

**Créer un état avec des données agrégées  
à l'aide des fonctions de groupe**

# Fonctions de groupe : Présentation

Les fonctions de groupe opèrent sur des ensembles de lignes et fournissent un résultat par ensemble.

## EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
...		
18	80	11000
19	80	8600
20	(null)	7000

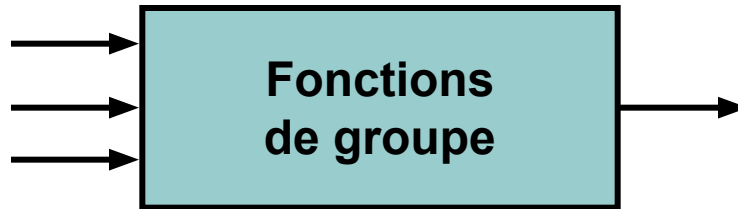
**Salaire maximum  
dans la table**

**EMPLOYEES**

MAX(SALARY)  
24000

# Types de fonction de groupe

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



# Fonctions de groupe : Syntaxe

```
SELECT      group_function(column), ...  
FROM        table  
[WHERE      condition]  
[ORDER BY   column];
```

# Utiliser les fonctions AVG et SUM

Vous pouvez utiliser les fonctions AVG et SUM pour les données numériques.

```
SELECT AVG(salary) , MAX(salary) ,  
       MIN(salary) , SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

# Utiliser les fonctions MIN et MAX

Vous pouvez utiliser les fonctions `MIN` et `MAX` pour les données de type nombre, caractère et date.

```
SELECT MIN(hire date), MAX(hire date)  
FROM   employees;
```

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	17-JUN-87	29-JAN-00

# Utiliser la fonction COUNT

COUNT (\*) renvoie le nombre de lignes d'une table :

1

```
SELECT COUNT(*)  
FROM employees  
WHERE department_id = 50;
```

	COUNT(*)
1	5

COUNT(*expr*) renvoie le nombre de lignes comportant des valeurs non NULL pour *expr* :

2

```
SELECT COUNT(commission_pct)  
FROM employees  
WHERE department_id = 80;
```

	COUNT(COMMISSION_PCT)
1	3

# Utiliser le mot-clé DISTINCT

- `COUNT (DISTINCT expr)` renvoie le nombre de valeurs non NULL uniques de *expr*.
- Pour afficher le nombre de valeurs de département distinctes dans la table `EMPLOYEES` :

```
SELECT COUNT(DISTINCT department id)  
FROM employees;
```

	COUNT(DISTINCTDEPARTMENT_ID)
1	7



# Fonctions de groupe et valeurs NULL

Les fonctions de groupe ignorent les valeurs NULL de la colonne considérée :

1

```
SELECT AVG(commission_pct)
FROM employees;
```

	AVG(COMMISSION_PCT)
1	0.2125

La fonction NVL force les fonctions de groupe à inclure les valeurs NULL :



2

```
SELECT AVG(NVL(commission_pct, 0))
FROM employees;
```

	AVG(NVL(COMMISSION_PCT,0))
1	0.0425

# Créer des groupes de données

## EMPLOYEES

	DEPARTMENT_ID		SALARY
1	10		4400
2	20		13000
3	20		6000
4	50		2500
5	50		2600
6	50		3100
7	50		3500
8	50		5800
9	60		9000
10	60		6000
11	60		4200
12	80		11000
13	80		8600
...			
18	110		8300
19	110		12000
20	(null)		7000

4400



9500

3500

6400

10033

**Salaire moyen dans la  
table EMPLOYEES pour  
chaque département**

	DEPARTMENT_ID		AVG(SALARY)
1	(null)		7000
2	20		9500
3	90		19333.333333333333...
4	110		10150
5	50		3500
6	80		10033.333333333333...
7	10		4400
8	60		6400

# Créer des groupes de données :

## Syntaxe de la clause GROUP BY

Vous pouvez diviser une table en groupes de plus petite taille à l'aide de la clause GROUP BY.

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

# Utiliser la clause GROUP BY

Toutes les colonnes de la liste `SELECT` qui ne figurent pas dans des fonctions de groupe doivent être présentes dans la clause `GROUP BY`.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id ;
```

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.333333333333...
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	10	4400
8	60	6400

# Utiliser la clause GROUP BY

La colonne GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

```
SELECT    AVG(salary)
FROM      employees
GROUP BY  department_id ;
```

	AVG(SALARY)
1	7000
2	9500
3	19333.333333333333333333...
4	10150
5	3500
6	10033.333333333333333333...
7	4400
8	6400

# Procéder à un regroupement sur la base de plusieurs colonnes

## EMPLOYEES

	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	2500
5	50	ST_CLERK	2600
6	50	ST_CLERK	3100
7	50	ST_CLERK	3500
8	50	ST_MAN	5800
9	60	IT_PROG	9000
10	60	IT_PROG	6000
11	60	IT_PROG	4200
12	80	SA_REP	11000
13	80	SA_REP	8600
14	80	SA_MAN	10500

...

19	110	AC_MGR	12000
20	(null)	SA_REP	7000

Ajoutez les salaires de la table **EMPLOYEES** pour tous les postes, regroupés par département.

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	110	AC_ACCOUNT	8300
2	110	AC_MGR	12000
3	10	AD_ASST	4400
4	90	AD_PRES	24000
5	90	AD_VP	34000
6	60	IT_PROG	19200
7	20	MK_MAN	13000
8	20	MK_REP	6000
9	80	SA_MAN	10500
10	80	SA_REP	19600
11	(null)	SA_REP	7000
12	50	ST_CLERK	11700
13	50	ST_MAN	5800

# Utiliser la clause GROUP BY sur plusieurs colonnes

```
SELECT    department_id, job_id, SUM(salary)
FROM      employees
WHERE     department_id > 40
GROUP BY  department_id, job_id
ORDER BY  department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	50	ST_CLERK	11700
2	50	ST_MAN	5800
3	60	IT_PROG	19200
4	80	SA_MAN	10500
5	80	SA_REP	19600
6	90	AD_PRES	24000
7	90	AD_VP	34000
8	110	AC_ACCOUNT	8300
9	110	AC_MGR	12000

# Interrogations non autorisées avec les fonctions de groupe

Toute colonne ou expression de la liste `SELECT` qui n'est pas une fonction d'agrégation doit figurer dans la clause `GROUP BY` :

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

ORA-00937: not a single-group group function  
00937. 00000 - "not a single-group group function"

**Il est nécessaire d'ajouter une clause `GROUP BY` pour compter les noms associés à chaque `department_id`.**

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id;
```

ORA-00979: not a GROUP BY expression  
00979. 00000 - "not a GROUP BY expression"

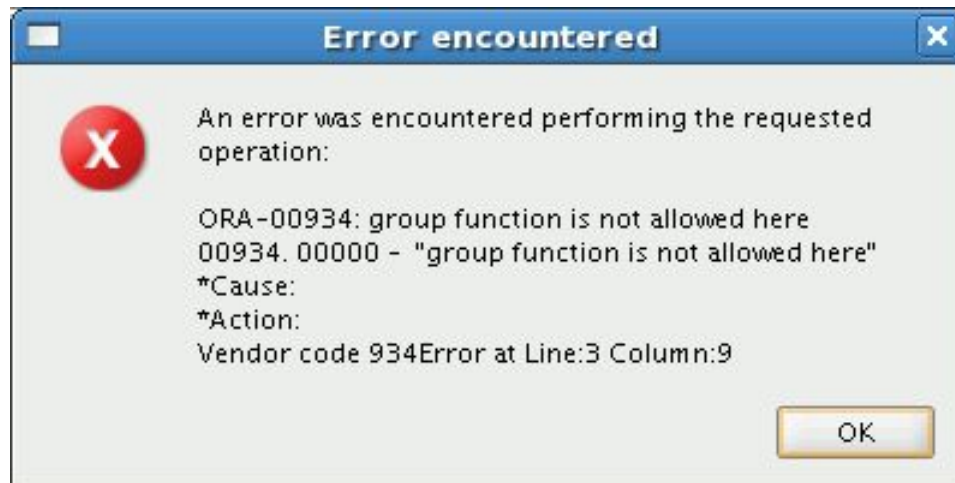
**Ajoutez `job_id` dans la clause `GROUP BY` ou supprimez la colonne `job_id` de la liste `SELECT`.**



# Interrogations non autorisées avec les fonctions de groupe

- Vous ne pouvez pas utiliser la clause `WHERE` pour restreindre des groupes.
- Pour cela, vous pouvez utiliser la clause `HAVING`.
- Vous ne pouvez pas utiliser de fonctions de groupe dans la clause `WHERE`.

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY  department_id;
```



**Vous ne pouvez pas  
utiliser la clause `WHERE`  
pour restreindre  
des groupes.**

# Restreindre les résultats d'un groupe

## EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

**Salaire maximum par département lorsqu'il est supérieur à 10 000 \$**

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

# Restreindre les résultats d'un groupe avec la clause HAVING

Lorsque vous utilisez la clause `HAVING`, le serveur Oracle restreint les groupes de la manière suivante :

1. Les lignes sont regroupées.
2. La fonction de groupe est appliquée.
3. Les groupes correspondant à la clause `HAVING` sont affichés.

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[HAVING    group condition]
[ORDER BY  column];
```



# Utiliser la clause HAVING

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary)>10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

# Utiliser la clause HAVING

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY  job_id
HAVING    SUM(salary) > 13000
ORDER BY  SUM(salary);
```

	 JOB_ID	 PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

# Imbriquer des fonctions de groupe

Affichez le salaire moyen maximum :

```
SELECT MAX (AVG (salary))
FROM employees
GROUP BY department id;
```

[illegible]