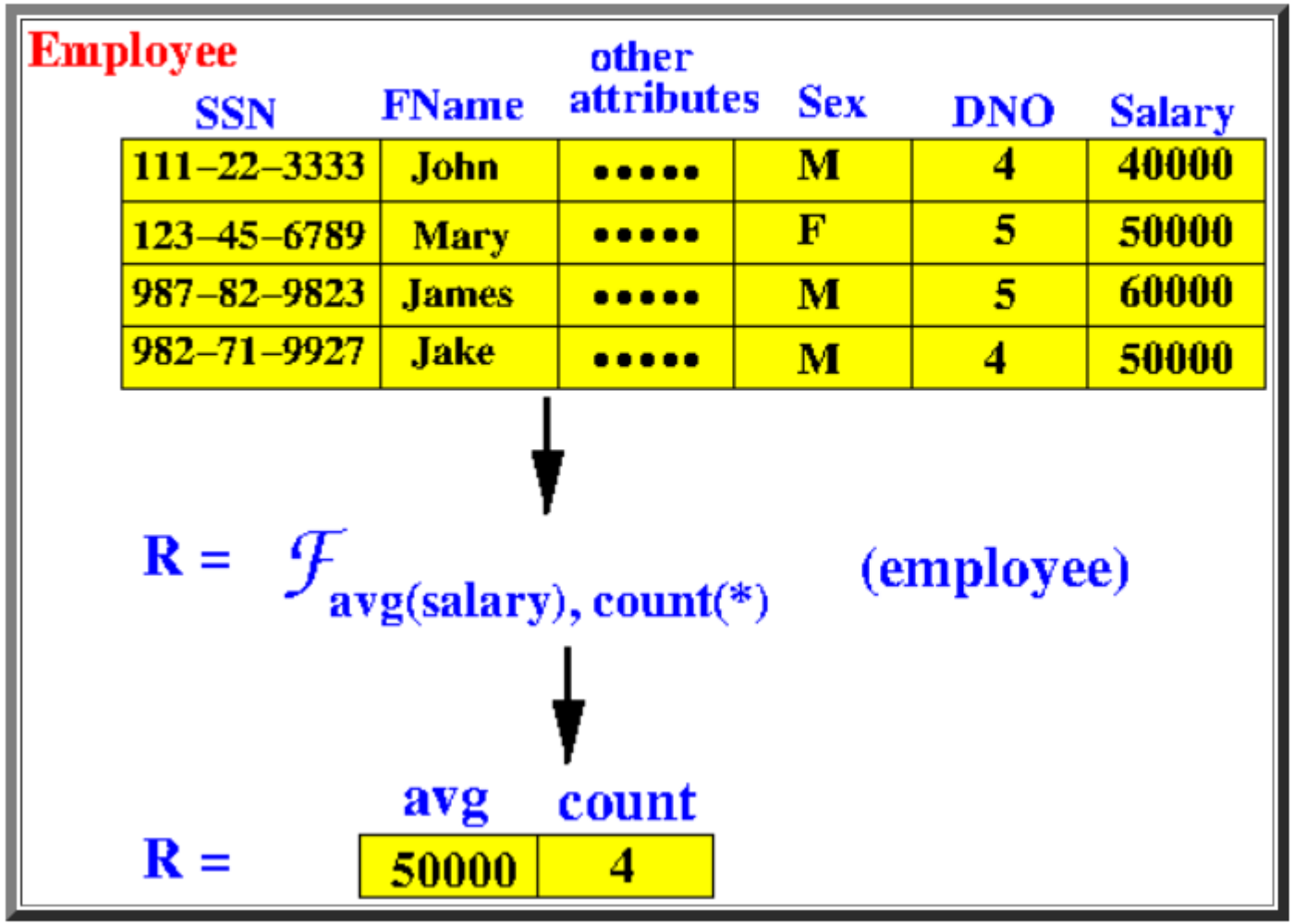
• When there are **no grouping attributes specified** in a **set function**, then the **set function** is **applied on ONE group that consists of *all tuples* of the relation** (i.e., all tuples in the relation are in one single group).

• **The most important fact to remember** in this case is:

> **Although** the **result set** will consist of **ONE single tuple (row)** , the **output of the set function** is ***still* a *relation***
>
> It is a **relation** with **one single tuple** in it !!!

- **Example:**

  determent the **average salary** of all employees in the company and the **total number of employees** in the company:



The SQL GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Example: GROUP BY With One Column

employee_number	last_name	first_name	salary	dept_id
1001	Smith	John	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Brad	71000	501
1004	Horvath	Jack	42000	501

Results:

dept_id	total_salaries
500	119500
501	113000

SQL statement:

```
SELECT dept_id, SUM(salary) AS total_salaries
FROM employees
GROUP BY dept_id;
```

Example - Using GROUP BY with the COUNT function

Let's look at how to use the GROUP BY clause with the COUNT function in SQL.

In this example, we have a table called *products* with the following data:

product_id	product_name	category_id
1	Pear	50
2	Banana	50
3	Orange	50
4	Apple	50
5	Bread	75
6	Sliced Ham	25
7	Kleenex	NULL

Enter the following SQL statement:

```
SELECT category_id, COUNT(*) AS total_products
FROM products
WHERE category_id IS NOT NULL
GROUP BY category_id
ORDER BY category_id;
```

There will be 3 records selected. These are the results that you should see:

category_id	total_products
25	1
50	4
75	1

In this example, we've used the COUNT function to calculate the number of products for each *category_id* and we've aliased the results of the COUNT function as *total_products*. We've excluded any *category_id* values that are NULL by filtering them out in the WHERE clause. Because the *category_id* is not encapsulated in the COUNT function, it must be listed in the GROUP BY clause.

Example - Using GROUP BY with the MIN function

Let's next look at how to use the GROUP BY clause with the MIN function in SQL.

In this example, we will use the *employees* table again that is populated the following data:

employee_number	last_name	first_name	salary	dept_id
1001	Smith	John	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Brad	71000	501
1004	Horvath	Jack	42000	501

Enter the following SQL statement:

```
SELECT dept_id, MIN(salary) AS lowest_salary
FROM employees
GROUP BY dept_id;
```

There will be 2 records selected. These are the results that you should see:

dept_id	lowest_salary
500	57500
501	42000

In this example, we've used the MIN function to return the lowest salary for each *dept_id* and we've aliased the results of the MIN function as *lowest_salary*. Because the *dept_id* is not encapsulated in the MIN function, it must be listed in the GROUP BY clause.

Example - Using GROUP BY with the MAX function

Finally, let's look at how to use the GROUP BY clause with the MAX function.

Let's use the *employees* table again, but this time find the highest salary for each *dept_id*:

employee_number	last_name	first_name	salary	dept_id
1001	Smith	John	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Brad	71000	501
1004	Horvath	Jack	42000	501

Enter the following SQL statement:

```
SELECT dept_id, MAX(salary) AS highest_salary
FROM employees
GROUP BY dept_id;
```

There will be 2 records selected. These are the results that you should see:

dept_id	highest_salary
500	62000
501	71000

In this example, we've used the MAX function to return the highest salary for each *dept_id* and we've aliased the results of the MAX function as *highest_salary*. The *dept_id* column must be listed in the GROUP BY clause because it is not encapsulated in the MAX function.

A. What is Stored Procedure:

- A stored procedure is a prepared SQL code that you can save, so the code can be reused repeatedly.
- So, if you have an SQL query that you write repeatedly, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

➢ Stored Procedure Syntax:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

➢ Execute a Stored Procedure:

```
EXEC procedure_name;
```

B. What is View:

- In SQL, a view is a virtual table based on the result-set of an SQL statement.

- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

➢ VIEW Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2,
FROM table_name
WHERE condition;
```

B. What is Database Diagram:

- Database diagrams are visual representations of underlying table structure and their relationships.
- Generally referred to as ER (Entity-Relationship) diagram.
- With database diagrams, we can easily identify the list of tables in a database and their association through the type of relationships.

C# Database Connection:

In the C# SQL connection tutorial, you will learn:

- Fundamentals of Database connectivity
- How to connect C# to Database
- Access data with the SqlDataReader
- C# Insert Into Database
- Updating Records
- Deleting Records
- Connecting Controls to Data
- C# DataGridView