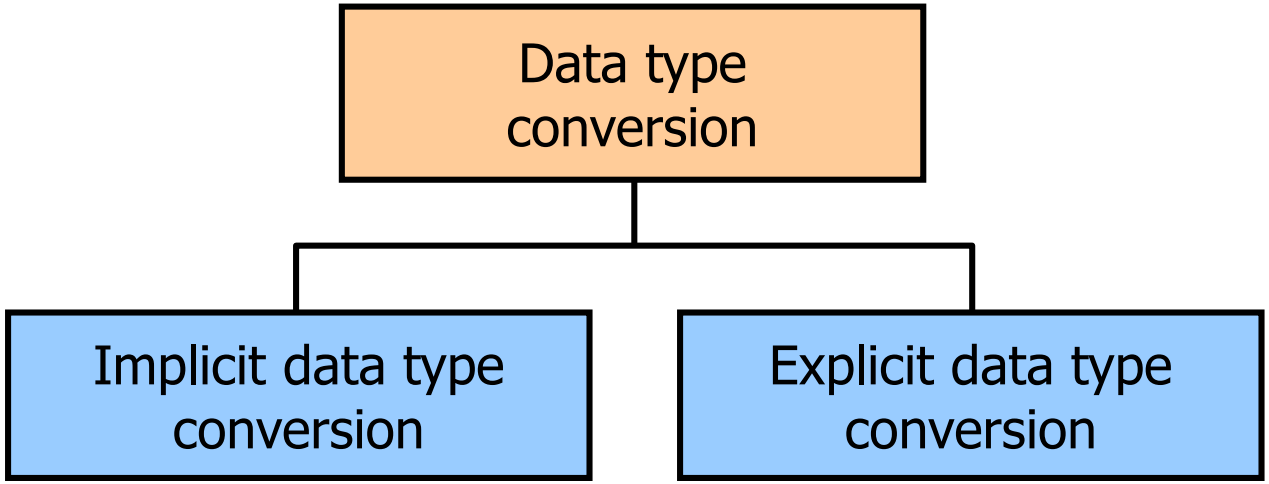
www.itsci.mju.ac.th/sayan

LEC 04: SQL FUNCTIONS

SAYAN UNANKARD

1/2558

CONVERSION FUNCTIONS



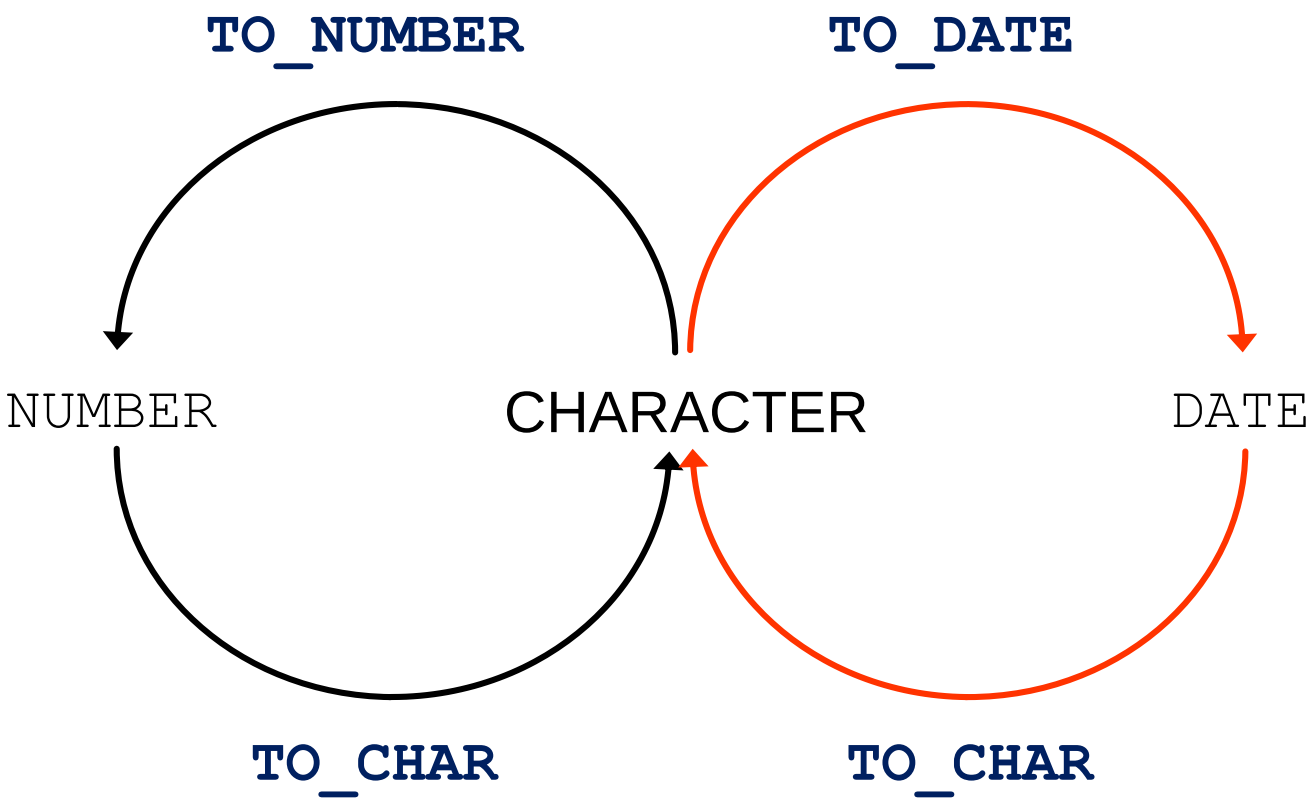
IMPLICIT DATA TYPE CONVERSION

Oracle server สามารถเปลี่ยนแปลงค่าได้โดยอัตโนมัติ ดังนี้

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

From	To
NUMBER	VARCHAR2 or CHAR
DATE	VARCHAR2 or CHAR

EXPLICIT DATA TYPE CONVERSION



USING THE TO_CHAR FUNCTION WITH DATES

TO_CHAR(*date*, '*format_model*')

รูปแบบของ format_model

- ต้องอยู่ภายในเครื่องหมาย single quotation marks
- เป็น case-sensitive
- สามารถกำหนดด้วยรูปแบบวันที่ตามที่ต้องการได้
- มีคำสั่ง `fm` element เพื่อจัดการช่องว่างหรือเลขศูนย์ที่นำหน้า ออกไปได้ เช่น 01/03/2008

ELEMENTS OF THE DATE FORMAT MODEL

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for the month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

ELEMENTS OF THE DATE FORMAT MODEL

รูปแบบของเวลา

HH24:MI:SS AM	15:45:32 PM

การแทรกข้อความเข้าไปในรูปแบบ โดยอยู่ภายในเครื่องหมาย double quotation marks

DD "of" MONTH	12 of OCTOBER

การสะกดคำต่อท้ายของตัวเลข

ddspth	fourteenth

USING THE TO_CHAR FUNCTION WITH DATES

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	King	17 June 1987
2	Kochhar	21 September 1989
3	De Haan	13 January 1993
4	Hunold	3 January 1990
5	Ernst	21 May 1991
6	Lorentz	7 February 1999
7	Mourgos	16 November 1999
8	Rajs	17 October 1995
9	Davies	29 January 1997
10	Matos	15 March 1998

19	Higgins	7 June 1994
20	Gietz	7 June 1994

USING THE TO_CHAR FUNCTION WITH NUMBERS

TO_CHAR(*number*,'*format_model*')

สามารถจัดการรูปแบบของตัวเลข โดยสามารถใช้คำสั่ง TO_CHAR function

Element	Result
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as a thousands indicator

USING THE TO_CHAR FUNCTION WITH NUMBERS

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

	SALARY
1	$6,000.00

to_char(1210.73, '9999.9') would return '1210.7'

to_char(1210.73, '9,999.99') would return '1,210.73'

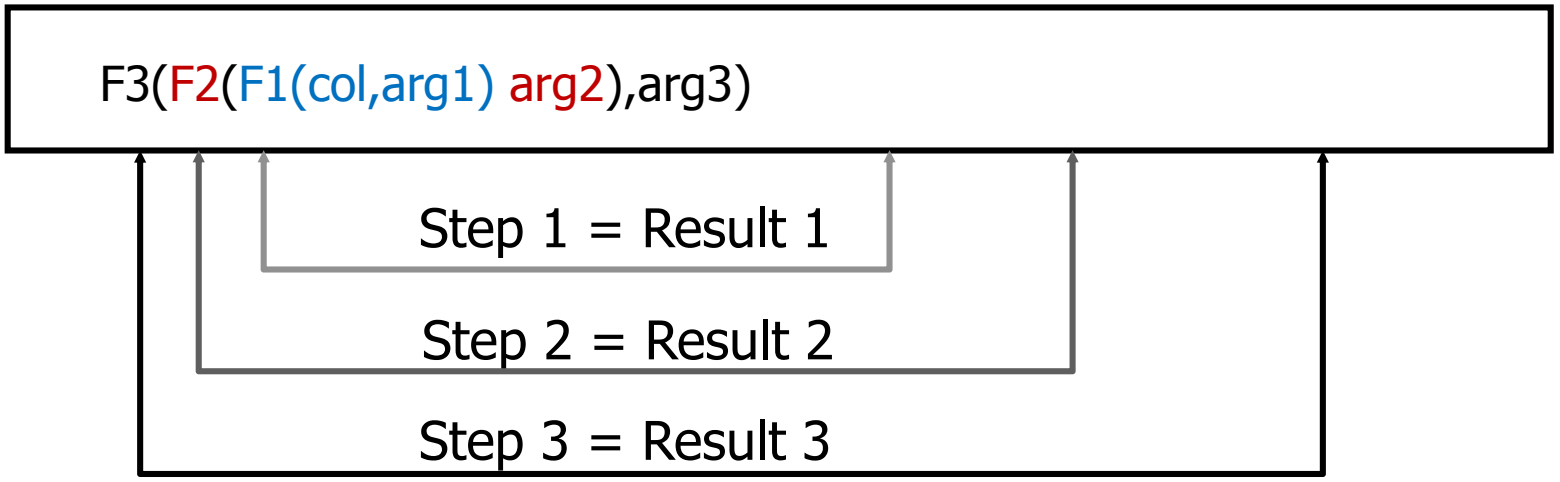
to_char(1210.73, '$9,999.00') would return '$1,210.73'

to_char(21, '000099') would return '000021'

NESTING FUNCTIONS

ฟังก์ชันสามารถที่จะเรียกใช้ ซ้อน ๆ กันได้

การประมวลผลจะเรียกจากระดับข้างในก่อน



NESTING FUNCTIONS

```
SELECT last_name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

NVL FUNCTION

การแปลงค่าว่างให้กลายเป็นค่าตามที่ต้องการ

- ชนิดของข้อมูลที่สามารถใช้ในฟังก์ชันนี้ได้คือ date, character, และ number
- ตัวอย่างการเรียกใช้ โดยที่ชนิดของข้อมูลที่เปลี่ยนต้องตรงกับข้อมูลเดิม
  - NVL(commission_pct,0)
  - NVL(hire_date,'01-JAN-08')
  - NVL(job_id,'No Job Yet')

USING THE NVL FUNCTION

```
SELECT last_name, salary, NVL(commission_pct, 0),
    (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200
11	Vargas	2500	0	30000
12	Zlotkey	10500	0.2	151200

...

CONDITIONAL EXPRESSIONS

ในคำสั่ง SQL สามารถกำหนดเงื่อนไขในการแสดงผล โดยใช้คำสั่ง IF-THEN-ELSE

ประกอบด้วย 2 วิธีคือ

- CASE expression
- DECODE function

SIMPLE CASE

Simple CASE	
Syntax	SELECT Field, Field, CASE Field | Expression WHEN value THEN result WHEN value THEN result ELSE result END As alias FROM Table
Examples	Select Firstname, Case Gender When 'F' Then 'woman' When 'M' Then 'man' End As Sex From Members

SIMPLE CASE EXAMPLE

```
Select Artistname, Region,
        Case Region
        When 'NC' Then 'South'
        When 'VA' Then 'South'
        When 'IL' Then 'Midwest'
        When 'VT' Then 'New England'
        Else 'Somewhere Else'
        End As Area
From Artists
```

SIMPLE CASE EXAMPLE

Artistname	Region	Area
The Neurotics	NC	South
Louis Holiday	IL	Midwest
Word	IN	Somewhere Else
Sonata	VA	South
The Bullets	TX	Somewhere Else
Jose MacArthur	CA	Somewhere Else
Confused	GA	Somewhere Else
The Kicks	NY	Somewhere Else
Today	ONT	Somewhere Else
21 West Elm	VT	New England
Highlander	OH	Somewhere Else

SWITCHED CASE

Switched CASE	
Syntax	SELECT Field, Field, CASE WHEN Field | Expression comparison Value | Field | Expression THEN result WHEN Field | Expression comparison Value | Field | Expression THEN result ELSE result END As alias FROM Table
	Select TrackNum, TrackTitle, LengthSeconds, Case When TrackNum=1 And LengthSeconds<240 Then 'Short 1st' When TrackNum=1 And LengthSeconds>480 Then 'Long 1st' Else 'Another Track' End as Eval From Tracks Where TrackNum<3

SWITCHED CASE EXAMPLE

```
Select TrackNum, TrackTitle, LengthSeconds,
   Case
      When TrackNum=1 And LengthSeconds<240 Then 'Short 1st Track'
      When TrackNum=1 And LengthSeconds>480 Then 'Long 1st Track'
      Else 'Another Track'
   End as Eval
From Tracks Where TrackNum<3
```

SWITCHED CASE EXAMPLE

TrackNum	TrackTitle	LengthSeconds	Eval
1	Bob's Dream	185	Short 1st Track
2	My Wizard	233	Another Track
1	Fat Cheeks	352	Another Track
1	Hottie	233	Short 1st Track
2	GoodtimeMarch	293	Another Track
2	Rocky and Natasha	283	Another Track
1	Violin Sonata No.1 in D Major	511	Long 1st Track
2	Violin Sonata No. 2 in A Major	438	Another Track
1	Song 1	285	Another Track
2	Song 2	272	Another Track
1	I Don't Know	201	Short 1st Track
2	What's the Day	332	Another Track

DECODE FUNCTION

การใช้คำสั่ง DECODE เพื่อกำหนดเงื่อนไขได้เช่นเดียวกับคำสั่ง CASE

```
DECODE(col|expression, search1, result1
                      [, search2, result2,...,]
                      [, default])
```

USING THE DECODE FUNCTION

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

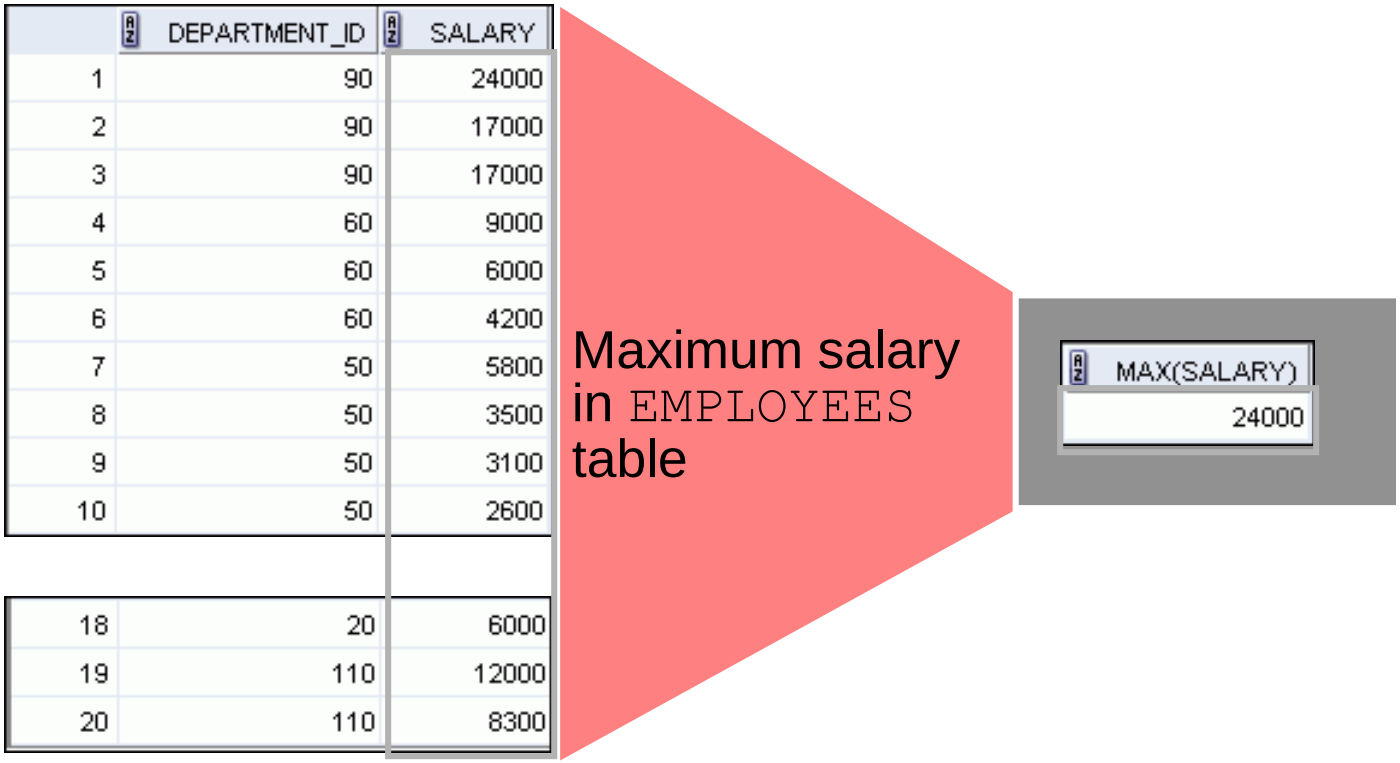
USING THE DECODE FUNCTION

ตัวอย่างการแสดงค่าภาษี จากเงื่อนไขที่กำหนดจากเงินเดือน เฉพาะแผนกหมายเลข 80

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                    0, 0.00,
                    1, 0.09,
                    2, 0.20,
                    3, 0.30,
                    4, 0.40,
                    5, 0.42,
                    6, 0.44,
                       0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

WHAT ARE GROUP FUNCTIONS?

ฟังก์ชันกลุ่มคือ ฟังก์ชันที่ได้จากการประมวลผลข้อมูลหลาย ๆ แถว



TYPES OF GROUP FUNCTIONS

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

GROUP FUNCTIONS: SYNTAX

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

USING THE AVG AND SUM FUNCTIONS

AVG ค่าเฉลี่ย SUM ค่าผลรวม ซึ่งจะใช้กับข้อมูลตัวเลขเท่านั้น

```
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

USING THE MIN AND MAX FUNCTIONS

MIN ค่าน้อยที่สุด MAX ค่ามากที่สุด สามารถใช้ได้กับ ตัวเลข ตัวอักษร และ วันที่ ได้

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

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

USING THE COUNT FUNCTION

COUNT(*) นับจำนวนแถวข้อมูลที่ตรงเงื่อนไข ที่กำหนด

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

1

	COUNT(*)
1	5

COUNT(*expr*) **นับจำนวนแถวที่ไม่เป็นค่าว่าง** สำหรับคอลัมน์ *expr*

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

2

	COUNT(COMMISSION_PCT)
1	3

USING THE DISTINCT KEYWORD

COUNT(DISTINCT expr) นับจำนวนแถวที่**ไม่เป็นค่าว่าง** โดย**ไม่รวมแถวที่ข้อมูลซ้ำของคอลัมน์** *expr*

ตัวอย่างการนับจำนวนแผนกของพนักงาน ในตาราง EMPLOYEES โดยหากแผนกซ้ำกันจะนับเป็น 1

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

	COUNT(DISTINCTDEPARTMENT_ID)
1	7

GROUP FUNCTIONS AND NULL VALUES

การใช้ฟังก์ชันกลุ่มโดยไม่มีการประมวลผลค่าว่าง

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

1

	AVG(COMMISSION_PCT)
1	0.2125

หากต้องการให้ประมวลผลค่าว่างด้วยสามารถใช้ฟังก์ชัน NVL functions

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

2

	AVG(NVL(COMMISSION_PCT,0))
1	0.0425

CREATING GROUPS OF DATA

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	5800
5	50	2500
6	50	2600
7	50	3100
8	50	3500
9	60	4200
10	60	6000
11	60	9000
12	80	11000
13	80	10500
14	80	8600
19	110	12000
20	(null)	7000

4400

9500

3500

6400

10033

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

CREATING GROUPS OF DATA: GROUP BY CLAUSE SYNTAX

สามารถใช้ฟังก์ชันแบบกลุ่ม เพื่อประมวลผลข้อมูลเป็นกลุ่มย่อย ๆ ได้ โดยระบุคำสั่ง GROUP BY clause.

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

USING THE GROUP BY CLAUSE

คอลัมน์ทั้งหมดที่ถูกเลือกให้แสดงผลในคำสั่ง SELECT ไม่รวมที่ถูกใช้ใน ฟังก์ชันแบบกลุ่ม จะต้องอยู่ในคำสั่ง GROUP BY clause

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

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	90	19333.33333333333...
3	20	9500
4	110	10150
5	50	3500
6	80	10033.33333333333...
7	60	6400
8	10	4400

USING THE GROUP BY CLAUSE

การใช้คำสั่ง GROUP BY โดยไม่มีในคอลัมน์ที่เลือกแต่อาจจะทำให้สับสนได้ว่าผลลัพธ์ที่ได้ของแต่ละแถวเป็นของแผนกอะไร

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

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

GROUPING BY MORE THAN ONE COLUMN

EMPLOYEES

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

19	110	AC_MGR	12000
20	(null)	SA_REP	7000

Add the salaries in the `EMPLOYEES` table for each job, grouped by department.

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

USING THE GROUP BY CLAUSE ON MULTIPLE COLUMNS

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

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

ILLEGAL QUERIES USING GROUP FUNCTIONS

คอลัมน์ที่เลือกไม่ได้ระบุในส่วนของ GROUP BY clause

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

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

A GROUP BY clause must be added to count the last names for each 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"
```

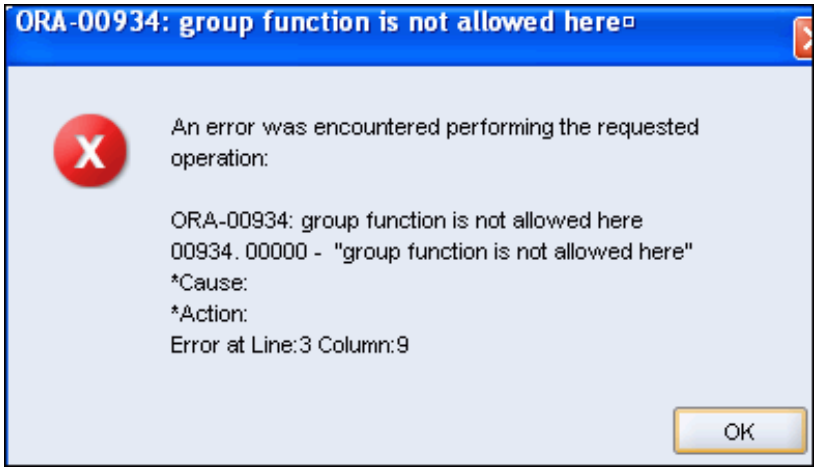
Either add job_id in the GROUP BY or remove the job_id column from the SELECT list.

ILLEGAL QUERIES USING GROUP FUNCTIONS

ไม่สามารถใช้เงื่อนไขฟังก์ชันกลุ่ม ในส่วนของคำสั่ง WHERE ได้

อนุญาตให้ใช้คำสั่ง HAVING clause ในการกำหนดเงื่อนไขของฟังก์ชันกลุ่ม เท่านั้น

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



Cannot use the WHERE clause to restrict groups

RESTRICTING GROUP RESULTS

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	5800
5	50	2500
6	50	2600
7	50	3100
8	50	3500
9	60	4200
10	60	6000
11	60	9000
12	80	11000
13	80	10500
14	80	8600
18	110	8300
19	110	12000
20	(null)	7000

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

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

RESTRICTING GROUP RESULTS WITH THE HAVING CLAUSE

กรณีที่ต้องการใช้คำสั่ง HAVING clause สามารถใช้กำหนดเงื่อนไขได้ดังนี้

1. แถวที่ถูกจัดกลุ่ม
2. ฟังก์ชันกลุ่ม
3. กลุ่มข้อมูลที่ตรงกับเงื่อนไขจะถูกแสดงผล

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

USING THE HAVING CLAUSE

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

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

USING THE HAVING CLAUSE

```
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

NESTING GROUP FUNCTIONS

แสดงค่าสูงสุดของค่าเฉลี่ยของเงินเดือนในแต่ละแผนก

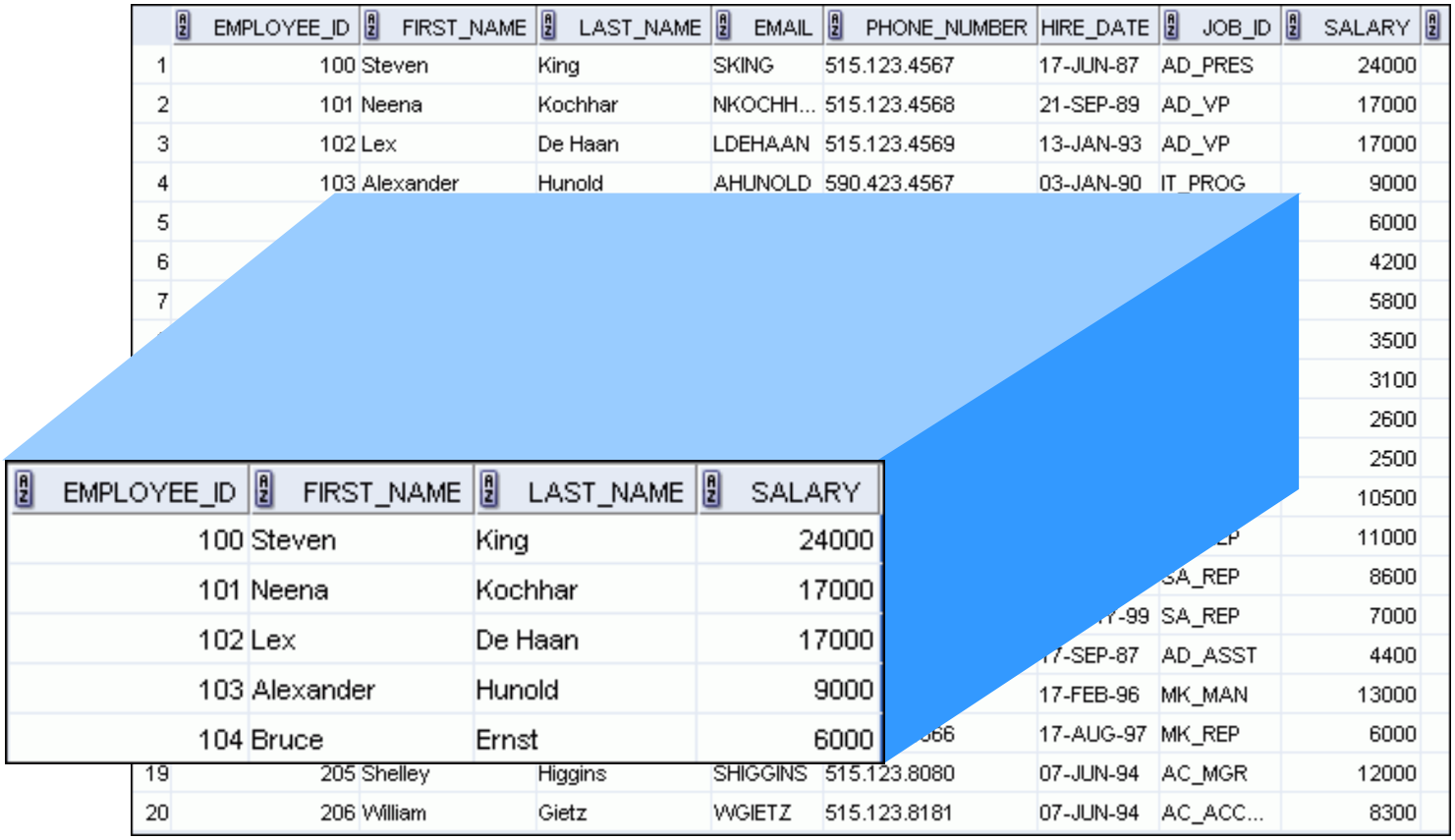
```
SELECT  MAX(AVG(salary))
FROM    employees
GROUP BY department_id;
```

	MAX(AVG(SALARY))
1	19333.333333333333333333333333333333333

VIEW

Object	Description
Table	Basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

WHAT IS A VIEW?



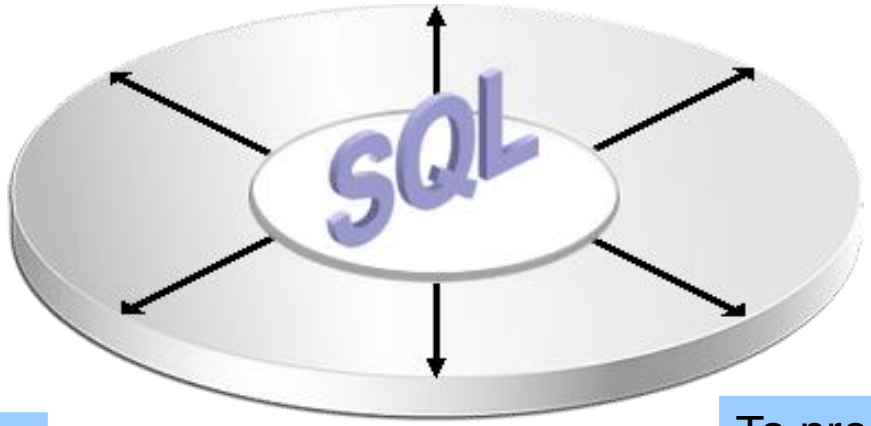
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
2	101	Neena	Kochhar	NKOCHH...	515.123.4568	21-SEP-89	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
5								6000
6								4200
7								5800
								3500
								3100
								2600
								2500
								10500
								11000
							SA_REP	8600
						-99	SA_REP	7000
						17-SEP-87	AD_ASST	4400
						17-FEB-96	MK_MAN	13000
					566	17-AUG-97	MK_REP	6000
19	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
20	206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACC...	8300

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
103	Alexander	Hunold	9000
104	Bruce	Ernst	6000

ADVANTAGES OF VIEWS

To restrict data access

To make complex queries easy



To provide data independence

To present different views of the same data

SIMPLE VIEWS AND COMPLEX VIEWS

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain functions	No	Yes
Contain groups of data	No	Yes
DML operations through a view	Yes	Not always

CREATING A VIEW

เป็นการนำชุดคำสั่ง SQL ในการสร้างวิว

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

NOFORCE Creates the view only if the base tables exist (This is the default.)

สามารถใช้คำสั่ง SQL ทั้งแบบปกติและแบบซับซ้อน

CREATING A VIEW

ตัวอย่างการสร้าง EMPVU80 view ซึ่งประกอบด้วยข้อมูลพนักงานในแผนกหมายเลข 80

```
CREATE VIEW      empvu80
 AS SELECT  employee_id, last_name, salary
    FROM    employees
    WHERE   department_id = 80;
CREATE VIEW succeeded.
```

การเรียกดูโครงสร้างของ วิว ใน SQL*Plus โดยใช้คำสั่ง DESCRIBE

```
DESCRIBE empvu80
```

CREATING A VIEW

การสร้างวิวโดยใช้นามแฝงของคอลัมน์

```
CREATE VIEW      salvu50
 AS SELECT  employee_id ID_NUMBER, last_name NAME,
           salary*12 ANN_SALARY
    FROM    employees
    WHERE   department_id = 50;
```

```
CREATE VIEW succeeded.
```

RETRIEVING DATA FROM A VIEW

```
SELECT *
FROM  salvu50;
```

	ID_NUMBER	NAME	ANN_SALARY
1	124	Mourgos	69600
2	141	Rajs	42000
3	142	Davies	37200
4	143	Matos	31200
5	144	Vargas	30000

MODIFYING A VIEW

การแก้ไขวิว EMPVU80 โดยใช้คำสั่ง CREATE OR REPLACE VIEW

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' '
          || last_name, salary, department_id
   FROM    employees
   WHERE   department_id = 80;
```

```
CREATE OR REPLACE VIEW succeeded.
```

การใช้นามแฝงของคอลัมน์ จะเรียงลำดับตามลำดับของคอลัมน์ที่ถูกเลือกในคำสั่ง SQL ของคิวรี

CREATING A COMPLEX VIEW

การสร้างวิวแบบซับซ้อน โดยประกอบด้วยฟังก์ชันกลุ่ม

```
CREATE OR REPLACE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT   d.department_name, MIN(e.salary),
           MAX(e.salary),AVG(e.salary)
   FROM     employees e JOIN departments d
   ON       (e.department_id = d.department_id)
   GROUP BY d.department_name;
```

```
CREATE OR REPLACE VIEW succeeded.
```

DENYING DML OPERATIONS

ข้อควรระวังคือจะต้องแน่ใจว่าไม่มีการใช้คำสั่ง WITH READ ONLY ในส่วนของการสร้างวิว

ซึ่งจะทำให้ผลลัพธ์ของวิวใน Oracle server เกิดข้อผิดพลาดได้

DENYING DML OPERATIONS

```
CREATE OR REPLACE VIEW empvu10
    (employee_number, employee_name, job_title)
AS SELECT  employee_id, last_name, job_id
   FROM     employees
   WHERE    department_id = 10
   WITH READ ONLY ;
```

```
CREATE OR REPLACE VIEW succeeded.
```

REMOVING A VIEW

การลบวิว ซึ่งจะไม่กระทบกับข้อมูลที่อยู่ในตาราง สามารถลบวิวได้ เพราะวิวเป็นเพียงส่วนที่ใช้ในการเลือกข้อมูลจากตารางจริง มาเท่านั้น

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
DROP VIEW empvu80 succeeded.
```

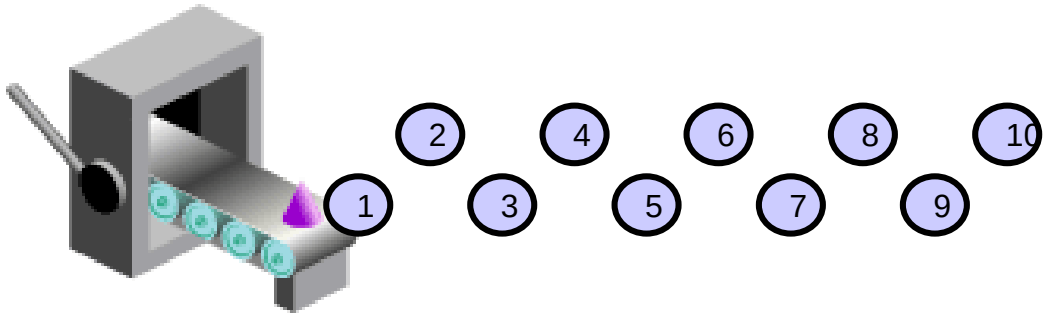
SEQUENCES

Object	Description
Table	Basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

SEQUENCES

A sequence:

- สามารถสร้างลำดับตัวเลขอัตโนมัติ
- อยู่ในรูปแบบที่แบ่งปันข้อมูล
- สามารถนำไปสร้างเป็น primary key
- สามารถใช้แทนรหัสในโปรแกรมได้
- เพิ่มความเร็วในการจัดการหน่วยความจำได้



CREATE SEQUENCE STATEMENT : SYNTAX

รูปแบบการสร้างตัวเลขเรียงลำดับอัตโนมัติ

```
CREATE SEQUENCE sequence
      [INCREMENT BY n]
      [START WITH n]
      [{MAXVALUE n | NOMAXVALUE}]
      [{MINVALUE n | NOMINVALUE}]
      [{CYCLE | NOCYCLE}]
      [{CACHE n | NOCACHE}];
```

CREATING A SEQUENCE

ตัวอย่างการสร้าง ตั้งชื่อว่า DEPT_DEPTID_SEQ เพื่อใช้เป็นคีย์หลักของตาราง DEPARTMENTS

ไม่อนุญาตให้ใช้ CYCLE

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 120
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
```

```
CREATE SEQUENCE succeeded.
```

NEXTVAL AND CURRVAL PSEUDOCOLUMNS

NEXTVAL คืนค่าตัวเลขต่อไป ของลำดับ ซึ่งจะเป็นค่าที่ไม่ซ้ำ สำหรับ ทุกๆ กลุ่มผู้ใช้

CURRVAL คืนค่าลำดับปัจจุบัน

NEXTVAL จะต้องถูกกำหนดค่าก่อน ที่ค่าของ CURRVAL จะถูกกำหนด

USING A SEQUENCE

ตัวอย่างการเพิ่มแถวข้อมูลใหม่ โดยใช้ตัวเลขลำดับ

```
INSERT INTO departments(department_id,
            department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
1 rows inserted
```

การเรียกดูค่าปัจจุบันของ ตัวเลขลำดับ DEPT_DEPTID_SEQ

```
SELECT      dept_deptid_seq.CURRVAL
FROM        dual;
```

CACHING SEQUENCE VALUES

การจัดการค่าแบบเรียงลำดับในหน่วยความจำจะทำให้การเข้าถึงข้อมูลได้เร็วขึ้น

ช่องว่างระหว่างตัวเลขสามารถเกิดขึ้นได้จาก

- rollback
- ระบบล้มเหลว
- ถูกนำไปใช้ในตารางอื่น

MODIFYING A SEQUENCE

การแก้ไขค่าที่เพิ่มขึ้น ค่าสูงสุด ค่าต่ำสุด การวนค่า และ การจัดการหน่วยความจำ

```
ALTER SEQUENCE dept_deptid_seq
            INCREMENT BY 20
            MAXVALUE 999999
            NOCACHE
            NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq succeeded.
```

GUIDELINES FOR MODIFYING A SEQUENCE

จะทำการแก้ไขได้จะต้องมีการกำหนดสิทธิ์ ให้สามารถใช้คำสั่ง ALTER sequence ได้

ตัวเลขถัดไปเท่านั้นที่จะมีผล หลังจากใช้คำสั่งแก้ไข ไม่สามารถจัดการกับค่าที่ผ่านมาแล้วได้

หากต้องการเริ่มต้นค่าใหม่ จะต้องทำการลบออกก่อน และ สร้างใหม่

การลบ sequence โดยใช้คำสั่ง DROP

```
DROP SEQUENCE dept_deptid_seq;
```

```
DROP SEQUENCE dept_deptid_seq succeeded.
```

SYNONYMS

Object	Description
Table	Basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

CREATING A SYNONYM FOR AN OBJECT

การเข้าถึงวัตถุต่าง ๆ ใน Oracle สามารถกำหนดให้เรียกใช้ได้ง่ายขึ้นโดยใช้นามแฝง หรือ ชื่อเหมือนของวัตถุนั้น ๆ

- สร้างการอ้างอิง และ การเข้าถึงตารางให้ง่ายขึ้นสำหรับบุคคลอื่น ๆ
- เป็นการตั้งชื่อให้สั้นลง

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

CREATING AND REMOVING SYNONYMS

การสร้างชื่อวิว DEPT_SUM_VU ให้สั้นลง

```
CREATE SYNONYM  d_sum
FOR  dept_sum_vu;
```

```
CREATE SYNONYM succeeded.
```

การลบ synonym

```
DROP SYNONYM d_sum;
```

```
DROP SYNONYM d_sum succeeded.
```