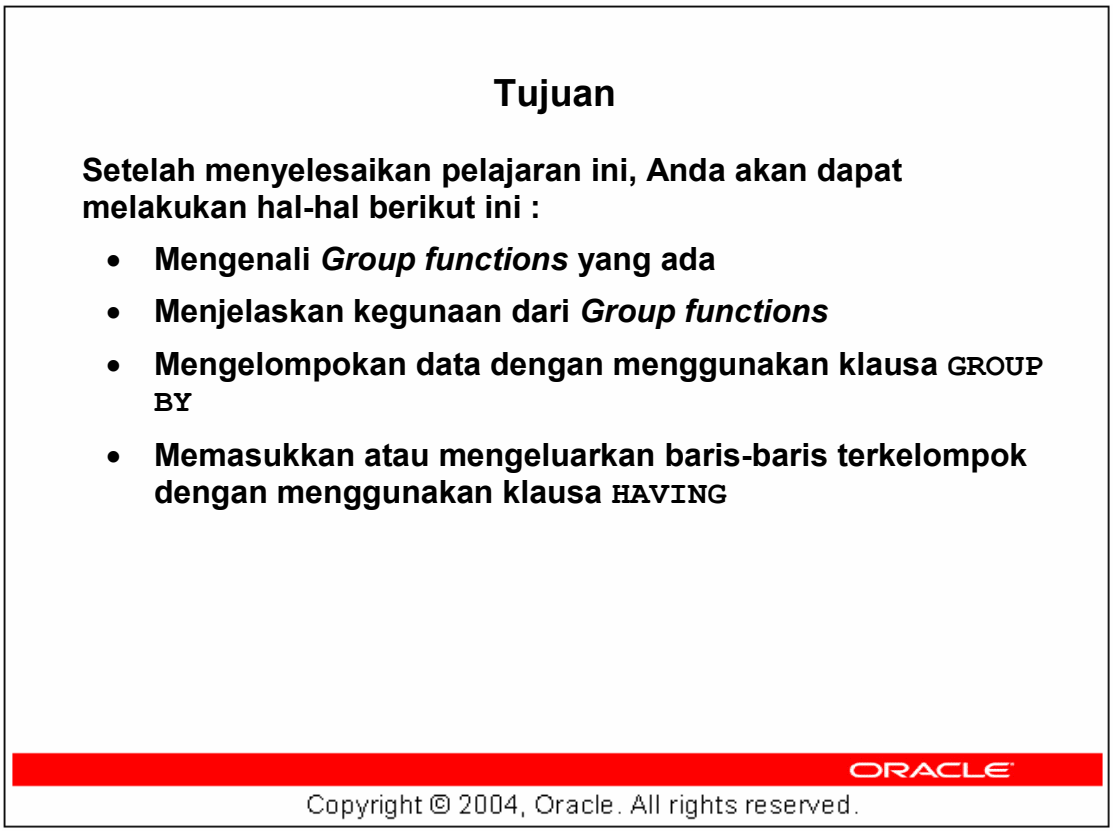
4

Melaporkan Data Agreget Menggunakan *Group Functions*

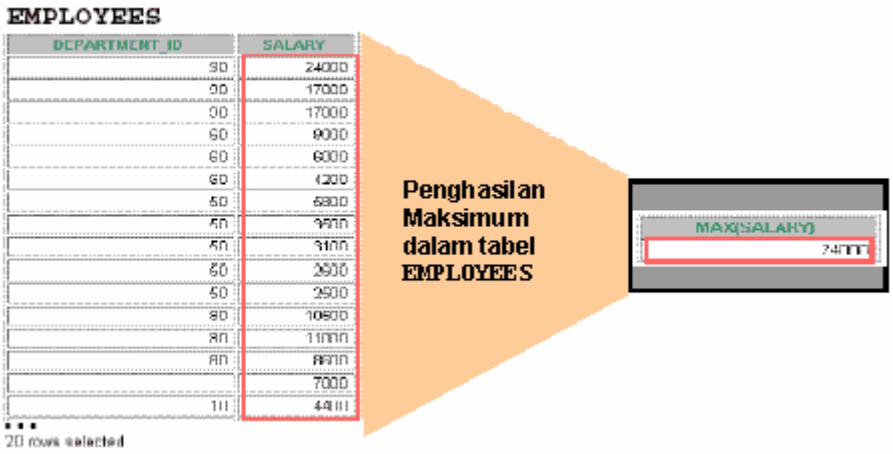


Tujuan

Pelajaran ini ditujukan pada fungsi-fungsi lebih lanjut. Fokusnya pada perolehan informasi ringkas (seperti rata-rata) untuk baris-baris terkelompok. Pelajaran ini mendiskusikan bagaimana mengelompokkan baris-baris dalam suatu tabel menjadi sekelompok kecil dan bagaimana untuk menentukan kriteria pencarian untuk baris-baris terkelompok.

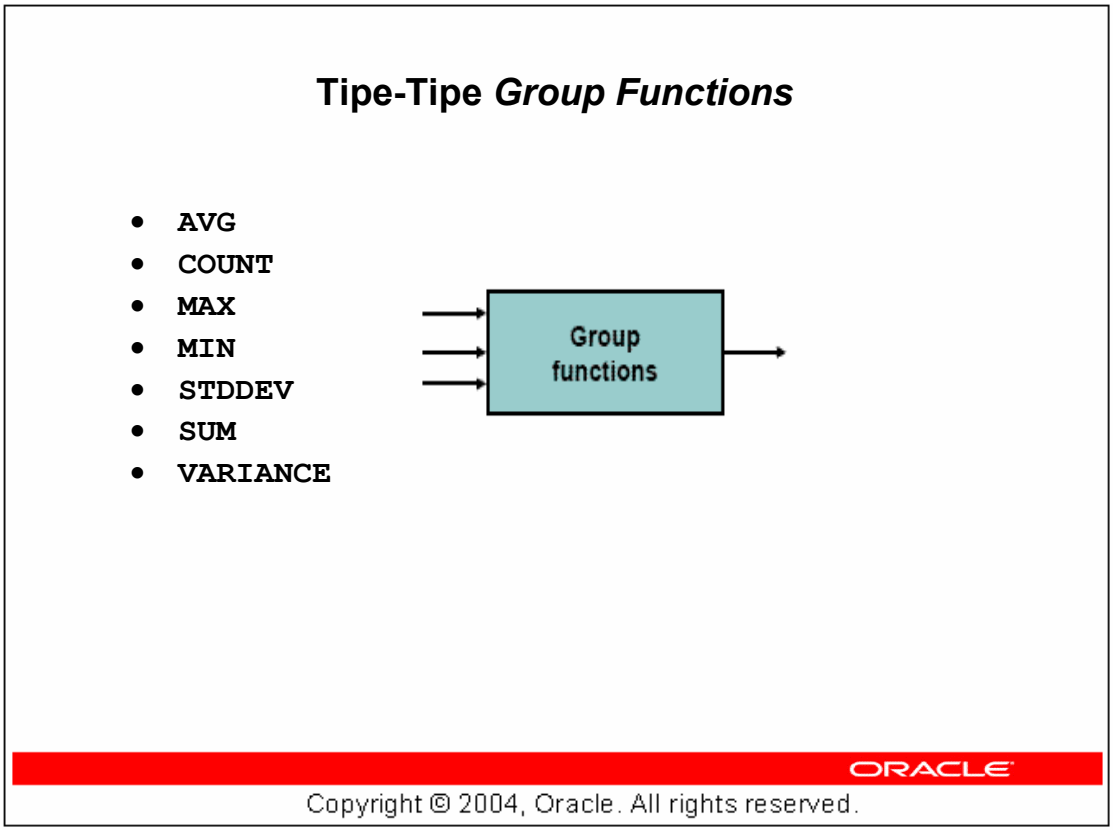
Apakah *Group Function* Itu ?

***Group function* beroperasi pada sekelompok baris-baris untuk memberikan satu hasil per kelompok.**



Group Functions

Tidak seperti *single-row function*, *group function* beroperasi pada sekelompok baris-baris untuk memberikan satu hasil per kelompok. Kelompok-kelompok ini mungkin terdiri dari seluruh tabel atau tabel yang terpisah ke dalam pengelompokkan.



Tipe-Tipe *Group Functions*

Setiap fungsi-fungsi menerima suatu argumen. Tabel berikut menunjukkan pilihan-pilihan yang dapat Anda gunakan di dalam sintak :

Fungsi	Keterangan
AVG([DISTINCT | ALL] *n*)	Rata-rata nilai dari suatu *n*, mengabaikan nilai-nilai *null*
COUNT ({ * | [DISTINCT | ALL] *expr*})	Jumlah baris-baris, dimana *expr* memeriksa ke sesuatu yang lain dari pada *null* (menghitung semua baris-baris yang dipilih menggunakan *, termasuk duplikat-duplikat dan baris-baris *null*)
MAX ([DISTINCT | ALL] *expr*)	Nilai maksimum dari *expr*, mengabaikan nilai-nilai *null*
MIN ([DISTINCT | ALL] *expr*)	Nilai minimum dari *expr*, mengabaikan nilai *null*
STDDEV ([DISTINCT | ALL] x)	Standar deviasi dari *n*, mengabaikan nilai-nilai *null*
SUM ([DISTINCT | ALL] n)	Nilai-nilai penjumlahan dari *n*, mengabaikan nilai-nilai *null*
VARIANCE ([DISTINCT | ALL] x)	Varian dari *n*, mengabaikan nilai-nilai *null*

Group Functions : Sintak

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

ORACLE

Pedoman-pedoman Untuk Menggunakan *Group Functions*

- `DISTINCT` membuat suatu fungsi hanya mencakup nilai-nilai yang tidak sama (*nonduplicate*); `ALL` membuatnya mencakup setiap nilai, termasuk duplikat-duplikat. *Default*-nya adalah `ALL` dan karena itu tidak perlu ditentukan.
- Tipe data-tipe data untuk fungsi-fungsi dengan suatu argumen `expr` mungkin `CHAR`, `VARCHAR2`, `NUMBER`, atau `DATE`.
- Semua *Group functions* mengabaikan nilai-nilai *null*. Untuk mengganti suatu nilai untuk nilai-nilai *null*, gunakanlah fungsi-fungsi `NVL`, `NVL2`, atau `COALESCE`.

Menggunakan Fungsi-Fungsi AVG dan SUM

Anda dapat menggunakan AVG dan SUM untuk data numerik.

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

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

ORACLE

Menggunakan *Group Functions*

Anda dapat menggunakan fungsi-fungsi AVG, SUM, MIN, dan MAX pada kolom-kolom yang dapat menyimpan data numerik. Contoh pada slide menampilkan penghasilan rata-rata, tertinggi, terendah, dan total penghasilan bulanan untuk semua sales representative.

Menggunakan Fungsi-Fungsi MIN dan MAX

Anda dapat menggunakan MIN dan MAX untuk tipe-tipe data *numeric*, *character*, dan *date*.

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

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

ORACLE

Menggunakan *Group Function* (lanjutan)

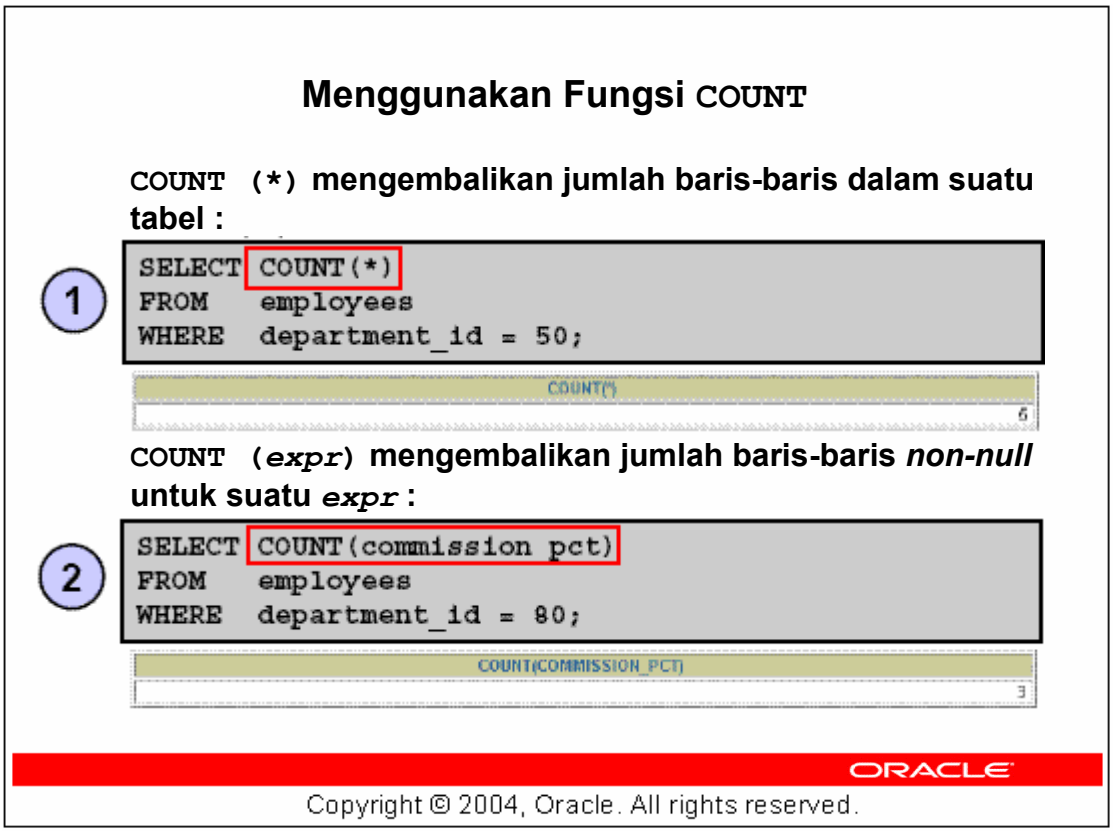
Anda dapat menggunakan fungsi MAX dan MIN untuk tipe data-tipe data *numeric*, *character*, dan *date*. Contoh pada slide menampilkan pegawai yang paling baru dan paling lama.

Contoh berikut ini menampilkan nama belakang pegawai yang pertama dan nama belakang pegawai yang terakhir dalam suatu daftar abjad dari semua pegawai :

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

MIN (LAST_NAME)	MAX (LAST_NAME)
Abel	Zlotkey

Catatan : Fungsi-fungsi AVG, SUM, VARIANCE, dan STDDEV hanya dapat digunakan pada tipe data *numeric*, MAX dan MIN tidak dapat digunakan pada tipe data-tipe data LOB atau LONG.



Fungsi COUNT

Fungsi COUNT memiliki tiga format :

- COUNT (*)
- COUNT (*expr*)
- COUNT (DISTINCT *expr*)

COUNT (*) mengembalikan jumlah baris-baris dalam suatu tabel yang memenuhi kriteria dari pernyataan SELECT, termasuk baris-baris yang sama dan baris-baris yang berisi nilai-nilai *null* di setiap kolom.

Jika suatu klausa WHERE adalah termasuk dalam pernyataan SELECT, COUNT (*) mengembalikan jumlah dari baris-baris yang memenuhi kondisi klausa WHERE.

Yang membedakannya, COUNT (*expr*) mengembalikan jumlah dari nilai-nilai *non-null* yang berada dalam kolom yang diidentifikasi oleh *expr*.

COUNT (DISTINCT *expr*) mengembalikan jumlah dari nilai-nilai yang unik ,*non-null* yang ada dalam kolom diidentifikasi oleh *expr*.

Contoh :

1. Contoh pada slide menampilkan jumlah dari pegawai di department 50
2. Contoh pada slide menampilkan jumlah dari pegawai di department 80 yang mendapat suatu komisi.

Menggunakan Kata Kunci DISTINCT

- **COUNT(DISTINCT expr) mengembalikan jumlah dari nilai-nilai *non-null* berbeda dari *expr*.**
- **Untuk menampilkan jumlah dari nilai-nilai departemen berbeda dalam tabel EMPLOYEES :**

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

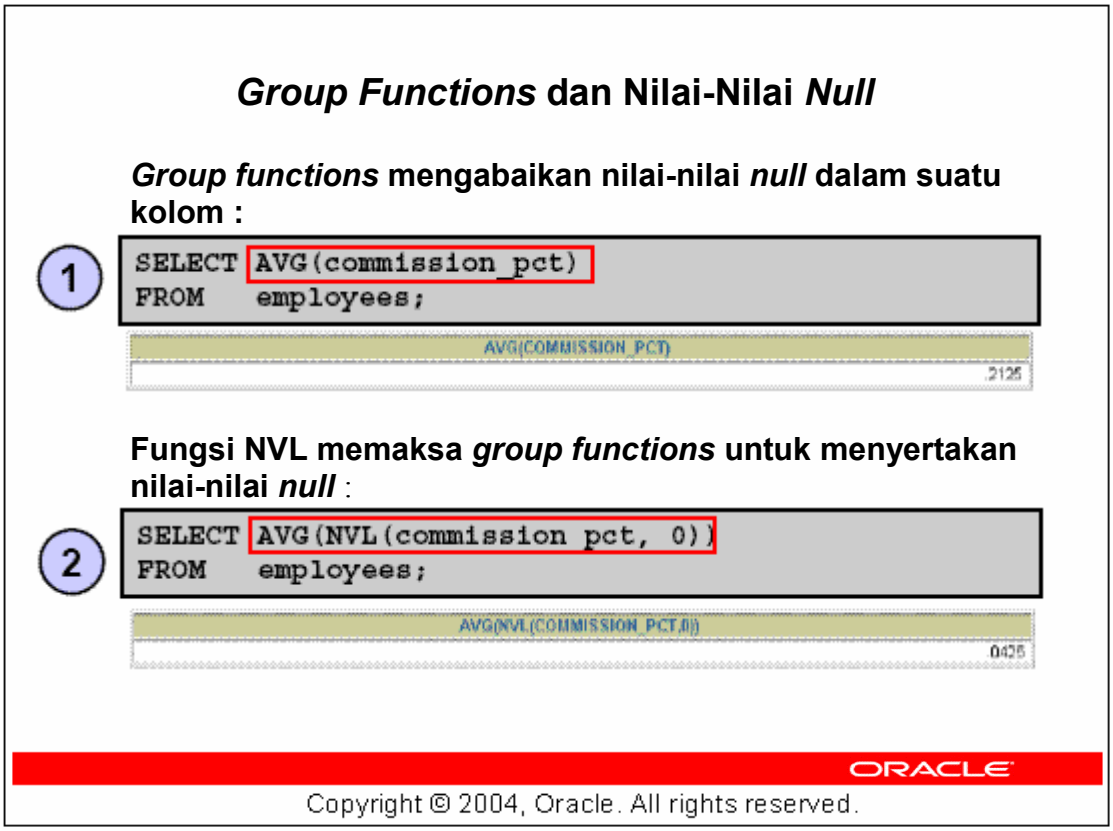
COUNT(DISTINCTDEPARTMENT_ID)
7

ORACLE

Menggunakan Kata Kunci DISTINCT

Gunakan kata kunci DISTINCT untuk menghilangkan penghitungan terhadap nilai-nilai yang sama dalam suatu kolom.

Contoh pada slide menampilkan jumlah hanya departemen tertentu pada tabel EMPLOYEES.



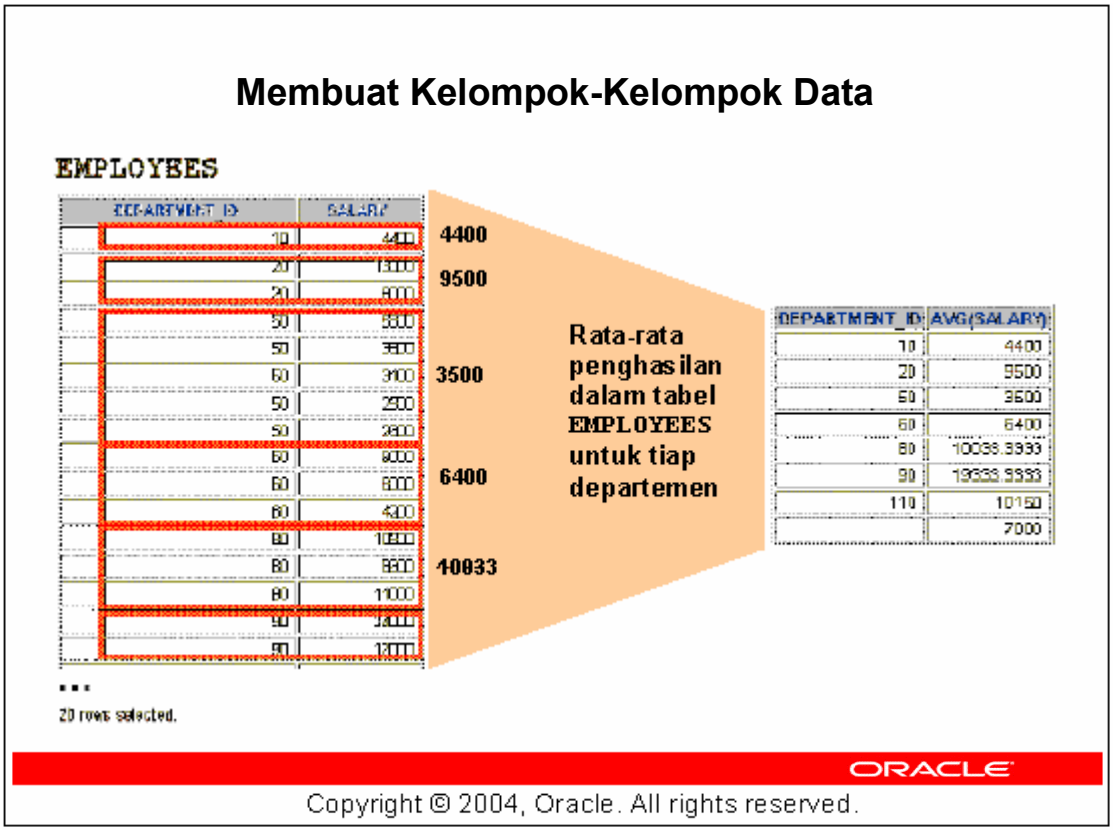
Group Functions dan Nilai-Nilai *Null*

Semua *group functions* mengabaikan nilai-nilai *null* dalam kolom.

Fungsi `NVL` memaksa *group functions* untuk menyertakan nilai-nilai *null*.

Contoh:

1. Rata-rata dihitung berdasarkan ***hanya*** pada baris-baris dalam suatu tabel yang menyimpan nilai yang valid di kolom `COMMISION_PCT`. Rata-rata dihitung sebagai total komisi yang dibayarkan ke semua pegawai dibagi dengan jumlah pegawai yang menerima komisi (empat).
2. Rata-rata dihitung berdasarkan ***semua*** baris-baris dalam suatu tabel, tidak peduli apakah nilai-nilai di kolom `COMMISION_PCT` *null* atau bukan. Rata-rata dihitung sebagai total komisi yang dibayarkan ke semua pegawai dibagi dengan jumlah total pegawai dalam perusahaan (20).



Membuat Kelompok-kelompok Data

Sampai poin diskusi kita ini, semua *group functions* sudah memperlakukan tabel sebagai sekelompok besar informasi.

Suatu saat, bagaimanapun, Anda perlu untuk membagi informasi kedalam kelompok-kelompok yang lebih kecil. Ini dapat dilakukan dengan menggunakan suatu klausa `GROUP BY`.

Membuat Data-Data Terkelompok : Sintak Klausa GROUP BY

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

Anda dapat membagi baris-baris dalam suatu tabel kedalam sekelompok kecil dengan menggunakan klausa GROUP BY.

Klausa GROUP BY

Anda dapat menggunakan klausa GROUP BY untuk membagi baris-baris dalam suatu tabel menjadi kelompok-kelompok. Kemudian Anda dapat menggunakan *group functions* untuk mengembalikan informasi ringkas untuk setiap kelompok.

Dalam sintak :

Group_by_expression	kolom-kolom tertentu yang nilai-nilainya menentukan dasar untuk pengelompokan baris-baris

Pedoman-pedoman

- Jika anda menyertakan *group functions* pada klausa SELECT, anda tidak dapat memilih hasil-hasil secara individu dengan baik, ***kecuali*** kolom individu muncul pada klausa GROUP BY. Anda akan menerima pesan kesalahan jika anda keliru menyertakan daftar kolom di klausa GROUP BY.
- Menggunakan klausa WHERE, anda dapat mengeluarkan baris-baris sebelum membaginya kedalam kelompok-kelompok.
- Anda harus menyertakan ***kolom-kolom*** dalam klausa GROUP BY.
- Anda tidak dapat menggunakan kolom alias dalam klausa GROUP BY.

Menggunakan Klausa GROUP BY

Semua kolom pada daftar SELECT yang bukan *group functions* harus ada pada klausa GROUP BY.

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

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

ORACLE

Menggunakan Klausa GROUP BY

Ketika menggunakan klausa GROUP BY, pastikan bahwa semua kolom pada daftar SELECT yang bukan *group functions* disertakan pada klausa GROUP BY. Contoh pada slide menampilkan nomor departemen dan rata-rata penghasilan untuk tiap departemen. Berikut ini adalah bagaimana pernyataan SELECT, beserta klausa GROUP BY, dievaluasi :

- Klausa SELECT menentukan kolom-kolom yang akan diambil, sebagai berikut :
  - Kolom nomor departement dalam tabel EMPLOYEES
  - Rata-rata dari semua penghasilan dalam suatu kelompok yang ditentukan oleh klausa GROUP BY.
- Klausa FROM menentukan tabel-tabel yang harus diakses database : tabel EMPLOYEES.
- Klausa WHERE menentukan baris-baris yang akan diambil. Karena tidak ada klausa WHERE, semua baris secara *default* akan diambil.
- Klausa GROUP BY menentukan bagaimana baris-baris akan dikelompokkan. Baris-baris dikelompokkan berdasarkan nomor departemen, jadi fungsi AVG yang diterapkan pada kolom penghasilan akan menghitung *rata-rata gaji untuk tiap departemen*.

Menggunakan Klausa GROUP BY

Kolom GROUP BY tidak harus ada pada daftar SELECT.

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

AVG(SALARY)
4400
9500
3500
6400
10033.3333
19333.3333
10150
7000

ORACLE

Menggunakan Klausa GROUP BY (lanjutan)

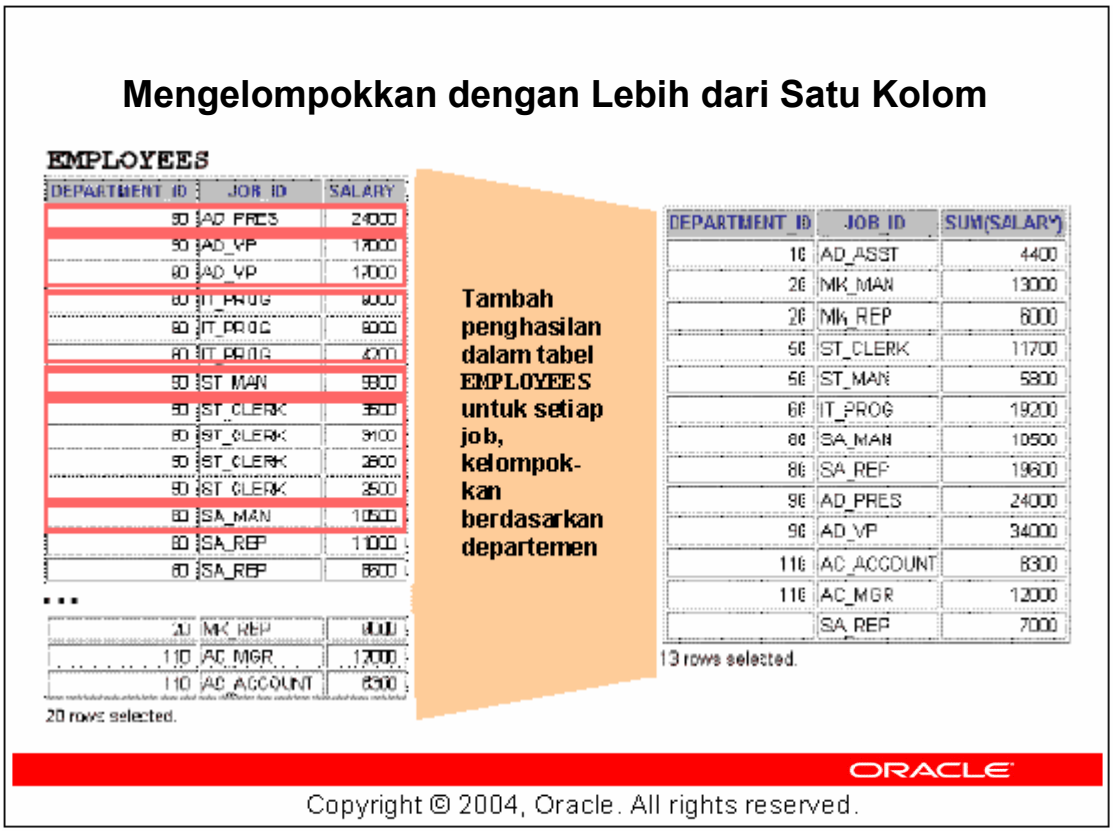
Kolom GROUP BY tidak harus ada pada klausa SELECT. Sebagai contoh, pernyataan SELECT pada slide menampilkan rata-rata penghasilan tiap departemen tanpa menampilkan nomor masing-masing departemen. Tanpa nomor-nomor departemen, bagaimanapun, hasilnya akan menjadi tidak ada artinya.

Anda dapat menggunakan *group functions* pada klausa ORDER BY:

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

DEPARTMENT_ID	AVG(SALARY)
50	3500
10	4400
60	6400
. . .	
90	19333.3333

8 rows selected.



Kelompok Dalam Kelompok

Kadang-kadang Anda perlu untuk melihat hasil untuk kelompok dalam kelompok. Slide menampilkan suatu laporan yang menunjukkan total penghasilan yang dibayarkan pada masing-masing job di tiap departemen.

Tabel `EMPLOYEES` dikelompokkan terlebih dahulu berdasarkan nomor departemen dan kemudian pengelompokkan berdasarkan job. Sebagai contoh, empat petugas stok di departemen 50 dikelompokkan bersama, dan suatu hasil tunggal (total penghasilan) dihasilkan untuk semua petugas stok dalam kelompok tersebut.

Menggunakan Klausa GROUP BY pada *Multiple* Kolom

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

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected

ORACLE

Kelompok Dalam Kelompok (lanjutan)

Anda dapat mengembalikan hasil ringkasan untuk kelompok dan sub-kelompok dengan mendaftar lebih dari satu kolom GROUP BY. Anda dapat menentukan *default* urutan pernyortiran suatu hasil berdasarkan urutan kolom-kolom pada klausa GROUP BY. Contoh pada slide, pernyataan SELECT yang menyertakan sebuah klausa GROUP BY yang dievalusi sebagai berikut:

- Klausa SELECT menentukan kolom yang akan diambil :
  - Nomor departement dalam tabel EMPLOYEES
  - Job ID dalam tabel EMPLOYEES
  - Total semua penghasilan dalam kelompok yang Anda tentukan pada klausa GROUP BY.
- Klausa FROM menentukan tabel-tabel yang harus diakses database : tabel EMPLOYEES.
- Klausa GROUP BY menentukan bagaimana Anda harus mengelompokkan baris-baris :
  - Pertama, baris-baris dikelompokkan berdasarkan nomor departemen.
  - Kedua, baris-baris dikelompokkan berdasar job ID dalam kelompok-kelompok nomor departemen.

Jadi fungsi SUM diterapkan ke kolom penghasilan untuk semua job ID di tiap kelompok nomor departemen.

Query-Query Ilegal Menggunakan *Group Functions*

Beberapa kolom atau ekpresi pada daftar `SELECT` yang bukan fungsi agreget harus ada dalam klausa `GROUP BY` :

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

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Kolom tidak ada pada klausa `GROUP BY`

ORACLE

Query-Query Ilegal Menggunakan *Group Functions*

Kapanpun Anda menggunakan suatu campuran dari item-item individual (`DEPARTMENT_ID`) dan *group function* (`COUNT`) pada pernyataan `SELECT` yang sama, Anda harus menyertakan suatu klausa `GROUP BY` yang menentukan item-item individu (dalam kasus ini, `DEPARTMENT_ID`). Jika klausa `GROUP BY` tidak ada, maka muncul pesan kesalahan "*not a single-group group function*" dan sebuah asterisk (*) akan menunjuk kepada kolom yang bermasalah. Anda dapat memperbaiki kesalahan pada slide dengan menambahkan klausa `GROUP BY` :

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

DEPARTEMENT_ID	COUNT(LAST_NAME)
10	1
20	2
. . .	
	1

8 rows selected

Setiap kolom atau ekspresi pada daftar `SELECT` yang bukan merupakan fungsi agreget harus ada pada klausa `GROUP BY`.

Query-Query Ilegal Menggunakan *Group Functions*

- **Anda tidak bisa menggunakan klausa `WHERE` untuk membatasi kelompok-kelompok.**
- **Anda gunakan klausa `HAVING` untuk membatasi (*restrict*) kelompok-kelompok.**
- **Anda tidak bisa menggunakan *group functions* pada klausa `WHERE`.**

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

```
WHERE  AVG(salary) > 8000
       *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

Klausa WHERE tidak bisa digunakan untuk membatasi kelompok-kelompok

ORACLE

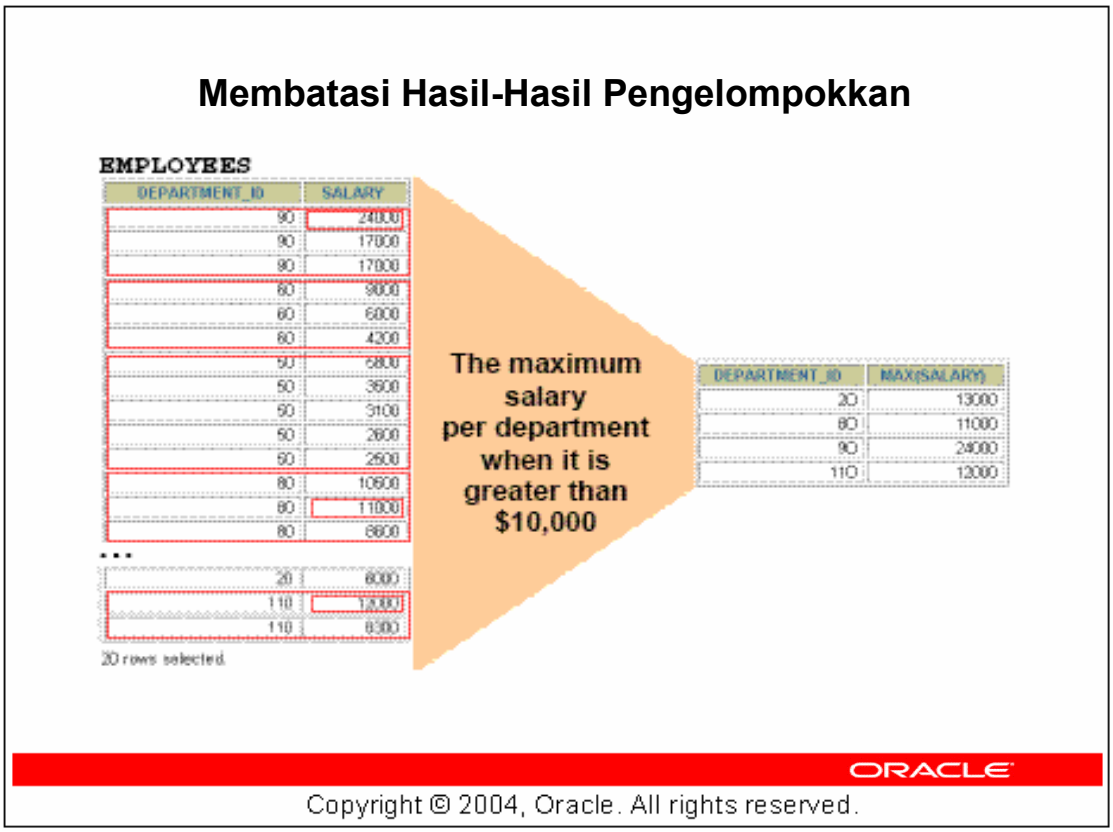
Query-Query Ilegal Menggunakan *Group Functions* (lanjutan)

Klausa `WHERE` tidak dapat digunakan untuk membatasi kelompok. Pernyataan `SELECT` pada contoh slide menghasilkan suatu kesalahan karena klausa `WHERE`-nya digunakan untuk membatasi hasil rata-rata penghasilan dari departemen-departemen yang memiliki rata-rata penghasilan di atas $8.000.

Anda dapat memperbaiki kesalahan pada contoh dengan menggunakan klausa `HAVING` untuk membatasi kelompok-kelompok :

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

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150



Membatasi Hasil-hasil Pengelompokkan

Dengan cara yang sama Anda gunakan klausa `WHERE` untuk membatasi baris-baris yang Anda pilih, begitupula Anda gunakan klausa `HAVING` untuk membatasi pengelompokkan. Untuk mencari penghasilan maksimum pada setiap departemen yang memiliki penghasilan maksimum lebih besar dari $10.000, Anda perlu melakukan hal-hal berikut:

1. Cari rata-rata penghasilan untuk setiap departemen dengan mengelompokkan berdasarkan nomor departemen.
2. Membatasi pengelompokkan ke departemen-departemen tersebut dengan penghasilan maksimum lebih besar dari $10.000.

Membatasi Hasil-hasil Pengelompokkan dengan Klausa HAVING

Saat Anda menggunakan klausa HAVING, server Oracle membatasi pengelompokkan sebagai berikut :

1. **Baris-baris dikelompokkan.**
2. ***Group Function* diterapkan.**
3. **Pengelompokkan yang memenuhi klausa HAVING ditampilkan.**

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

ORACLE

Membatasi Hasil-Hasil Pengelompokkan dengan Klausa HAVING

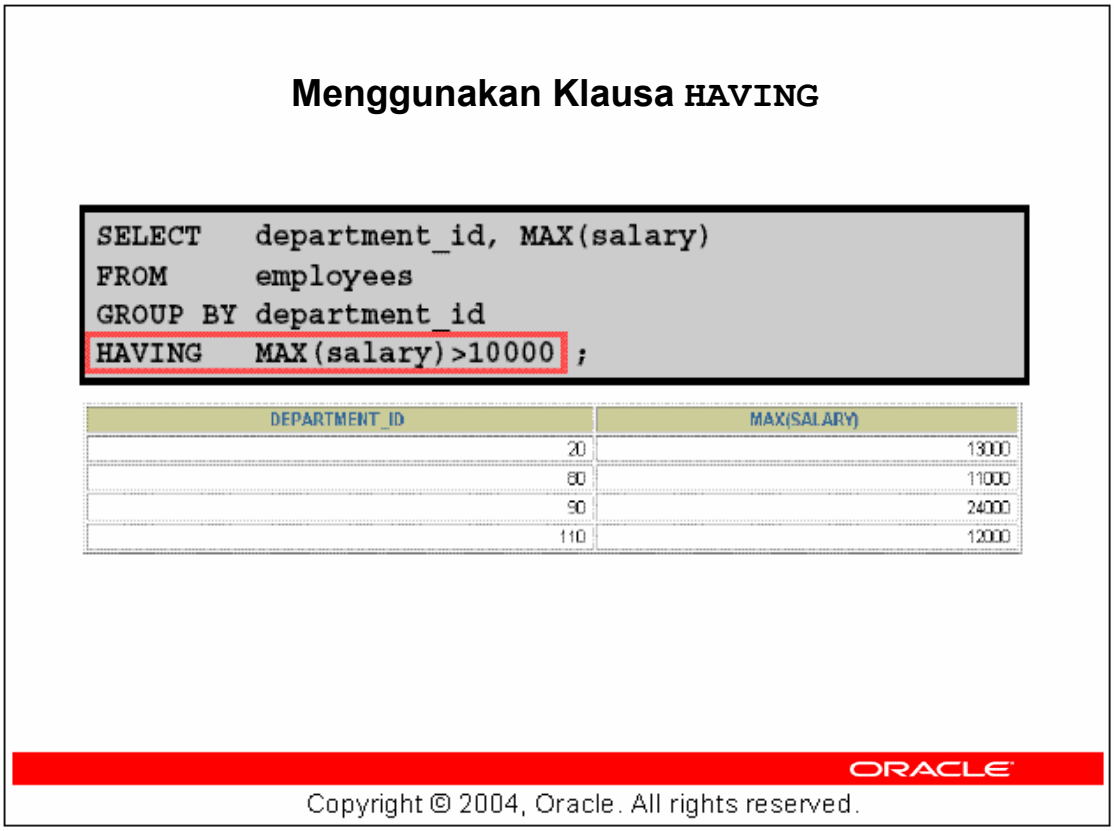
Anda menggunakan klausa HAVING untuk menentukan pengelompokkan-pengelompokkan yang akan ditampilkan, lebih jauh lagi membatasi pengelompokkan berdasarkan informasi agreget.

Dalam sintak, *group_condition* mengembalikan baris-baris pengelompokkan tertentu ke pengelompokkan tersebut yang memenuhi kondisi *true*.

Server Oracle melakukan langkah-langkah berikut ketika Anda menggunakan klausa HAVING:

1. Baris-baris dikelompokkan
2. Suatu *Group function* diterapkan pada pengelompokkan
3. Suatu pengelompokkan yang memenuhi kriteria pada klausa HAVING ditampilkan

Klausa HAVING dapat mendahului klausa GROUP BY, tetapi disarankan bahwa Anda menempatkan GROUP BY lebih dulu karena lebih logis. Pengelompokkan adalah kondisi dan *group functions* dihitung sebelum klausa HAVING diterapkan ke pengelompokkan pada daftar SELECT.



Penggunaan Klausa HAVING

Contoh pada slide menampilkan nomor-nomor departemen dan penghasilan maksimum untuk departemen-departemen dengan penghasilan suatu maksimum lebih dari $10.000

Anda dapat menggunakan klausa `GROUP BY` tanpa menggunakan *group function* pada daftar `SELECT`.

Jika Anda membatasi baris-baris berdasarkan hasil dari suatu *group function*, Anda harus menggunakan klausa `GROUP BY` sebagaimana klausa `HAVING`.

Contoh berikut menampilkan nomor-nomor departemen dan penghasilan rata-rata untuk departemen-departemen dengan suatu penghasilan maksimum lebih besar dari $10.000

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

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

Menggunakan Klausa 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
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Penggunaan Klausa HAVING (lanjutan)

Contoh pada slide menampilkan job ID dan total penghasilan bulanan untuk setiap pekerjaan yang memiliki suatu total daftar penghasilan melebihi $13.000. Contoh tersebut tidak menyertakan para sales representative dan daftar disortir berdasarkan total penghasilan bulanan.

Group Functions Bersarang

Menampilkan penghasilan rata-rata tertinggi :

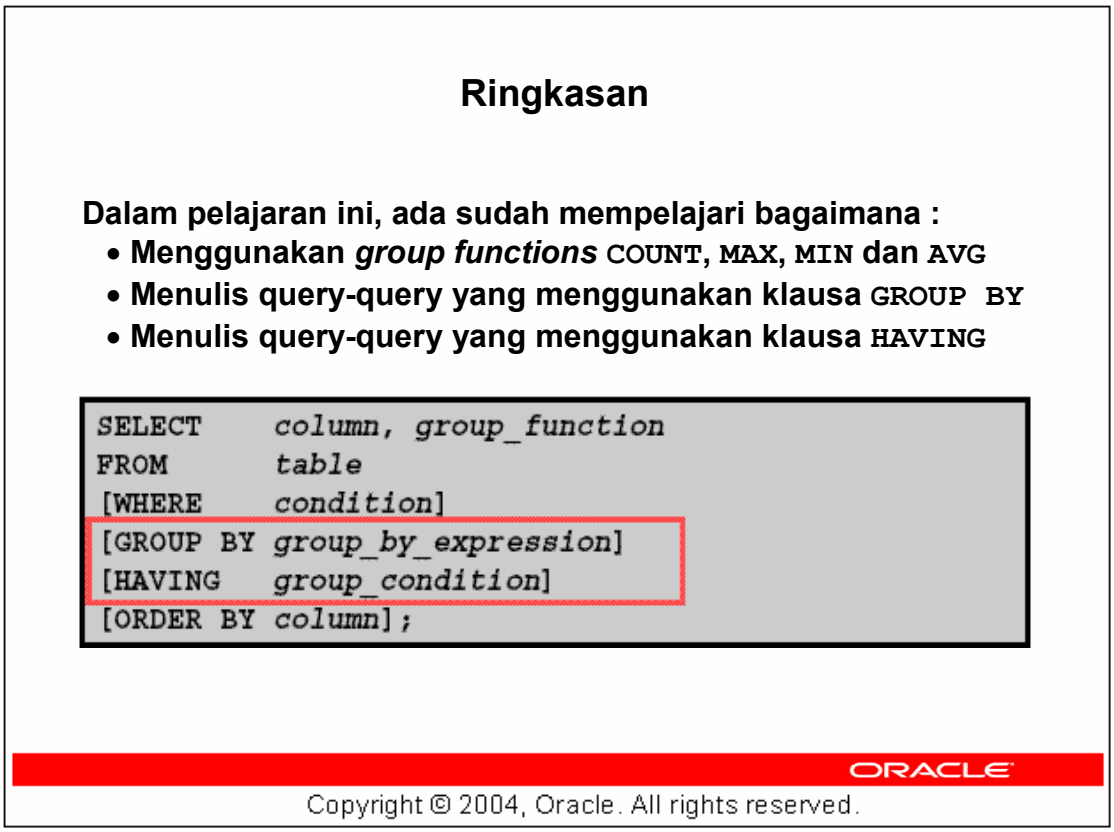
```
SELECT   MAX(AVG(salary))
FROM     employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

ORACLE

Group Functions Bersarang

Group functions dapat disarangkan hingga 2 kedalaman. Contoh pada slide menampilkan penghasilan rata-rata tertinggi.



Ringkasan

Beberapa *group function* ada dalam SQL, seperti berikut ini :
AVG, COUNT, MAX, MIN, SUM, STDDEV, dan VARIANCE

Anda dapat membuat sub-pengelompokkan dengan menggunakan GROUP BY. Pengelompokkan dapat dibatasi menggunakan klausa HAVING.

Tempatkan klausa-klausa HAVING dan GROUP BY setelah klausa WHERE dalam suatu pernyataan. Urutan dari klausa-klausa HAVING dan GROUP setelah WHERE tidak penting. Tempatkan klausa ORDER BY diakhir.

Server Oracle memeriksa klausa-klausa dengan urutan sebagai berikut :
1. Jika pernyataan mengandung klausa WHERE, server membentuk baris-baris kandidat.
2. Server mengidentifikasi pengelompokkan yang ditentukan pada klausa GROUP BY.
3. Klausa HAVING lebih jauh membatasi hasil pengelompokkan yang tidak memenuhi kriteria pada klausa HAVING.

Catatan : Untuk daftar lengkap dari *group functions*, lihat *Oracle SQL Reference*.