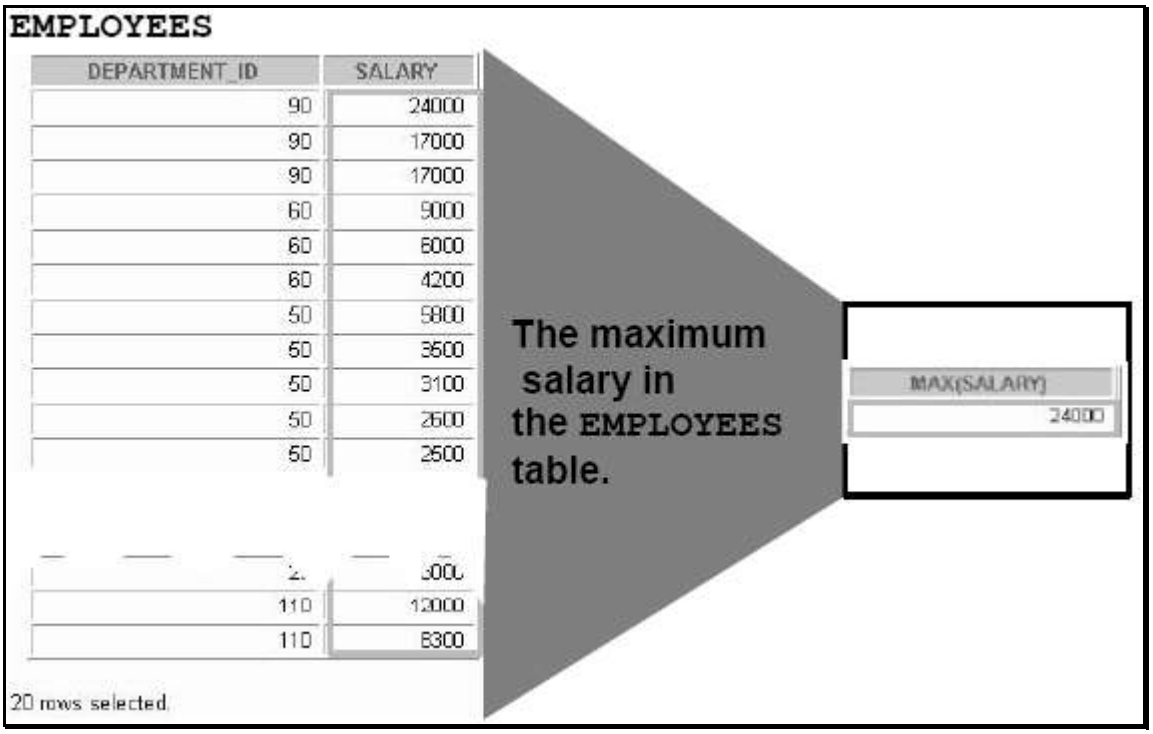
:Group Functions

تعمل هذه التوابع على مجموعات من الصفوف وتعيد قيمة وحيدة لكل مجموعة.



وهذه التوابع هي:

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

Function	Description
AVG([DISTINCT|ALL]*n*)	Average value of *n*, ignoring null values
COUNT({*|[DISTINCT|ALL]*expr*})	Number of rows, where *expr* evaluates to something other than null (count all selected rows using *, including duplicates and rows with nulls)
MAX([DISTINCT|ALL]*expr*)	Maximum value of *expr*, ignoring null values
MIN([DISTINCT|ALL]*expr*)	Minimum value of *expr*, ignoring null values
STDDEV([DISTINCT|ALL]*x*)	Standard deviation of *n*, ignoring null values
SUM([DISTINCT|ALL]*n*)	Sum values of *n*, ignoring null values
VARIANCE([DISTINCT|ALL]*x*)	Variance of *n*, ignoring null values

ملاحظات:

- استخدام DISTINCT يجعل التابع يأخذ بعين الاعتبار القيم غير المكررة فقط، بينما عند استخدام ALL (وهي الحالة الافتراضية ولا داعي لذكرها) فسيتم الأخذ بعين الاعتبار كل قيمة موجودة.
- الوسيط expr يمكن أن يكون من النوع Char, Varchar2, Number, Date.
- جميع التوابع السابقة تتجاهل القيم غير المعرفة Null Values.
- التوابع AVG, SUM, VARIANCE , STDDEV تستخدم فقط مع البيانات الرقمية.

مثال:

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

يمكن استخدام التابعين MIN, MAX مع أي نوع من أنواع البيانات...

مثال :

```
SELECT MIN(hire_date), MAX(hire_date)
FROM   employees;
```

```
SELECT  MIN(last_name) , MAX(last_name)
FROM     employees;
```

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

يعيد التابع COUNT (*) عدد الأسطر في الجدول، بما في ذلك القيم المكررة وقيم NULL.

أما التابع COUNT (expr) فهو يعيد عدد الأسطر متجاهلاً قيم NULL.

والتابع COUNT (DISTINCT expr) يعيد عدد القيم غير المكررة متجاهلاً قيم NULL.

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

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

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

كما ذكرنا سابقاً، فإن جميع التوابع السابقة الذكر تتجاهل القيم غير المعرفة NULL

لاحظ الفرق بين المثالين التاليين:

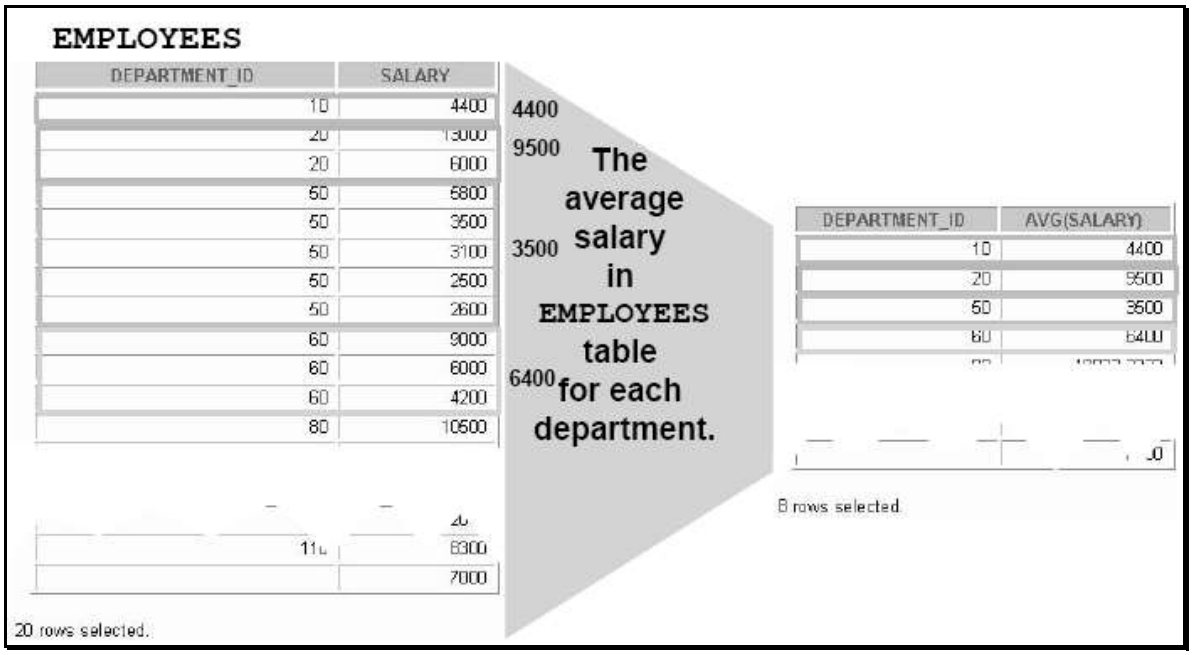
```
SELECT AVG(commission_pct)
FROM   employees;
```

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

التابع NVL يحول قيم NULL إلى قيم معرفة (وهي هنا ٠) وبالتالي سيتم أخذها بعين الاعتبار عند حساب المتوسط الحسابي.

م. رفاه مختار

إنشاء مجموعات من البيانات:



يمكن تقسيم الجدول إلى مجموعات صغيرة، وتطبيق التوابع السابقة الذكر على تلك المجموعات، وذلك باستخدام عبارة GROUP BY التي تحدد كيف سيتم تجميع الصفوف. الشكل العام لاستخدام تلك العبارة:

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

ملاحظات:

١. عند تضمين عبارة SELECT أي من التوابع السابقة الذكر، فإن أي عمود آخر سيتم ذكره في عبارة SELECT يجب أن يذكر في عبارة GROUP BY (أي سيتم تجميع الصفوف حسب هذه الأعمدة) ، وإلا سنحصل على رسالة خطأ.

٢. استخدام عبارة WHERE تساعد على استبعاد الأسطر التي لا تحقق الشرط قبل تقسيم الجدول إلى مجموعات.

٣. لا يمكن استخدام أسماء مستعارة للأعمدة في عبارة GROUP BY.

مثال: نريد حساب المتوسط الحسابي لرواتب كل قسم.

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

حاول تطبيق المثال السابق بدون كتابة عبارة GROUP BY.

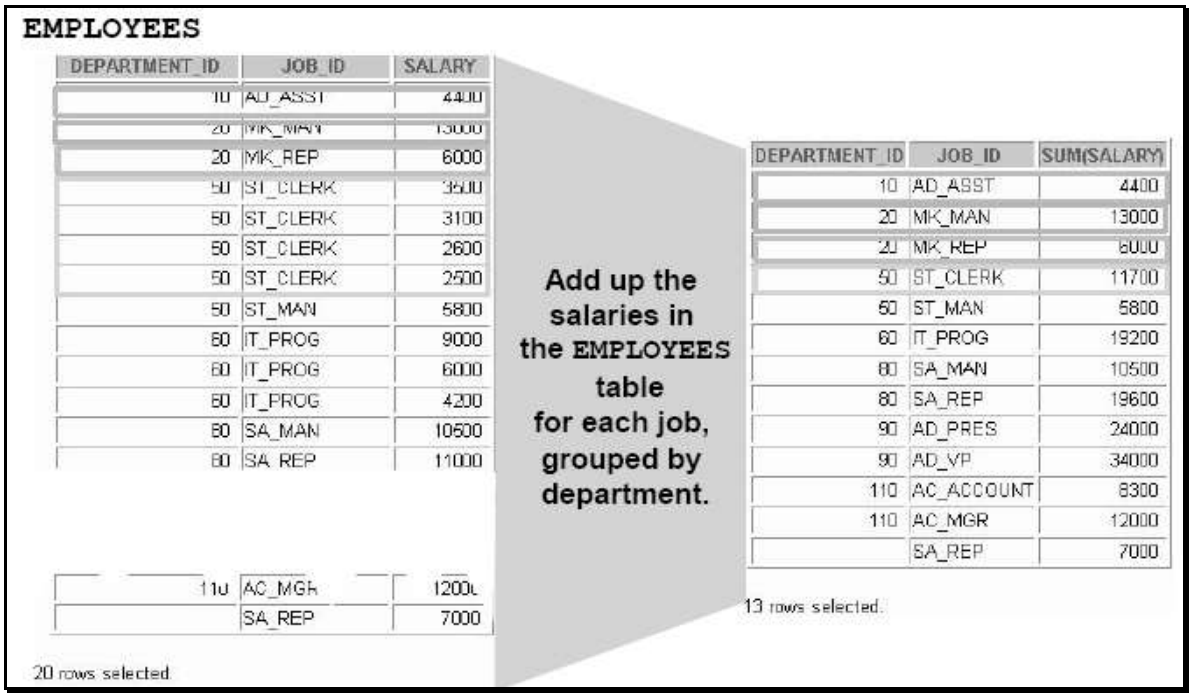
يجب الإشارة إلى أنه ليس من الضرورة أن الأعمدة المذكورة في عبارة GROUP BY ستكون موجودة في عبارة SELECT.

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

يمكن استخدام التوابع السابقة الذكر في عبارة ORDER BY

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

التجميع حسب أكثر من عمود:



يبين الشكل السابق مجموع الرواتب الذي يدفع لكل عمل ضمن كل قسم، ويكون الاستعلام الموافق:

```
SELECT   department_id dept_id, job_id, SUM(salary)
FROM     employees
GROUP BY department_id, job_id;
```

وهنا سيتم تطبيق تابع الجمع على حقل الراتب لكل عمل ضمن كل قسم.

لا يمكن استخدام التوابع السابقة الذكر في عبارة WHERE ، المثال التالي سيعطي رسالة خطأ عند تنفيذه:

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

وإنما عند الحاجة لتطبيق شرط معين على توابع التجميع فإننا نستخدم عبارة جديدة هي HAVING، نضع فيها الشروط المراد تطبيقها على توابع التجميع.

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

استخدام توابع التجميع المتداخلة:

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

الاستعلامات الفرعية: (SubQueries)

بفرض أننا نريد كتابة استعلام لمعرفة الأشخاص الذين رواتبهم أكبر من راتب موظف معين، فإننا نحتاج إلى استعلامين الأول يعيد لنا راتب هذا الموظف، والثاني يعيد لنا أسماء الأشخاص الذين يتقاضون راتباً أكبر من الراتب المعاد من الاستعلام الأول.

الاستعلام الأول يدعى *subquery* ويعيد قيمة سيتم استخدامها من قبل الاستعلام الثاني الذي هو الاستعلام الرئيسي *main query* .

الشكل العام للاستعلام الفرعي *Subquery:*

```
SELECT    select_list
FROM      table
WHERE     expr operator
                    (SELECT       select_list
                    FROM          table);
```

المقصود بـOperator في الشكل أي عامل من عوامل المقارنة مثل IN , = , < , >.....

يتم تنفيذ الاستعلام الفرعي (الداخلي) *subquery* أولاً، ثم يتم استخدام القيمة الناتجة في تنفيذ الاستعلام الخارجي أو الرئيسي *main query.*

يُعرّف الاستعلام الفرعي *subquery بأنه عبارة SELECT* موجودة ضمن عبارة *SELECT أخرى،* وهو مفيد عند الحاجة لعرض بيانات تعتمد على بيانات أخرى.

يمكن استخدام الاستعلام الفرعي في:

١. عبارة WHERE
٢. عبارة HAVING
٣. عبارة FROM

مثال:

```
SELECT last_name
FROM   employees
WHERE  salary >
                (SELECT salary
                 FROM employees
                 WHERE  last_name = 'Abel');
```

11000

ملاحظات:

١. يجب إحاطة الاستعلام الفرعي بأقواس هلالية ( ).
٢. يفضل وضع الاستعلام الفرعي على الطرف الأيمن من شرط المقارنة.
٣. يوجد نوعان من معاملات المقارنة يمكن استخدامها مع الاستعلامات الفرعية، النوع الأول يدعى single-row operators والثاني: multiple-row operators.

لدينا نوعان من الاستعلامات الفرعية:

١. Single-row subquery: هذا النوع يعيد صف وحيد فقط.
٢. Multiple-row subquery: هذا النوع يعيد أكثر من سطر.

Single-Row Subqueries: يعيد هذا النوع من الاستعلامات الفرعية صف واحد فقط، ويستخدم مع عوامل المقارنة التالية:

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

مثال:

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141);
```

مثال:

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =                    ST_CLERK
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >                     2600
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

ويمكن استخدام توابع التجميع في الاستعلامات الفرعية:

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

2500

```
SELECT   department_id, MIN(salary)
FROM     employees
GROUP BY department_id
HAVING   MIN(salary) >
                        (SELECT MIN(salary)
                         FROM employees
                         WHERE department_id = 50);
```

2500

Multiple-Row Subqueries: يعيد هذا النوع أكثر من صف، ويستخدم مع عوامل المقارنة التالية:

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

ملاحظة: يمكن استخدام معامل النفي NOT مع المعاملات السابقة.

مثال:

```
SELECT last_name, salary, department_id
FROM   employees
WHERE  salary IN (SELECT   MIN(salary)
                  FROM     employees
                  GROUP BY department_id);
```

```
SELECT employee_id, last_name, job_id, salary
FROM   employees     9000, 6000, 4200
WHERE  salary < ANY
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

ملاحظات:

< ANY : means less than the maximum.
> ANY : means more than the minimum.
< ALL : means less than the minimum.
> ALL : means more than the maximum.

م. رفاه مختار

مثال:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```