



Tatap MUka

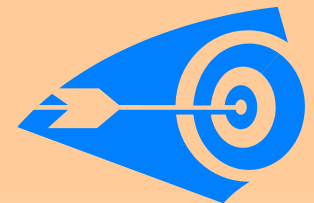
3

**SQL Query &
Agregate Function**

Arif Basofi

Topik

- **Sintaks Dasar SQL Query**
- Operasi Aritmatika dalam SQL
- SQL Fungsi Agregate dan Group
- SQL Joint Table



SQL - Pendahuluan

- Bahasa user yang meminta (request) pada database untuk menyediakan suatu data yang diperlukan menggunakan tipe bahasa khusus yang disebut dengan **Structured Query Language (SQL** atau eja: "sequel)" .
- **SQL** adalah **bahasa fungsional**, yaitu suatu bahasa yang memungkinkan user untuk menentukan tipe dari sesuatu yang ingin mereka dapatkan.
- Bahasa fungsional tersebut **tidak sama** dengan bahasa pemrograman yang lain semisal C++, pascal atau COBOL.
- Bahasa-bahasa tersebut disebut sebagai bahasa '**prosedural**' karena **membutuhkan penulisan program atau prosedur** untuk mendapatkan suatu informasi.
- Sebaliknya, **SQL** secara eksplisit mendefinisikan **hasil akhir** yang diinginkan, sedangkan metode untuk mendapatkan data tersebut dilakukan sendiri oleh database.

SQL - Pendahuluan

- Bentuk SQL Query umum:

```
SELECT [DISTINCT] < attribute-list >  
FROM < table-list >  
WHERE < condition >
```

- **Attribute- list:** adalah daftar nama atribut/kolom table yang berada dalam *table-list* dan nilainya didapatkan melalui query.
- **Table- list:** adalah daftar table relasi yang memiliki nama (dengan domain variabel pada tiap nama yang diberikan) untuk memproses query.
- **Condition:** adalah statemen perbandingan dalam SQL Query yang mengkombinasikan operator perbandingan AND, OR dan NOT.
- Sedangkan **DISTINCT** adalah keyword yang bersifat optional (boleh ditulis, boleh tidak) yang mengindikasikan suatu hasil query yang tidak memiliki duplikasi data. Secara default, didapatkan duplikasi pada hasil query (tanpa distinct).

SQL - Pendahuluan

- SQL dengan pemilihan data dapat dilakukan dengan menggunakan klausa **WHERE** pada contoh SQL seperti berikut :

```
SELECT *  
FROM emp  
WHERE empid = 39334;
```

- Statement SQL diatas meminta untuk menyediakan semua (*) data dari table EMP dimana nilai yang diminta ada pada kolom EMPID yang berisi nilai 39334.

SQL - Pendahuluan

- Sedangkan Blok kode berikut ini adalah bahasa pemrograman **prosedural** yang mengilustrasikan fungsi yang sama dengan statement SQL diatas.

```
Include <stdio.h>
Include <string.h>
Include <rdbms.h>

Int *empid;
Char *statement;

Type emp_rec is record (
Int empid;
Char[10] emp_name;
Int salary; )
Void main()
{ Access_table(emp);
  Open(statement.memaddr);
  Strcpy("SELECT * FROM EMP WHERE EMPID = 39334",statement.text);
  parse(statement);
  execute(statement);
  for (l=1,l=statement.results,l+1) {
  fetch(statement.result[l],emp_rec);
  printf(emp_rec);
  }
  close(statement.memaddr);
}
```

Perintah Select dengan Tanda * (Asterisk)

- Tanda * dalam perintah **SELECT** berfungsi untuk menampilkan **semua data** pada **semua kolom** dalam table database.
- Contoh:

```
SELECT * FROM DEPARTMENTS;
```

Hasilnya:

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
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
150	Shareholder Services		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700
210	IT Support		1700
220	NOC		1700
230	IT Helpdesk		1700
240	Government Sales		1700
250	Retail Sales		1700
260	Recruiting		1700
270	Payroll		1700

27 rows selected.

Perintah Select pada Kolom Tertentu

- Untuk menampilkan satu atau beberapa kolom tertentu atau bahkan pada urutan kolom tertentu saja dapat dilakukan dengan perintah SELECT dengan pilihan atribut kolom yang diinginkan.
- Contoh:

```
SELECT Department_id, Department_Name  
FROM DEPARTMENTS;
```

Hasilnya:

```
DEPARTMENT_ID DEPARTMENT_NAME  
-----  
10 Administration  
20 Marketing  
30 Purchasing  
40 Human Resources  
50 Shipping  
60 IT  
70 Public Relations  
80 Sales  
90 Executive  
100 Finance  
110 Accounting  
120 Treasury  
130 Corporate Tax  
140 Control And Credit  
150 Shareholder Services  
160 Benefits  
170 Manufacturing  
180 Construction  
190 Contracting  
200 Operations  
210 IT Support  
220 NOC  
230 IT Helpdesk  
240 Government Sales  
250 Retail Sales  
260 Recruiting  
270 Payroll  
  
27 rows selected.
```


Menampilkan Data Secara Urut

- Untuk menampilkan data secara urut pada kolom tertentu, dapat digunakan perintah ORDER BY.
- Secara default ORDER BY akan mengurutkan secara ASCENDING (urut naik), sebaliknya mengurutkan secara DESCENDING (urut menurun).
- Syntax sebagai berikut:

```
SELECT [DISTINCT] < attribute-list >
```

```
FROM < table-list >
```

```
[WHERE < condition >]
```

```
ORDER BY column_name ASC or DESC
```

- Contoh: Tampilkan data nama department berdasarkan huruf abjad.

```
SELECT department_id, department_name
FROM DEPARTMENTS
ORDER BY department_name;
```

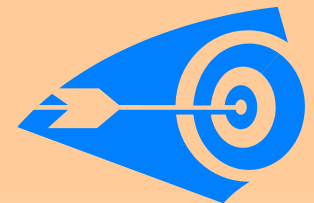
Hasilnya: 

```
DEPARTMENT_ID DEPARTMENT_NAME
-----
110 Accounting
10 Administration
160 Benefits
180 Construction
190 Contracting
140 Control And Credit
130 Corporate Tax
90 Executive
100 Finance
240 Government Sales
40 Human Resources
60 IT
230 IT Helpdesk
210 IT Support
170 Manufacturing
20 Marketing
220 NOC
200 Operations
270 Payroll
70 Public Relations
30 Purchasing
260 Recruiting
250 Retail Sales
80 Sales
150 Shareholder Services
50 Shipping
120 Treasury

27 rows selected.
```

Topik

- Sintaks Dasar SQL Query
- **Operasi Aritmatika dalam SQL**
- SQL Fungsi Agregate dan Group
- SQL Joint Table



Ekspresi Aritmetika pada SQL Query

- Ekspresi aritmetika dalam SQL, dapat menggunakan operator:

OPERATOR	DESKRIPSI
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian

- Eksresi aritmetika dapat diterapkan pada klausa **SELECT**.

- Contoh:

```
SELECT last_name, salary, salary+300
```

```
FROM employees;
```

LAST NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernet	6000	6300

■ ■ ■

20 rows selected.

Nilai NULL pada SQL Query

- Hasil SQL Query ada kemungkinan menghasilkan nilai **NULL**.
- Nilai **NULL** adalah nilai **unavailable**, **unassigned**, **unknown**, atau **inapplicable**
- **NULL** tidak sama dengan nilai **NOL** (\emptyset , **zero**) atau **spasi kosong** (**blank space**).
- Nilai **NULL** jika digunakan pada operasi aritmetik tetap akan bernilai **NULL**.

```
SELECT last_name, job_id, salary, commission_pct  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

Penggunaan Kolom Alias pada SQL Query

- Kolom **alias** :
 - Memberikan nama lain kolom
 - Berguna saat melakukan kalkulasi aritmetika
 - Gunakan tanda petik (tunggal / dobel tergantung DBMS yang digunakan), jika terdapat spasi atau karakter khusus dalam alias.

```
SELECT last_name AS name, commission_pct comm
FROM employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

Name	Annual Salary
King	286000
Kochhar	204000
De Haan	204000

...

20 rows selected.

Membatasi Data pada SQL Query

- Untuk membatasi data pada SQL query dapat menggunakan operator **pembanding** atau **Logika** pada klausa **WHERE**:

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Operator	Meaning
AND	Returns TRUE if <i>both</i> component conditions are true
OR	Returns TRUE if <i>either</i> component condition is true
NOT	Returns TRUE if the following condition is false

Membatasi Data pada SQL Query

- Contoh:

```
SELECT employee_id, last_name, job_id, department_id
FROM employees
WHERE department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000 ;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

Membatasi Data pada SQL Query

- Contoh:

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

Lower limit

Upper limit

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Membatasi Data pada SQL Query

- Contoh:

```
SELECT employee_id, last_name, salary, manager_id
FROM employees
WHERE manager_id IN (100, 101, 201) ;
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Membatasi Data pada SQL Query

- Kondisi **LIKE** :
 - **LIKE** digunakan untuk mencari kondisi karakter string yang cocok
 - Pencocokan dengan menggunakan:
 - % : menunjukkan nol atau sembarang karakter
 - _ : menunjukkan satu karakter yang memenuhi

- Contoh :

```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```

```
SELECT last_name
FROM employees
WHERE last_name LIKE '_o%';
```

LAST_NAME
Kochhar
Lorentz
Mourgos

Membatasi Data pada SQL Query

- Contoh:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >=10000
AND job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
OR job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

Membatasi Data pada SQL Query

- Contoh:

```
SELECT last_name, job_id
FROM employees
WHERE job_id
NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

Contoh SQL Query

Berikut contoh SQL Query:

1. Tampilkan tanggal lahir dan alamat dari semua pegawai yang bernama 'John B. Smith'.

```
SELECT BDATE, ADDRESS  
FROM EMPLOYEES  
WHERE FNAME='John' AND MINIT='B' AND LNAME='Smith';
```

2. Tampilkan nama belakang, gaji dan gaji setahun dari semua pegawai yang bekerja pada department no 20.

```
SELECT LNAME, SAL, SAL*12  
FROM EMPLOYEE  
WHERE DEPARTMENT_ID = 20;
```

Contoh SQL Query

3. Tampilkan nama depan, alamat, gaji dari semua pegawai yang memiliki gaji lebih besar dari \$500 dan berada pada department 50.

```
SELECT FNAME, ADDRESS, SALARY  
FROM EMPLOYEE  
WHERE SALARY > 500 AND DEPARTMENT_ID=50;
```











Latihan SQL Query

1. Tampilkan data kode pegawai, nama terakhir pegawai, gaji dan nomor departemen untuk pegawai yang berada di departemen 20, 30 dan 40.
2. Tampilkan data kode pegawai, nama terakhir pegawai, gaji dan "Kenaikan Gaji" (tambahan 20% dari gaji) untuk pegawai yang memiliki gaji antara 5 juta dan 10 juta.
3. Tampilkan data kode pegawai, nama pertama pegawai dan kode pekerjaan untuk pegawai yang memiliki email di 'gmail', 'hotmail' dan 'yahoo'.



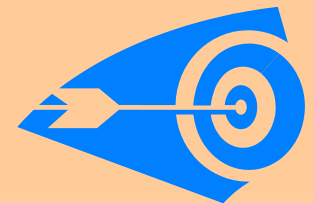
Column Name	Data Type	Index
EMPLOYEE_ID	NUMBER (6)	IDX_2
FIRST_NAME	VARCHAR2 (20)	IDX_6
LAST_NAME	VARCHAR2 (25)	IDX_6
EMAIL	VARCHAR2 (25)	IDX_1
PHONE_NUMBER	VARCHAR2 (20)	
HIRE_DATE	DATE	
JOB_ID	VARCHAR2 (10)	IDX_4
SALARY	NUMBER (8,2)	
COMMISSION_PCT	NUMBER (2,2)	
MANAGER_ID	NUMBER (6)	IDX_3
DEPARTMENT_ID	NUMBER (4)	IDX_5

Skema fisik Company:

EMPLOYEES			
	EMPLOYEE_ID	NUMBER (6)	 IDX_2
	FIRST_NAME	VARCHAR2 (20)	IDX_6
	LAST_NAME	VARCHAR2 (25)	 IDX_6
	EMAIL	VARCHAR2 (25)	 IDX_1
	PHONE_NUMBER	VARCHAR2 (20)	
	HIRE_DATE	DATE	
	JOB_ID	VARCHAR2 (10)	 IDX_4
	SALARY	NUMBER (8,2)	
	COMMISSION_PCT	NUMBER (2,2)	
	MANAGER_ID	NUMBER (6)	IDX_3
	DEPARTMENT_ID	NUMBER (4)	IDX_5

Topik

- Sintaks Dasar SQL Query
- Operasi Aritmatika dalam SQL
- **SQL Fungsi Agregate dan Group**
- SQL Joint Table



SQL – Fungsi Agregat dan Group Function

- **Fungsi agregat** adalah fungsi-fungsi yang mengambil kumpulan (collection) suatu himpunan data atau beberapa himpunan data dan mengembalikan dalam bentuk nilai tunggal.
- Terdapat 5 fungsi agregasi (agregat) baku, yaitu:

1. AVG

2. COUNT

3. MAX

4. MIN

5. SUM

- Contoh:

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

...

20 rows selected.

**Gaji maximum
pada table
EMPLOYEES.**

MAX(SALARY)
24000

S Q L – Fungsi Agregat dan Group Function

- Syntax dari Group Function, yaitu:

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

SQL – Fungsi Agregat dan Group Function

Fungsi AVG dan SUM

- Fungsi **AVG** digunakan untuk mencari nilai rata-rata pada suatu kolom data.
- Fungsi **SUM** digunakan untuk mencari nilai jumlah total pada suatu kolom

```
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

SQL – Fungsi Agregat dan Group Function

Fungsi MIN dan MAX

- Fungsi **MIN** digunakan untuk mencari nilai data paling kecil (minimum).
- Fungsi **MAX** digunakan untuk mencari nilai data paling besar (Maximum).

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

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

SQL – Fungsi Agregat dan Group Function

Fungsi COUNT

- Fungsi **COUNT** digunakan untuk mencari jumlah record data row (jumlah baris data yang dihasilkan dari query/banyaknya data).

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

COUNT(*)
5

S Q L – Fungsi Agregat dan Group Function

Fungsi COUNT

- Fungsi **COUNT** mengabaikan adanya data yang sifatnya **NULL VALUE**.
- Contoh berikut menampilkan jumlah data record pada suatu kolom.

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

COUNT(COMMISSION_PCT)
3

SQL – Fungsi Agregat dan Group Function

Membuat Group Data

- Ada keadaan penggunaan fungsi agregat untuk menghasilkan beberapa record data sekaligus berdasarkan kondisi khusus atau group dari suatu kolom tertentu.
- Maka dapat digunakan klausa **GROUP BY**.

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

- Membagi rows (baris-baris) dalam tabel menjadi group-group data yang lebih kecil dengan klausa **GROUP BY**.

SQL – Fungsi Agregat dan Group Function

Membuat Group Data

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

4400

9500

Rata-rata gaji

3500

Table
EMPLOYEES
untuk tiap
Department
(per-department)

6400

10033

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

...

20 rows selected.

SQL – Fungsi Agregat dan Group Function

Membuat Group Data

```
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.

SQL – Fungsi Agregat dan Group Function

Membuat Group Data

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

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

S Q L – Fungsi Agregat dan Group Function

Membuat Group Data

Meng-groupkan data pada lebih dari satu kolom

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600

...

20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Menjumlahkan gaji dalam table Employees per-department, per-job.

DEPARTMENT_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.

SQL – Fungsi Agregat dan Group Function

Membuat Group Data

Menggunakan GROUP BY Clause pada Multiple Columns

```
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.

S Q L – Fungsi Agregat dan Group Function

Membuat Pembatasan Group Data

- Fungsi **GROUP BY** dapat dibuat pembatasan dari data yang akan dihasilkan dengan menggunakan fungsi **HAVING**.

Dengan klausa **HAVING** dapat membatasi groups data:

1. Rows (baris-baris) akan di group.
2. Fungsi group dapat diaplikasikan.
3. Hasil group data yang match/sesuai dari klausa **HAVING** akan ditampilkan.

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

SQL – Fungsi Agregat dan Group Function

Membuat Pembatasan Group Data

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
...	...
20	6000
110	12000
110	8300

20 rows selected.

The maximum salary per department when it is greater than \$10,000

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

SQL – Fungsi Agregat dan Group Function

Membuat Pembatasan Group Data

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

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

SQL – Fungsi Agregat dan Group Function

Membuat Pembatasan Group Data

```
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

SQL – Fungsi Agregat dan Group Function

Membuat Pembatasan Group Data

Nesting Group Functions

Display the maximum average salary.

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

MAX(AVG(SALARY))
19333.3333

SQL – Fungsi Agregat dan Group Function

Illegal Query pada Group Function

- SQL Query yang menggunakan klausa GROUP BY harus lebih **hati-hati** dalam menentukan daftar kolom yang di-SELECT.
- **Kolom** / ekspresi dalam **SELECT** list yang **tidak menggunakan fungsi agregat**, harus **dimasukkan** 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
```

Column missing in the GROUP BY clause

S Q L – Fungsi Agregat dan Group Function

Illegal Query pada Group Function

- Jangan menggunakan klausa WHERE untuk membatasi hasil group (tidak cocok).
- Gunakan klausa **HAVING**.
- Jangan gunakan group function 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
```

Cannot use the WHERE clause to restrict groups

SQL

Contoh Soal:

1. Tampilkan gabungan antara nomor departemen pada table pegawai dengan nomor departemen pada table departemen.

```
select department_id
from employees
UNION
select department_id
from departments;
```

```
DEPARTMENT_ID
-----
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
28 rows selected.
```

SQL

2. Tampilkan data nomor departemen pada table Department yang tidak ada di table Employees (Department yang belum memiliki pegawai).

```
select department_id
from Departments
MINUS
select department_id
from Employees;
```

SQL

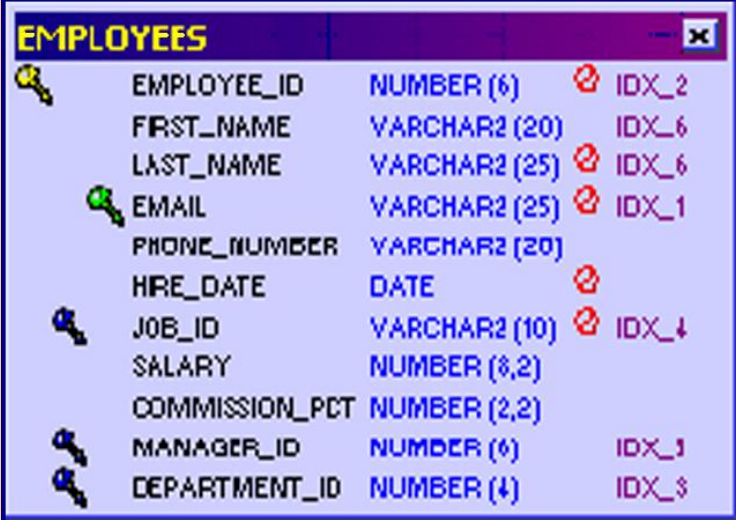
3. Tampilkan nomor departemen dan rata-rata gaji setahun pegawai untuk tiap-tiap department, dengan rata-rata gaji setahun tersebut antara \$10000 dan \$50000.











```
select department_id, avg(salary*12)
from employees
group by department_id
having avg(salary*12) between 10000 and 50000;
```

```
DEPARTMENT_ID  AVG(SALARY*12)
-----
              70          120000
              90          232000
              110          121800
```

Latihan S Q L

1. Tampilkan nama pertama, email dan gaji pegawai yang memiliki nama depan berawalan “Dwi” atau berakhiran “suki” dan memiliki domain email baik di yahoo.com, gmail.com, hotmail.com maupun eepis-its.edu. Tampilkan menurut abjad nama pertama.
2. Tampilkan nomor departemen, rata-rata gaji dan total gaji pegawai untuk tiap manager departemen yang memimpin.
3. Tampilkan nomor departemen, rata-rata gaji dan total gaji setahun pegawai untuk tiap-tiap manager departemen, dan yang memiliki total gaji setahun tadi $> \$5000$, serta nama belakang manager mengandung kata “sal”.



EMPLOYEES			
	EMPLOYEE_ID	NUMBER (6)	 IDX_2
	FIRST_NAME	VARCHAR2 (20)	 IDX_6
	LAST_NAME	VARCHAR2 (25)	 IDX_6
	EMAIL	VARCHAR2 (25)	 IDX_1
	PHONE_NUMBER	VARCHAR2 (20)	
	HIRE_DATE	DATE	
	JOB_ID	VARCHAR2 (10)	 IDX_4
	SALARY	NUMBER (8,2)	
	COMMISSION_PCT	NUMBER (2,2)	
	MANAGER_ID	NUMBER (6)	 IDX_3
	DEPARTMENT_ID	NUMBER (4)	 IDX_5