Chapter 10 : ภาษามาตรฐานสำหรับนิยามข้อมูลและการใช้ข้อมูล

SQL - Structure Query Language

ระบบฐานข้อมูล

วัตถุประสงค์

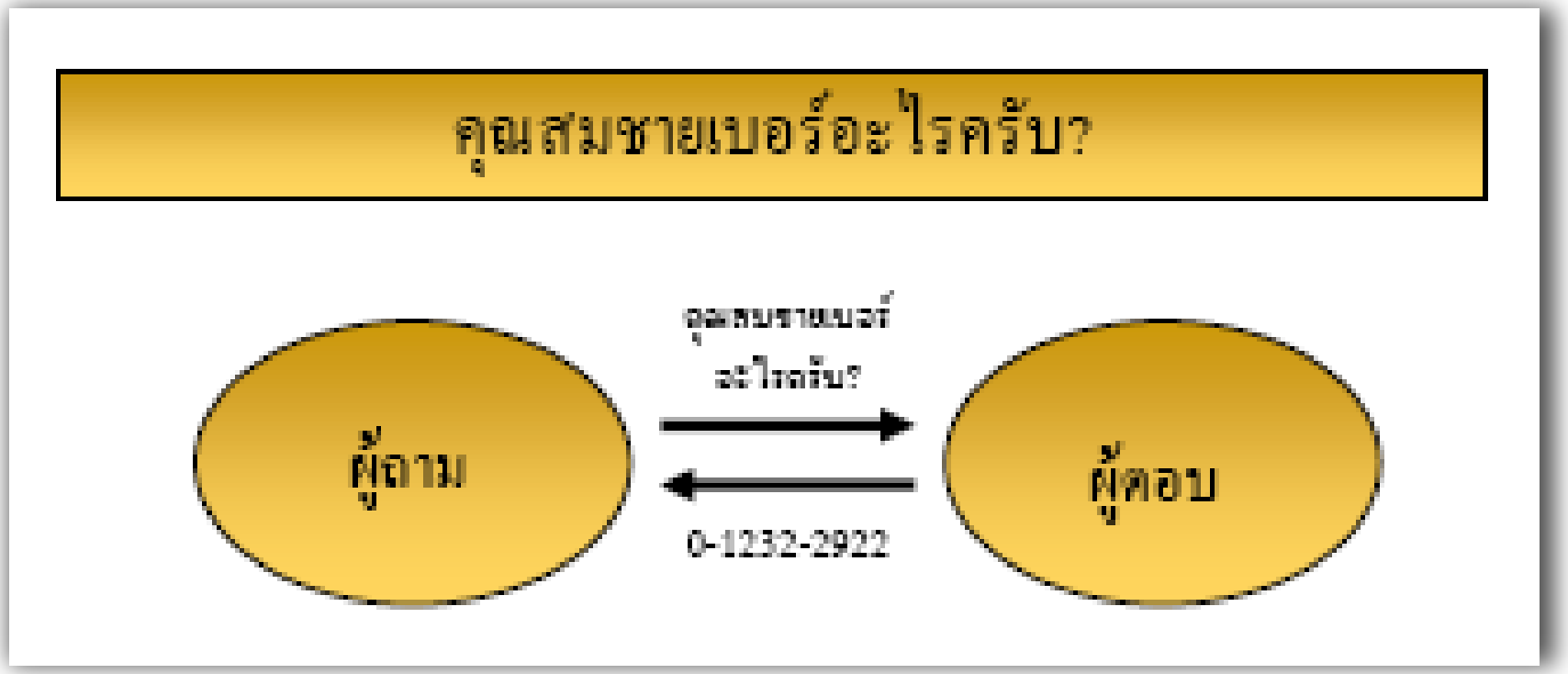
- สามารถอธิบายแนวคิดและการใช้ SQL ได้
- นำ SQL ไปใช้เพื่อให้เกิดประโยชน์ได้

ภาษา SQL

- สามารถสร้างและปฏิบัติการกับฐานข้อมูลเชิงสัมพันธ์โดยเฉพาะ
- Structured Query Language
- สำหรับใช้กับ Relational Database
- อยู่ในรูปแบบของภาษาอังกฤษ

ภาษา SQL

- Structured Query Language



ภาษา SQL

บอกหน่อยสิครับ/คะว่าใครบ้างในบริษัทที่มีเงินเดือนระหว่าง 10,000 ถึง 20,000 บาท

บอกหน่อยสิคะว่าใครเป็นพนักงานที่มีเงินเดือนระหว่าง 10,000 ถึง 20,000 บาท

แสดงรายชื่อพนักงานที่มีรายได้ระหว่าง 10,000 ถึง 20,000 บาทต่อเดือน

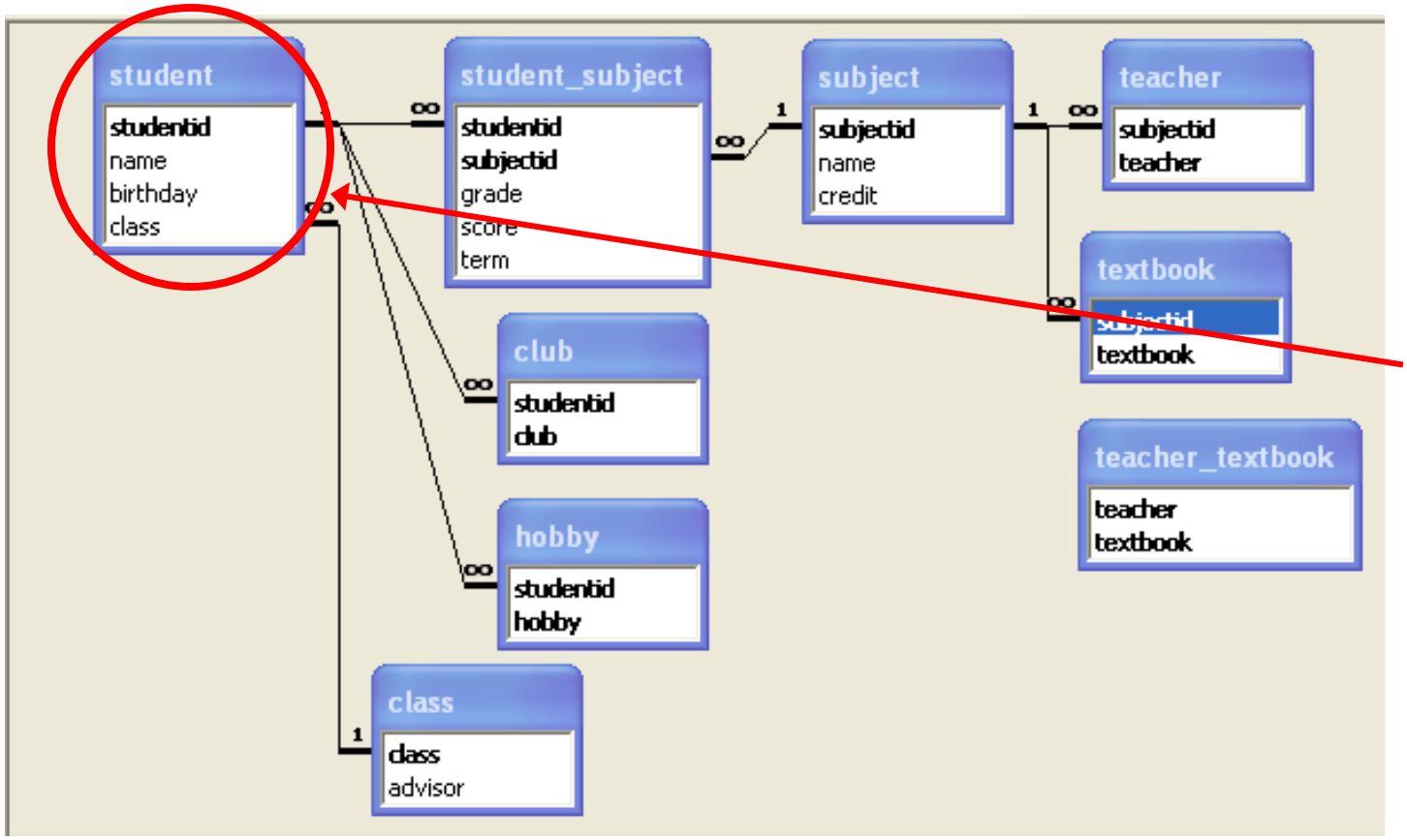
ภาษา SQL

แสดงรายชื่อพนักงานที่มีรายได้ระหว่าง 10,000 ถึง 20,000 บาท

```
SELECT emp_name FROM employee
WHERE emp_salary > 10,000 AND emp_salary < 20,000
```

ภาษา SQL

- สำหรับใช้กับ Relational Database



select
name
from
student

ประเภทของคำสั่งของภาษา SQL

1. ภาษาสำหรับการนิยามข้อมูล (Data Definition Language : DDL)
2. ภาษาสำหรับการจัดการข้อมูล (Data Manipulation Language : DML)
3. ภาษาควบคุม (Data Control Language : DCL)

1. Data Definition Language : DDL

- เป็นกลุ่มคำสั่งที่ใช้กระทำกับโครงสร้างของฐานข้อมูล
- ตัวอย่างคำสั่ง เช่น
  - CREATE ใช้สร้างตาราง
  - ALTER ใช้แก้ไข/เปลี่ยนแปลงตาราง
  - DROP ใช้สำหรับลบตาราง

ชนิดของข้อมูลที่ใช้ในภาษา SQL

- จำนวนเลข (Number)
  - เลขจำนวนจริง (Number) เก็บเลขที่เป็นจำนวนเต็มหรือเป็นทศนิยม รูปแบบที่ใช้คือ NUMBER
  - เลขจำนวนเต็ม (Integer) รองรับเลขได้ 11 หลัก รูปแบบที่ใช้คือ INTEGER หรือ INT
  - เลขจำนวนเต็มขนาดเล็ก (Small integer) รองรับเลขได้ 6 หลัก รูปแบบที่ใช้คือ SMALLINT
  - เลขทศนิยม (Decimal) รูปแบบที่ใช้คือ DECIMAL

ชนิดของข้อมูลที่ใช้ในภาษา SQL

- ตัวหนังสือ
  - ความยาวคงที่ (Fixed-length Character) เก็บข้อมูลได้สูงสุด 255 ตัวอักษร รูปแบบที่ใช้คือ CHAR
  - ความยาวไม่คงที่ เก็บข้อมูลได้สูงสุด 255 ตัวอักษร (Variable-length Character) รูปแบบที่ใช้คือ VARCHAR
- ข้อมูลในลักษณะอื่น ๆ
  - วันที่และเวลา (Date/Time) รูปแบบที่ใช้คือ DATE

ชนิดของข้อมูลใน Microsoft Access

- ในโปรแกรม Microsoft Access สามารถใช้ชนิดข้อมูลตามมาตรฐานของ SQL ได้
- หรือสามารถใช้ชนิดข้อมูลของ Microsoft Access ที่ระบุขึ้นเองก็ได้ อาทิเช่น
  - Text ตัวอักษร
  - Number ตัวเลข
  - Date/Time วันที่และเวลา

ชนิดของข้อมูลใน Oracle

- สิ่งที่กำหนดในโครงสร้างเทเบิล ได้แก่ คอลัมน์ ซึ่งจะต้องระบุประเภทข้อมูลที่จัดเก็บตัวอักษร จำนวนเลข วันที่ ออบเจ็คต์หรือไบนารี (Binary) โดยแบ่งประเภทของข้อมูลได้ดังต่อไปนี้

กลุ่มประเภท	ประเภทข้อมูล	ความหมายของประเภท
ตัวอักษร (Character)	CHAR(n)	จัดเก็บตัวอักษรหรือตัวเลขที่ไม่ได้นำไปคำนวน โดยมีการกำหนดความยาวแบบคงที่ และจัดเก็บตามความยาวที่กำหนดไว้ และจัดเก็บได้สูงสุด 2,000 ไบต์
	VARCHAR2(n)	จัดเก็บตัวอักษรหรือตัวเลขที่ไม่ได้นำไปคำนวน โดยความยาวที่กำหนดควเป็นแบบยืดหยุ่น เก็บตามความเป็นจริง แต่ไม่เกินที่หนด และจัดเก็บได้สูงสุด 4,000 ไบต์

ชนิดของข้อมูลใน Oracle

<table>
<tr><th>กลุ่มประเภท</th><th>ประเภทข้อมูล</th><th colspan="3">ความหมายของประเภท</th></tr>
<tr><td rowspan="8">ตัวเลข
(Number)</td><td rowspan="8">NUMBER(p,s)</td><td colspan="3">จัดเก็บตัวเลขที่สามารถนำไปคำนวณได้ โดยกำหนดได้ทั้งจำนวนเต็มและทศนิยม โดยที่ P คือความยาวทั้งหมด (รวมความยาว s ด้วย) s คือความยาวของทศนิยม เช่น</td></tr>
<tr><td>ข้อมูลจริง</td><td>ประเภทข้อมูล</td><td>ค่าที่เก็บ</td></tr>
<tr><td>123.89</td><td>NUMBER</td><td>123.89</td></tr>
<tr><td>123.89</td><td>NUMBER(3)</td><td>124</td></tr>
<tr><td>123.89</td><td>NUMBER(5,2)</td><td>123.89</td></tr>
<tr><td>123.89</td><td>NUMBER(6,-2)</td><td>100</td></tr>
<tr><td>0.01234</td><td>NUMBER(4,5)</td><td>0.01234</td></tr>
<tr><td>0.012345</td><td>NUMBER(4,5)</td><td>0.01235</td></tr>
</table>

ชนิดของข้อมูลใน Oracle

กลุ่มประเภท	ประเภทข้อมูล	ความหมายของประเภท
วันที่ (Date)	DATE	จัดเก็บวันที่และเวลาได้แก้ ศตวรรษ ปี เดือน วัน ชั่วโมง นาที และวินาที ซึ่งสามารถแสดงผลได้ทั้งตัวอักษรและตัวเลข โดยใช้ร่วมกับ ฟังชั่น TO_CHAR
LARGE OBJECT	BLOP	จัดเก็บแบบไบนารี ได้แก่ รูปภาพ และเสียง โดยสามารถจัดเก็บได้สูงสุด 4 กิกะไบต์
	BFILE	จัดเก็บแบบไฟล์ไบนารี (Binary File) ได้แก่ ไดเรกทอรี (Directory) และชื่อไฟล์ (Filename)
	CLOB	จัดเก็บแบบตัวอักษรที่มีความยาวมาก ๆ โดยสามารถจัดเก็บได้สูงสุด 4 กิกะไบต์

คำสั่ง SQL : DDL

- **CREATE TABLE : สร้างตาราง**
  - **รูปแบบการใช้คำสั่ง**

CREATE TABLE *ชื่อตาราง*

(

ชื่อคอลัมน์1 ***ประเภทข้อมูล*** Constraint,

ชื่อคอลัมน์2 ***ประเภทข้อมูล*** Constraint,

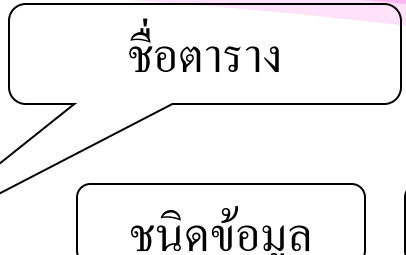
PRIMARY KEY (ชื่อคอลัมน์) ,

FOREIGN KEY (ชื่อคอลัมน์) REFERENCES ชื่อตาราง,

);

คำสั่ง SQL : DDL

- CREATE TABLE
  - ตัวอย่าง




```
CREATE TABLE  EMPLOYEE (
    ID              CHAR(5)         NOT NULL ,
    NAME            VARCHAR(35)     NOT NULL ,
    ADDRESS         VARCHAR(15)     NOT NULL ,
    PHONE           CHAR(8)         NOT NULL ,
    E-MAIL          CHAR(1)         NOT NULL ,
    PRIMARY KEY (ID)
);
```

คำสั่ง SQL : DDL

```
CREATE TABLE  PRODUCT (
   P_CODE          VARCHAR(10)     NOT NULL  ,
   P_DESCRIPT      VARCHAR(35)     NOT NULL ,
   P_INDATE        DATE            NOT NULL ,
   P_ONHAND        SMALLINT        NOT NULL ,
   P_MIN           SMALLINT        NOT NULL ,
   P_PRICE         NUMBER          NOT NULL ,
   P_DISCOUNT      NUMBER          NOT NULL ,
   ID              CHAR(5) ,
   PRIMARY KEY (P_CODE) ,
   FOREIGN KEY (ID) REFERENCES EMPLOYEE(ID)
);
```

ให้ ID เป็น FK

คำสั่ง SQL : DDL

ALTER TABLE : เปลี่ยนแปลงตาราง

- รูปแบบการใช้คำสั่ง

  ALTER TABLE *ชื่อตาราง*

  (คำสั่งการเปลี่ยนแปลง *ชื่อคอลัมน์ ชนิดข้อมูล*);

  - คำสั่งการเปลี่ยนแปลง เช่น
    - ADD เพิ่มคอลัมน์
    - ALTER เปลี่ยนแปลงชนิดข้อมูล
    - DROP ลบคอลัมน์

คำสั่ง SQL : DDL

- **ALTER TABLE : เปลี่ยนแปลงตาราง**
  - **ตัวอย่าง**
    - **ALTER TABLE PRODUCT**

      ADD SALECODE VARCHAR(10);
    - **ALTER TABLE PRODUCT**

      ALTER Column SALECODE VARCHAR(50);
    - **ALTER TABLE PRODUCT**

      DROP SALECODE;

คำสั่ง SQL : DDL

DROP TABLE : ลบตาราง

- รูปแบบการใช้คำสั่ง

  DROP TABLE *ชื่อตาราง*;

- ตัวอย่าง

  DROP TABLE PRODUCT;

  DROP TABLE EMPLOYEE;

  DROP TABLE PROJECT;

2. Data Manipulation Language : DML

- เป็นกลุ่มคำสั่งที่กระทำกับข้อมูลในฐานข้อมูล
- ตัวอย่างคำสั่ง เช่น
  - SELECT ใช้เรียกข้อมูลในตารางมาแสดงผล
  - INSERT ใช้เพิ่มข้อมูลเข้าไปในตาราง
  - UPDATE ใช้แก้ไขข้อมูลในตาราง
  - DELETE ใช้ลบข้อมูลในตาราง

คำสั่ง SQL : DML

SELECT <การเรียกดูข้อมูล>

- การเรียกดูข้อมูลในคอลัมน์ที่ต้องการ

  •รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง ;*

คำสั่ง SQL : DML

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

- ตัวอย่าง แสดงชื่อและนามสกุลของพนักงาน

```
SELECT First_name, Last_name FROM Employees ;
```

First_name	Last_name
สมชาย	ชาตรี
สมหญิง	งามแท้

คำสั่ง SQL : DML

SELECT <การเรียกดูข้อมูล>

- การเรียกดูข้อมูลในทุก ๆ คอลัมน์

  •รูปแบบ

```
SELECT *
FROM ชื่อตาราง ;
```

คำสั่ง SQL : DML

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

❖ ตัวอย่าง แสดงข้อมูลทั้งหมดของพนักงาน

```
SELECT * FROM Employees ;
```

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

คำสั่ง SQL : DML

- **SELECT <การเรียกดูข้อมูลเฉพาะบางแถวที่ตรงตามเงื่อนไข>**
  - **การเรียกดูข้อมูลในคอลัมน์ที่ต้องการ**
    - **รูปแบบ**

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *เงื่อนไข* ;

โดยภายในเงื่อนไขประกอบด้วย

เงื่อนไข => คอลัมน์ที่เป็นเงื่อนไข Operator Value

คำสั่ง SQL : DML

Employees

Emp_ID	First_name	Last_name	Address
001	สมชาย	ชาตรี	เชียงใหม่
002	สมหญิง	งามแท้	อุตรดิตถ์

❖ **ตัวอย่าง แสดงชื่อ นามสกุล และที่อยู่ของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่อยู่จังหวัดอุตรดิตถ์**

```
SELECT  First_name, Last_name, Address
FROM Employees
WHERE Address = 'อุตรดิตถ์';
```

Address คอลัมน์ที่เป็นเงื่อนไข

= คือ operator

อุตรดิตถ์ คือ value

First_name	Last_name	Address
สมหญิง	งามแท้	อุตรดิตถ์

Operator (ตัวปฏิบัติการ) ที่ใช้ใน Where

- Operator ที่สามารถใช้กำหนดเงื่อนไขใน where มีดังนี้
- ตัวเปรียบเทียบ (Comparison operators)
  - ได้แก่เครื่องหมาย =, <, >, <=, >=, <>
- ตัวปฏิบัติการทางด้านลอจิก (Logical operators)
  - AND
  - OR

  } เชื่อมต่อเงื่อนไขในกรณีที่มีมากกว่า 1 เงื่อนไข

  - NOT

Operators ในการเปรียบเทียบ

- ตัว Operators ในการเปรียบเทียบ
  - = เท่ากับ
  - < น้อยกว่า
  - <= น้อยกว่าหรือเท่ากับ
  - > มากกว่า
  - >= มากกว่าหรือเท่ากับ
  - <> หรือ != ไม่เท่ากับ

Operators ทางลอจิก

- ตัว Logical operators
  - AND และ
  - OR หรือ
  - NOT ตรงกันข้าม, ไม่เท่ากับ

Operator (ตัวปฏิบัติการ) ที่ใช้ใน Where

- ตัวปฏิบัติการพิเศษ (Special operators)
  - **BETWEEN…AND….** ตรวจสอบช่วงของค่าใน Attribute
  - **IS NULL** ตรวจสอบว่ามีค่าว่างหรือไม่
  - **IS NOT NULL** ตรวจสอบว่าไม่มีค่าว่างหรือไม่
  - **LIKE** ตรวจสอบค่า String ใน Attribute ว่ามีส่วนคล้ายกับที่กำหนดไว้หลัง LIKE หรือไม่
  - **IN** ตรวจสอบค่าใน Attribute ว่าตรงกันกับที่กำหนดไว้หลัง IN หรือไม่
  - **EXISTS** ตรวจสอบว่ามีการ Return ค่าของ Subquery หรือไม่
  - **DISTINCT** จำกัดค่าให้แสดงผลโดยค่าไม่ซ้ำกัน

ตัวอย่างการใช้ AND

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000

❖ **ตัวอย่าง แสดงชื่อ นามสกุล ที่อยู่ และเงินเดือน โดยจะแสดงข้อมูลเฉพาะพนักงานที่อยู่จังหวัดอุตรดิตถ์และมีเงินเดือนมากกว่า 5000 บาท**

จากโจทย์

สิ่งที่ต้องแสดงคือ *ชื่อ นามสกุล เงินเดือน*

จากตาราง *พนักงาน*

เงื่อนไข *พนักงานคนนั้นต้องอยู่จังหวัดอุตรดิตถ์ และมีเงินเดือนมากกว่า 5000 บาท*

คอลัมน์เงื่อนไข *Address และ Salary*

ตัวอย่างการใช้ AND

❖ จากโจทย์จะเห็นว่า มีสองเงื่อนไข และต้องเป็นจริงทั้งสองเงื่อนไข ดังนั้นจึงต้องใช้ AND เชื่อมระหว่าง 2 เงื่อนไขนี้

❖ เขียน SQL ได้ดังนี้

```
SELECT  First_name, Last_name, Address, Salary
FROM Employees
WHERE Address = 'อุตรดิตถ์' AND Salary > 5000;
```

First_name	Last_name	Address	Salary
สมหญิง	งามแท้	อุตรดิตถ์	6000

จะเห็นว่าผลลัพธ์ที่ได้พนักงานคนนั้นต้องอยู่อุตรดิตถ์และมีเงินเดือนมากกว่า 5000

ตัวอย่างการใช้ OR

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000

❖ **ตัวอย่าง แสดงชื่อ นามสกุล ที่อยู่ และเงินเดือน โดยจะแสดงข้อมูลเฉพาะพนักงานที่อยู่จังหวัดอุตรดิตถ์ <u>หรือ</u>มีเงินเดือนมากกว่า 5000 บาท**

จากโจทย์

สิ่งที่ต้องแสดงคือ *ชื่อ นามสกุล เงินเดือน*

จากตาราง *พนักงาน*

เงื่อนไข *พนักงานคนนั้นต้องอยู่จังหวัดอุตรดิตถ์ <u>หรือ</u>มีเงินเดือนมากกว่า 5000 บาท*

คอลัมน์เงื่อนไข *Address และ Salary*

ตัวอย่างการใช้ OR

จากโจทย์จะเห็นว่า มีสองเงื่อนไข แต่โจทย์ใช้คำว่าหรือ แสดงว่ามีเงื่อนไขใดเงื่อนใดหนึ่งเป็นจริง หรือเป็นจริงทั้งสองเงื่อนไขก็ได้ ดังนั้นจึงต้องใช้ OR เชื่อมระหว่าง 2 เงื่อนไขนี้

- เขียน SQL ได้ดังนี้

```
SELECT First_name, Last_name, Address, Salary
FROM Employees
WHERE Address = 'อุตรดิตถ์' OR Salary > 5000;
```

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000

จะเห็นว่าผลลัพธ์ที่ได้ สมชายไม่ได้อยู่จังหวัดอุตรดิตถ์ แต่ที่สมชายแสดงในผลลัพธ์เพราะมีเงินเดือนมากกว่า 5000 สรุป ผลลัพธ์ที่ได้จาก OR เข้าเงื่อนไขใดเงื่อนไขหนึ่งก็เป็นจริงแล้ว

การใช้ Between…AND….

Between…AND… <กำหนดเงื่อนไขให้กับ where โดยเลือกช่วงข้อมูลที่สนใจ>

- **การกำหนดเงื่อนไข โดยที่สามารถกำหนดช่วงข้อมูลได้**

  •รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **Between** *Value1* **AND** *value2* **;**

ใน oracle การกำหนดช่วงข้อมูล จะรวมค่า Value1 และ Value2 ด้วย

เช่น WHERE Salary Between 4800 AND 12000

ผลลัพธ์ที่จะได้จะแสดงเงินเดือนตั้งแต่ 4800 จนถึง 12000

การใช้ Between…AND….

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร่	5000

❖ ตัวอย่าง แสดงชื่อ และเงินเดือนของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่มีเงินเดือน 6000 ถึง 15000

```
SELECT  First_name, Salary
FROM Employees
WHERE Salary between 6000 AND 15000;
```

First_name	Salary
สมชาย	15000
สมหญิง	6000

การใช้ Not Between…AND….

การใช้ Not ร่วมกับ Between….AND… เป็นการระบุเงื่อนไขว่า ข้อมูลต้องไม่อยู่ในช่วงที่กำหนด

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร่	5000

❖ ตัวอย่าง แสดงชื่อ และเงินเดือนของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่มีเงินเดือนไม่อยู่ในช่วง 6000 ถึง 15000

```
SELECT First_name, Salary
FROM Employees
WHERE Salary Not between 6000 AND 15000;
```

First_name	Salary
สมใจ	5000

การใช้ Like

- Like <ใช้ในการเปรียบเทียบคำในรูปแบบ String เพื่อหาคำที่ต้องการ โดยข้อความหรือคำที่ต้องการค้นหา จะทราบค่าหรือระบุเพียงบางส่วนเท่านั้น>

- รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **Like** *pattern* **;**

การใช้ Like

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **Like** *pattern* **;**

คือสิ่งที่ต้องต้องการค้นหา

โดย Pattern มีรูปแบบดังนี้

% หมายถึง ตัวอักษรใด ๆ จำนวนไม่จำกัดตัวอักษร

_ หมายถึง ตัวอักษรใด ๆ จำนวน 1 ตัวอักษร

การใช้ Like

ตัวอย่าง

รูปแบบ	ตัวอย่าง	คำอธิบาย	ผลลัพธ์
'%value'	'%s'	ตัวอักษรด้านหน้าเป็นอะไรก็ได้ ตัวสุดต้องเป็น s	Sandnes , Louis
'value%'	'N%'	ข้อความนั้นต้องขึ้นต้นด้วย N	Nerissa
'%value%'	'%a%'	ข้อความต้องมี a ประกอบอยู่ ในข้อความ	Baer, Louisa
'%value%value%'	'%a%e%'	ข้อความต้องมี a และ e ประกอบอยู่ในข้อความ แต่ลำดับของ a ต้องมาก่อนตัวอักษร e	Nayer, Baer
'_ value%'	'_ a%'	ในข้อความ ตัวอักษรตัวแรกเป็นอะไรก็ได้ ขอให้ตัวอักษรตัวที่ 2 เป็น a	Baida,Nayer
‘%value_ _ '	'%a_ _'	ในข้อความ ตัวอักษรสองตัวหลังจะเป็นอะไรก็ได้ แต่ตัวที่สาม (นับจากด้านหลัง) ต้องเป็น a	Rajs, Hall

การใช้ Not Like

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,...,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **Not Like** *pattern* **;**

การใช้ **Not Like** *คือการกำหนดเงื่อนว่า ข้อความที่เราต้องการค้นหาต้องไม่ใช่ที่เราระบุลงไปใน Pattern*

เช่น

SELECT *last_name*
FROM *Employees*
WHERE *last_name* **Not Like '%a%';**

ผลลัพธ์ที่ได้คือภาย last_name ที่แสดงผลจะไม่มีตัวอักษร a ปรากฏอยู่ใน last_name

การใช้ IN

IN <เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยเลือกเฉพาะค่าที่กำหนด>

รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **IN** *(value1,value2,...)*

การใช้ IN

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร่	5000
004	สมหวัง	ทุกเรื่อง	พะเยา	7800

❖ ตัวอย่าง แสดงรหัส และชื่อของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่มีรหัส 001 ,002,004

```
SELECT  Emp_ID, First_name
FROM Employees
WHERE Emp_ID IN  ('001', '002','C004')
```

Emp_ID	First_name
001	สมชาย
002	สมหญิง
004	สมหวัง

การใช้ Not IN

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร่	5000
004	สมหวัง	ทุกเรื่อง	พะเยา	7800

❖ ตัวอย่าง แสดงรหัส และชื่อของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่ไม่มีรหัส 001 ,002,004

```
SELECT  Emp_ID, First_name
FROM Employees
WHERE Emp_ID Not IN  ('001', '002','C004')
```

Emp_ID	First_name
003	สมใจ

การใช้ IS NULL

IS NULL <เป็นการระบุเงื่อนไขว่าคอลัมน์ที่ต้องการต้องมีค่าเป็น NULL>

รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **IS NULL**

การใช้ IS NULL

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม		5000
004	สมหวัง	ทุกเรื่อง		7800

❖ **ตัวอย่าง แสดงชื่อและที่อยู่ของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่ไม่ได้ระบุที่อยู่**

```
SELECT  First_name, Address
FROM Employees
WHERE Address IS NULL
```

First_name	Address
สมใจ	
สมหวัง	

การใช้ IS NOT NULL

IS NULL <เป็นการระบุเงื่อนไขว่าคอลัมน์ที่ต้องการต้องไม่มีค่าเป็น NULL>

รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง*
WHERE *ชื่อคอลัมน์* **IS NOT NULL**

การใช้ IS NOT NULL

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม		5000
004	สมหวัง	ทุกเรื่อง		7800

❖ **ตัวอย่าง แสดงชื่อและที่อยู่ของพนักงาน โดยจะแสดงข้อมูลเฉพาะพนักงานที่ระบุที่อยู่**

```
SELECT  First_name, Address
FROM Employees
WHERE Address IS NOT NULL
```

First_name	Address
สมชาย	เชียงใหม่
สมหญิง	อุตรดิตถ์

การใช้ DISTINCT

- **DISTINCT : ใช้สำหรับการแสดงผลค่าที่ซ้ำกันเพียง 1 ค่า**
  - **รูปแบบการใช้คำสั่ง**

    ให้ใส่ DISTINCT ไว้หน้า ATTRIBUTE ที่ต้องการให้แสดงผลค่าซ้ำเพียง 1 ค่า

รูปแบบ

```
SELECT DISTINCT ชื่อคอลัมน์
FROM ชื่อตาราง
```

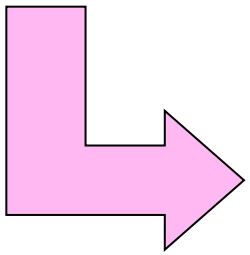
การใช้ DISTINCT

ตัวอย่าง

- **SELECT DISTINCT V_CODE**

  FROM PRODUCT;

ผลลัพธ์

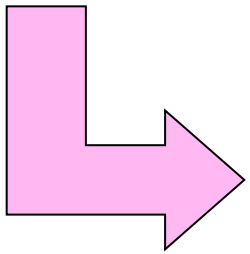


	V_CODE
	V0001
	V0003
	V0004
	V0006
	V0008
▶	V0011

การใช้ DISTINCT

- หากไม่ใช้ DISTINCT ผลลัพธ์จะได้ค่าที่ซ้ำกัน
  - SELECT V_CODE
    FROM PRODUCT;

ผลลัพธ์



V_CODE
V0011
V0004
V0004
V0006
V0006
V0008
V0008
V0011
V0001
V0004
V0008
V0001
V0003
V0011

ตัวอย่าง: SELECT DISTINCT

ตัวอย่าง

```
SELECT DISTINCT Amphur
FROM Personal
```

Personal

FirstName	LastName	Amphur
มานะ	พากเพียร	เมือง
อดทน	ตั้งใจเรียน	เมือง
มานี	หมั่นเพียร	ปง

ผลลัพธ์

Amphur
เมือง
ปง

การใช้ EXISTS

- **EXISTS**

  เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขโดยทำการตรวจสอบ ข้อมูลจากอีกตารางหนึ่งว่ามีข้อมูล หรือว่าไม่มีข้อมูลที่ต้องการเปรียบเทียบ

รูปแบบ

SELECT *ชื่อคอลัมน์ที่ต้องการดูข้อมูลที่ 1, ชื่อคอลัมน์ที่ 2,…,ชื่อคอลัมน์ที่ n*
FROM *ชื่อตาราง 1*
WHERE EXISTS *(***SELECT** *ชื่อคอลัมน์* **FROM** *ชื่อตารางที่ 2)*

การใช้ EXISTS

CustomerID	Name	Email	CountryCode	Budget	Used
C001	Win Weerachai	win.weerachai@thaicreate.com	TH	1000000	600000
C002	John Smith	john.smith@thaicreate.com	EN	2000000	800000
C003	Jame Born	jame.born@thaicreate.com	US	3000000	600000
C004	Chalee Angel	chalee.angel@thaicreate.com	US	4000000	100000

Table : customer1

CustomerID	Name	Email	CountryCode	Budget	Used
C001	Win Weerachai	win.weerachai@thaicreate.com	TH	1000000	600000
C002	John Smith	john.smith@thaicreate.com	EN	2000000	800000
C004	Chalee Angel	chalee.angel@thaicreate.com	US	4000000	100000

Table : customer2

การใช้ EXISTS

Sample1 การเลือกข้อมูลจากตาราง customer1 โดยข้อมูลจะเปรียบเทียบในตาราง customer2 ว่ามีข้อมูลเหมือนกันหรือไม่

```
SELECT * FROM customer1 WHERE EXISTS (SELECT * FROM customer2)
```

CustomerID	Name	Email	CountryCode	Budget	Used
C001	Win Weerachai	win.weerachai@thaicreate.com	TH	1000000	600000
C002	John Smith	john.smith@thaicreate.com	EN	2000000	800000
C004	Chalee Angel	chalee.angel@thaicreate.com	US	4000000	100000

การใช้ ORDER BY

❖ **ORDER BY : ใช้สำหรับการจัดเรียงลำดับในการแสดงผล**

- รูปแบบการใช้คำสั่ง

```
SELECT columnlist
FROM tablelist
[WHERE conditionlist]
[ORDER BY Columnlist [ ASC | DESC ]];
```

- โดย
  - ASC คือ การเรียงจากน้อยไปมาก
  - DESC คือ การเรียงจากมากไปน้อย

ตัวอย่าง: ORDER BY

ตัวอย่าง

```
SELECT *
FROM Address
ORDER BY ZipCode DESC
```

Address

Amphur	ZipCode
เมืองพะเยา	56000
จุน	56002
ปง	56001

ผลลัพธ์

Amphur	ZipCode
จุน	56002
ปง	56001
เมืองพะเยา	56000

แสดงรหัส, ชื่อ, ที่อยู่ และเงินค้าชำระของลูกค้าโดยเรียงตามเงินค้าชำระจากมากไปน้อย

Select Id, Name, Addr, Curr_Bal

From customer order by Curr_Bal desc ;

Id	Name	Addr	Curr_Bal
197	วรชาติ สีคล้ำ	อยุธยา	500000
110	ศิรี สุขพานิช	กรุงเทพฯ	200000
217	อนันต์ บุญญานุพงศ์	กรุงเทพฯ	200000
309	สุภาวดี เพชรสุข	ระยอง	150000
100	โสภา สีคล้ำ	กรุงเทพฯ	100000
...	...	...	

การใช้ ORDER BY

ตัวอย่าง

- ต้องการดูรหัสสินค้า ชื่อสินค้า วันที่สินค้าเข้ามา และราคาสินค้า โดยให้เรียงลำดับตามราคาสินค้าจากน้อยไปมาก
- SELECT P_CODE, P_DESCRIPT, P_INDATE, P_PRICE

  FROM PRODUCT

  ORDER BY P_PRICE;

  การเรียงจากน้อยไปมากไม่จำเป็นต้องใส่ ASC

การใช้ ORDER BY

ตัวอย่าง

- ต้องการดูรหัสสินค้า ชื่อสินค้า วันที่สินค้าเข้ามา และราคาสินค้า โดยให้เรียงลำดับตามราคาสินค้าจากมากไปน้อย
- SELECT P_CODE, P_DESCRIPT, P_INDATE, P_PRICE

  FROM PRODUCT

  ORDER BY P_PRICE DESC;

การใช้ ORDER BY

ตัวอย่าง

- ต้องการดูชื่อสินค้า รหัสผู้ค้าส่ง วันที่สินค้าเข้ามา และราคาสินค้า โดยให้เรียงลำดับตามรหัสผู้ค้าส่งมากน้อยไปหามาก ถ้าหากผู้ค้าส่งเป็นรายเดียวกันให้แสดงผลเรียงลำดับตามราคาสินค้า (P_PRICE) จากมากไปหาน้อย

```
SELECT  P_DESCRIPT, V_CODE, P_INDATE, P_PRICE
FROM  PRODUCT
ORDER  BY  V_CODE ASC, P_PRICE  DESC;
```

การใช้ ORDER BY

ตัวอย่าง

- ต้องการดูชื่อสินค้า รหัสผู้ค้าส่ง และราคาสินค้า โดยให้เรียงลำดับตามรหัสผู้ค้าส่งมากน้อยไปหามาก ถ้าหากผู้ค้าส่งเป็นรายเดียวกันให้แสดงผลเรียงลำดับตามราคาสินค้า (P_PRICE) จากมากไปหาน้อย

```
SELECT  P_DESCRIPT, V_CODE, P_PRICE
FROM  PRODUCT
WHERE P_PRICE <= 500
ORDER  BY  V_CODE ASC, P_PRICE  DESC;
```

การใช้ ALIAS

❖ **ALIAS**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดย **ALIAS** คือการสร้างชื่อจำลองขึ้นมาใหม่ให้คอลัมน์ในขณะที่แสดงผล

รูปแบบ

SELECT *ชื่อคอลัมน์ที่ 1 AS ชื่อคอลัมน์ใหม่ , ชื่อคอลัมน์ที่ 2,…*
FROM *ชื่อตาราง*

หมายเหตุ จะเปลี่ยนชื่อคอลัมน์ในการแสดงผลเท่านั้น โดยที่จะไม่ไปเปลี่ยนในโครงสร้างของตารางจริง

การใช้ ALIAS

Employees

Emp_ID	First_name	Last_name	Address	Salary
001	สมชาย	ชาตรี	เชียงใหม่	15000
002	สมหญิง	งามแท้	อุตรดิตถ์	6000
003	สมใจ	สุขสม	แพร่	5000

❖ **ตัวอย่าง แสดงรหัสและชื่อของพนักงาน โดยมีการเปลี่ยนชื่อคอลัมน์รหัส เป็น “ID”**

```
SELECT  Emp_ID AS ID, First_name
FROM Employees
WHERE Salary between 6000 AND 15000;
```

ID	First_name
001	สมชาย
002	สมหญิง
003	สมใจ

Operators ในการคำนวณ

ตัว Operators ในการคำนวณ

–	+	บวก
–	-	ลบ
–	*	คูณ
–	/	หาร
–	^	ยกกำลัง

การคำนวณในคำสั่ง SQL และการใช้ชื่อแทน (Alias)

การคำนวณ

- **SELECT P_DESCRIPT, P_ONHAND, P_PRICE, P_ONHAND*P_PRICE**

  FROM PRODUCT;

P_DESCRIPT	P_ONHAND	P_PRICE	Expr1003
หัวพ่นสี	8	1090.99	8727.92
ใบเลื่อย 7.25 นิ้ว	32	140.99	4511.68
ใบเลื่อย 9 นิ้ว	18	170.49	3068.82
แผ่นเหล็กหนา 1/4 นิ้ว	23	390.95	8991.85
แผ่นเหล็กหนา 1/2 นิ้ว	23	430.99	9912.77
เลื่อยจิ๊กซอร์ 12 นิ้ว	8	1090.92	8727.36
เลื่อยจิ๊กซอร์ 8 นิ้ว	6	990.87	5945.22
สว่านไร้สาย	12	380.95	4571.4
ค้อน	23	90.95	2091.85
ค้อน 12 ปอนด์	8	140.4	1123.2
ตะไบ	43	40.99	1762.57
เลื่อย Hicut 16 นิ้ว	11	2560.99	28170.89
ท่อ PVC 3.5 นิ้ว	188	50.87	9563.56
สกรู 1.25 นิ้ว	172	60.99	10490.28
สกรู 2.5 นิ้ว	237	80.45	19066.65
ตาข่ายเหล็ก 4 x 8 ฟุต	18	1190.95	21437.1
*			

สังเกตเมื่อมีการคำนวณโดยใช้ P_ONHAND*P_PRICE ผลลัพธ์ที่ได้จะแสดงในคอลัมน์ Expr1003 ซึ่งคอลัมน์ Expr1003 นี้เป็นคอลัมน์ที่ Access สร้างขึ้นมาให้เองโดยอัตโนมัติ

การคำนวณในคำสั่ง SQL และการใช้ชื่อแทน (Alias)

- การคำนวณและใช้ชื่อแทน (Alias)
  - ต้องการหาว่าสินค้าที่เหลืออยู่ในแต่ละตัวมีมูลค่ารวมแล้วเท่ากับเท่าไร
  - SELECT P_DESCRIPT, P_ONHAND, P_PRICE,
    P_ONHAND*P_PRICE AS TOTAL_VALUE
    FROM PRODUCT;

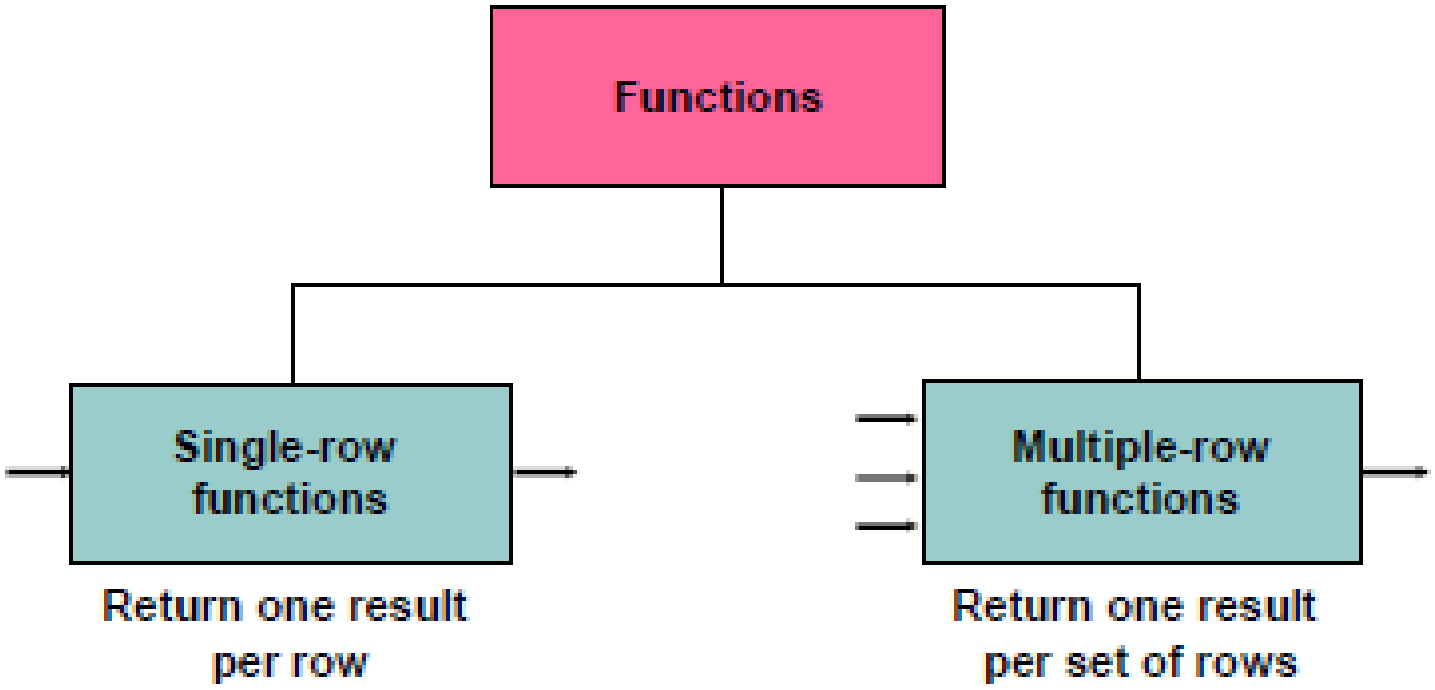
การคำนวณในคำสั่ง SQL และการใช้ชื่อแทน (Alias)

- การคำนวณและใช้ชื่อแทน (Alias)
  - ต้องการแสดงวันที่สินค้าแต่ละตัวจะมีวันสิ้นสุดการรับประกันเมื่อใด ในที่นี้คิดที่ 90 วัน โดยการคิดจะคิดเริ่มจากวันที่สินค้านั้นเข้ามาในคลังสินค้า
  - SELECT P_CODE, P_INDATE, P_INDATE + 90 AS EXPIRE_DATE

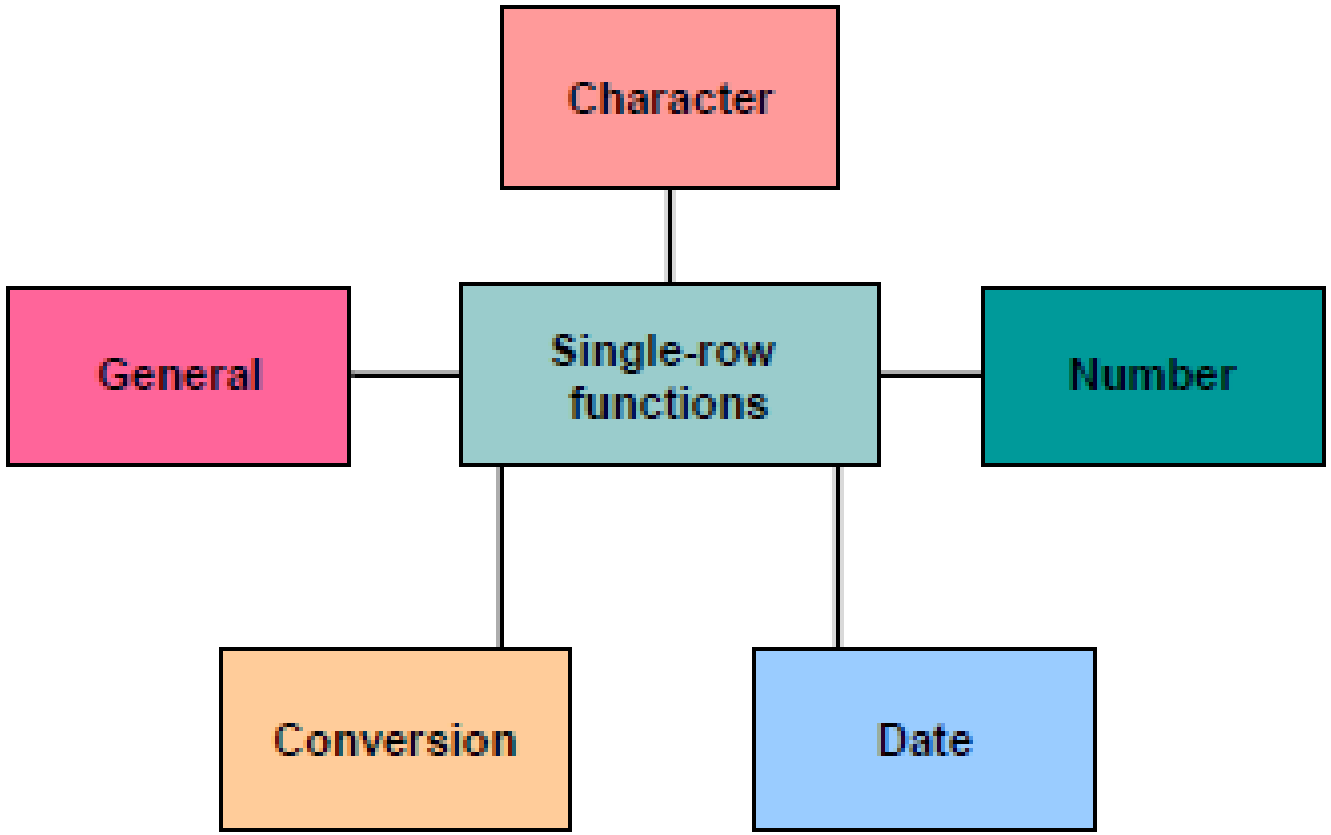
    FROM PRODUCT;

ฟังก์ชัน (Function)

ชนิดของฟังก์ชันใน SQL

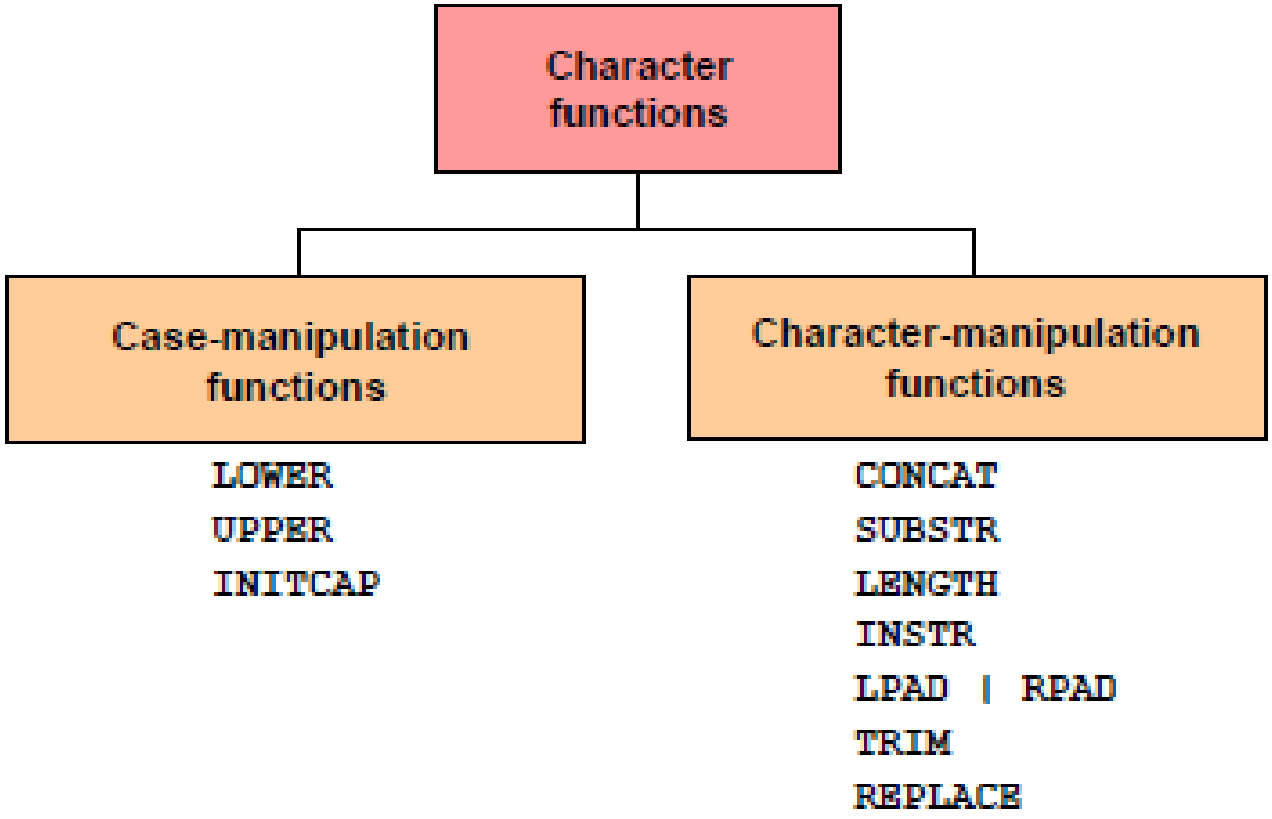


ฟังก์ชันเดี่ยว (Single Row function)



ฟังก์ชันเดี่ยว (Single Row function)

ฟังก์ชันในการจัดการตัวอักษร



ฟังก์ชันเดี่ยว (Single Row function)

ฟังก์ชันในการปรับเปลี่ยนตัวอักษร

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้

Display the employee number, name, and department number for employee Higgins:

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

ฟังก์ชันเดี่ยว (Single Row function)

ฟังก์ชันในการปรับเปลี่ยนตัวอักษร

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE','J','BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') 'Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
```

1 2 3

EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2

1 2 3

ฟังก์ชันเดี่ยว (Single Row function)

ฟังก์ชันเกี่ยวกับตัวเลข (Number Function)

การปัดเศษทศนิยม ตามจำนวนตำแหน่งที่ระบุ โดยมีการคำนึงทศนิยม

- ROUND: Rounds value to specified decimal
- TRUNC: Truncates value to specified decimal
- MOD: Returns remainder of division

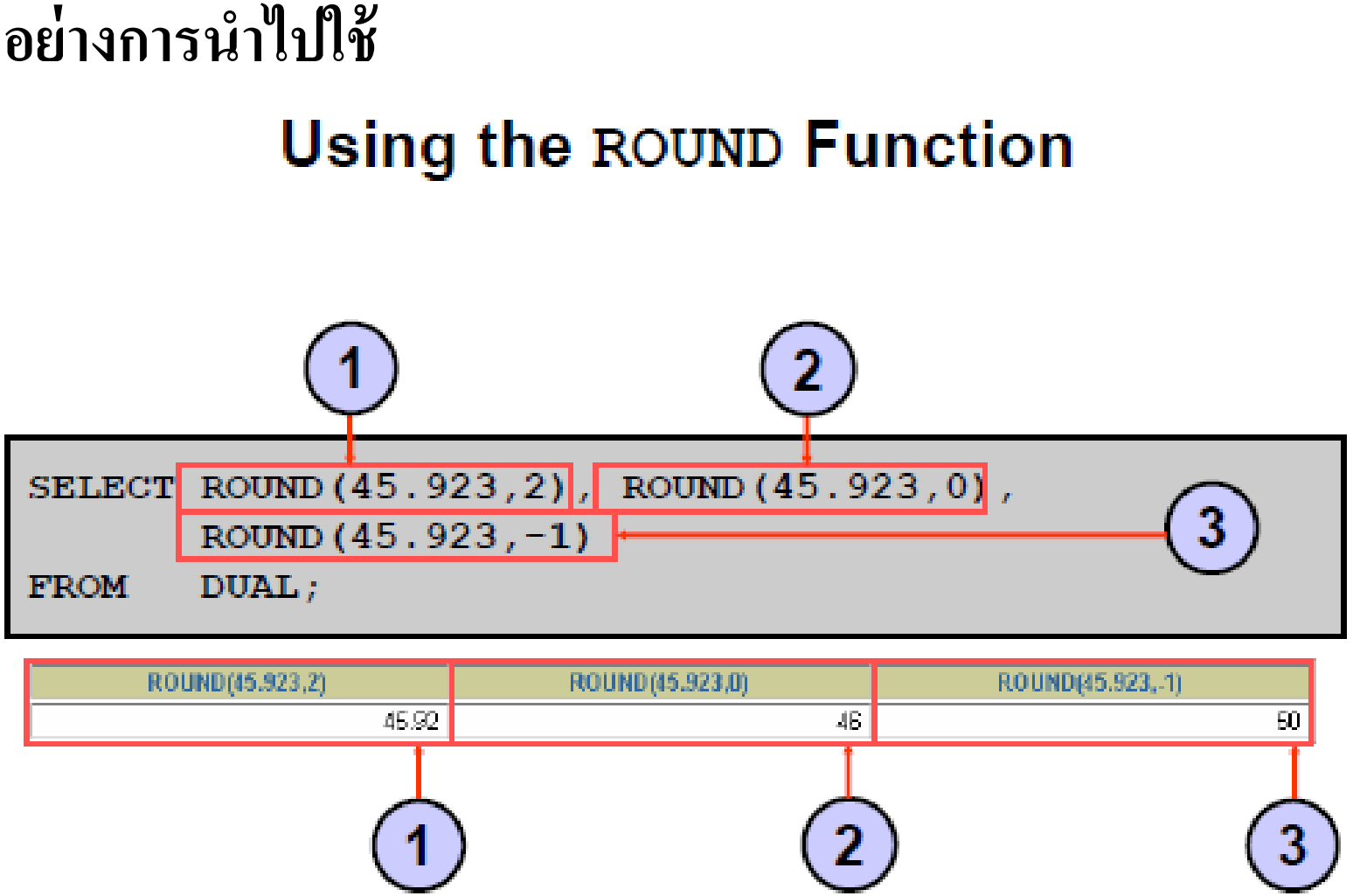
การปัดเศษทศนิยมทิ้งตามจำนวนตำแหน่งที่ระบุ

คือการหารที่สนใจแต่เศษ คำตอบที่ได้จะนำเศษที่เหลือมาตอบ

Function	Result
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

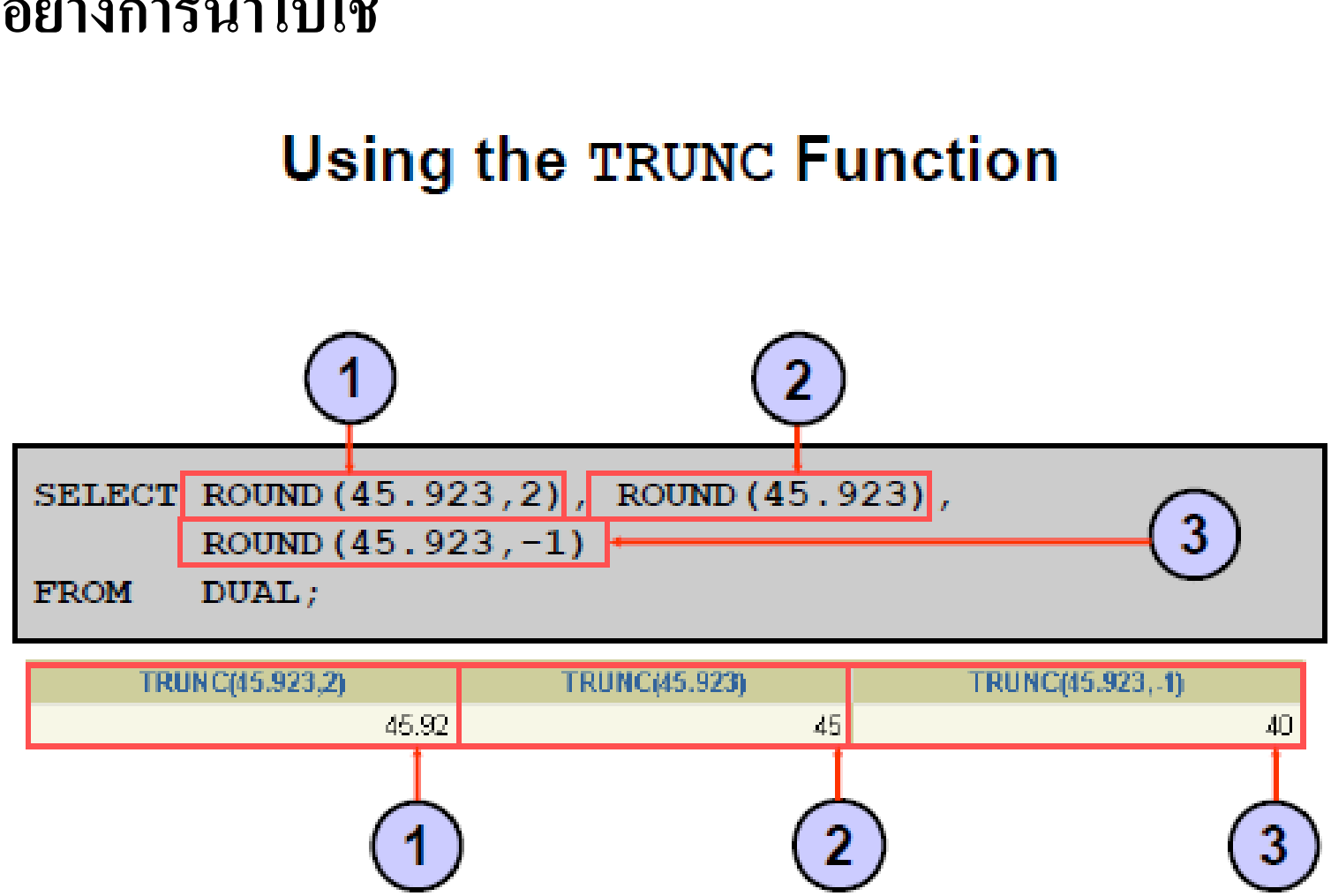
ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้



ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้



ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้

Using the MOD Function

For all employees with job title of Sales Representative, calculate the remainder of the salary after it is divided by 5,000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

ฟังก์ชันเดี่ยว (Single Row function)

ฟังก์ชันเกี่ยวกับ DATE

SYSDATE เป็นคำสั่งที่ใช้ในอ่านวันที่ปันจุบันของ Oracle Database

```
SELECT sysdate as "Date"
FROM dual;
```

Date
13-FEB-12

ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้

Using Arithmetic Operators with Dates

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

ฟังก์ชันเดี่ยว (Single Row function)

ฟังก์ชันเกี่ยวกับ DATE

Function	Result
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้

Function	Result
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94',6)	'11-JUL-94'
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

ฟังก์ชันเดี่ยว (Single Row function)

ตัวอย่างการนำไปใช้

Assume SYSDATE = '25-JUL-03':

Function	Result
ROUND(SYSDATE,'MONTH')	01-AUG-03
ROUND(SYSDATE ,'YEAR')	01-JAN-04
TRUNC(SYSDATE ,'MONTH')	01-JUL-03
TRUNC(SYSDATE ,'YEAR')	01-JAN-03

ฟังก์ชันสรุป (Aggregate function)

Group functions operate on sets of rows to give one result per group.

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
	7000
10	4400

...

Maximum salary in EMPLOYEES table

MAX(SALARY)
24000

ฟังก์ชันสรุป (Aggregate function)

❖ คำสั่งที่ใช้

- **COUNT** ใช้สำหรับนับค่า
- **MIN** หาค่าต่ำสุด
- **MAX** หาค่าสูงสุด
- **SUM** หาผลรวม
- **AVG** หาค่าเฉลี่ย

Aggregate function

- **COUNT : ใช้สำหรับการนับค่า**
- **ตัวอย่าง**
  - **ต้องการนับจำนวนผู้ค้าส่งทั้งหมด**
  - **SELECT COUNT(*)**

    FROM VENDOR;

Aggregate function

❖ ตัวอย่าง

- ต้องการนับจำนวนผู้ค่าส่งทั้งหมดที่ทำการส่งสินค้าให้เรา
- SELECT COUNT(*)

  FROM

  (SELECT DISTINCT V_CODE FROM PRODUCT

  WHERE V_CODE IS NOT NULL);

Aggregate function

ตัวอย่าง

- ต้องการนับจำนวนผู้ค่าส่งที่ส่งสินค้าให้กับเรา และราคาสินค้าที่ส่งน้อยกว่า 100
- SELECT COUNT(*)

  FROM

  (SELECT DISTINCT V_CODE FROM PRODUCT

  WHERE V_CODE IS NOT NULL

  AND P_PRICE < 100);

Aggregate function

- **MAX : หาค่าสูงสุด**
- **ตัวอย่าง**
  - **ต้องการหาราคาสินค้าที่แพงที่สุดในร้าน**
  - **SELECT MAX(P_PRICE)**

    FROM PRODUCT;

Aggregate function

❖ ตัวอย่าง

- ต้องการดูรายละเอียดของสินค้าที่แพงที่สุดในร้าน
- SELECT P_CODE, P_DESCRIPT, P_PRICE

  FROM PRODUCT

  WHERE P_PRICE = (SELECT MAX(P_PRICE)

  FROM PRODUCT);

Aggregate function

- **MIN : หาค่าต่ำสุด**
- **ตัวอย่าง**
  - **ต้องการหาราคาสินค้าที่ถูกที่สุดในร้าน**
  - **SELECT MIN(P_PRICE)**

    FROM PRODUCT;

Aggregate function

❖ ตัวอย่าง

- ต้องการดูรายละเอียดของสินค้าที่ถูกที่สุดในร้าน
- SELECT P_CODE, P_DESCRIPT, P_PRICE

  FROM PRODUCT

  WHERE P_PRICE = (SELECT MIN(P_PRICE)

  FROM PRODUCT);

Aggregate function

- **SUM : หาผลรวม**
- **ตัวอย่าง**
  - **ต้องการหามูลค่าสินค้าที่มีอยู่ในร้านทั้งหมด**
  - **SELECT SUM(P_ONHAND * P_PRICE) AS TOTAL_VALUE**

    FROM PRODUCT;

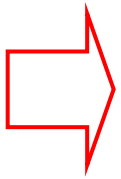
ตัวอย่าง: SUM

ตัวอย่าง

```
SELECT Name, SUM(Amount)
FROM Income
```

Income

Name	Amount
ANT	5500
BAT	4500
ANT	7100



ผลลัพธ์

Name	SUM(Amount)
ANT	17100
BAT	17100
ANT	17100

Aggregate function

- **AVG : หาค่าเฉลี่ย**
- **ตัวอย่าง**
  - **ต้องการหามูลค่าสินค้าเฉลี่ยที่อยู่ในร้าน**
  - **SELECT AVG(P_PRICE)**

    FROM PRODUCT;

Aggregate function

❖ ตัวอย่าง

- ต้องการดูสินค้าที่มีราคาสูงกว่าราคาสินค้าเฉลี่ย
- SELECT P_DESCRIPT, P_ONHAND, P_PRICE

  FROM PRODUCT

  WHERE P_PRICE > (SELECT AVG(P_PRICE)

  FROM PRODUCT)

  ORDER BY P_PRICE DESC;

GROUP BY และ HAVING

- **GROUP BY** เป็นคำสั่งที่ใช้สำหรับการแสดงผลในลักษณะจัดกลุ่ม อาจใช้ร่วมกับ **HAVING** ในการกำหนดเงื่อนไขของการแสดงผล
- รูปแบบการใช้คำสั่ง

SELECT *columnlist*

FROM *tablelist*

[WHERE *conditionlist***]**

[GROUP BY *columnlist***]**

[HAVING *conditionlist***]**

[ORDER BY *columnlist* **[ASC | DESC]];**

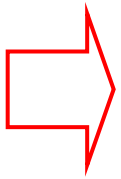
ตัวอย่าง: ใช้ GROUP BY

ตัวอย่าง

```
SELECT Name, SUM(Amount)
FROM Income
GROUP BY Name
```

Income

Name	Amount
ANT	5500
BAT	4500
ANT	7100



ผลลัพธ์

Name	SUM(Amount)
ANT	12600
BAT	4500

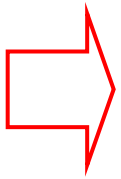
ตัวอย่าง: ใช้ GROUP BY และ HAVING

ตัวอย่าง

```
SELECT Name, SUM(Amount)
FROM Income
GROUP BY Name
HAVING NAME = “ANT”
```

Income

Name	Amount
ANT	5500
BAT	4500
ANT	7100



ผลลัพธ์

Name	SUM(Amount)
ANT	12600

GROUP BY และ HAVING

ตัวอย่าง

- ต้องการดูมูลค่ารวมของสินค้าที่แต่ละผู้ค้าส่ง ส่งสินค้าให้กับเรา
- SELECT V_CODE, SUM(P_PRICE * P_ONHAND)

  FROM PRODUCT

  WHERE V_CODE <> NULL

  GROUP BY V_CODE;

GROUP BY และ HAVING

ตัวอย่าง

- ต้องการดูมูลค่าเฉลี่ยของสินค้าที่แต่ละผู้ค้าส่ง ส่งสินค้าให้กับเรา
- SELECT V_CODE, AVG(P_PRICE)

  FROM PRODUCT

  WHERE V_CODE <> NULL

  GROUP BY V_CODE;

GROUP BY และ HAVING

ตัวอย่าง

- ต้องการดูมูลค่าเฉลี่ยของสินค้าที่แต่ละผู้ค้าส่ง ส่งสินค้าให้กับเรา โดยดูเฉพาะราคาเฉลี่ยที่ต่ำกว่า 100
- SELECT V_CODE, AVG(P_PRICE)

  FROM PRODUCT

  WHERE V_CODE <> NULL

  GROUP BY V_CODE

  HAVING AVG(P_PRICE) < 100;

GROUP BY และ HAVING

ตัวอย่าง

- ต้องการดูมูลค่าเฉลี่ยของสินค้าที่แต่ละผู้ค้าส่ง ส่งสินค้าให้กับเรา และให้นับจำนวนสินค้าที่ผู้ค่าส่งส่งให้ด้วย โดยดูเฉพาะราคาเฉลี่ยที่ต่ำกว่า 100

```
SELECT  V_CODE, COUNT(P_CODE), AVG(P_PRICE)
FROM PRODUCT
WHERE V_CODE <> NULL
GROUP BY V_CODE
HAVING AVG(P_PRICE) < 100;
```

GROUP BY และ HAVING

ตัวอย่าง

- ใช้ Alias (ชื่อแทน) เพื่อการแสดงผลที่ดียิ่งขึ้น
- SELECT V_CODE,

```
        COUNT(P_CODE) AS AMOUNT_OF_PRODUCT,
        AVG(P_PRICE) AS PRICE_AVERAGE
FROM PRODUCT
WHERE V_CODE <> NULL
GROUP BY V_CODE
HAVING AVG(P_PRICE) < 100;
```

GROUP BY และ HAVING

ตัวอย่าง

- ต้องการดูมูลค่ารวมของสินค้าที่แต่ละผู้ค้าส่ง ส่งสินค้าให้กับเรา และให้แสดงผลเรียงตามราคาจากมากไปหาน้อย
-

```
SELECT  V_CODE,
                SUM(P_ONHAND * P_PRICE)  AS   TOTAL_PRICE
FROM  PRODUCT
WHERE  V_CODE  <>  NULL
GROUP BY  V_CODE
ORDER BY  SUM(P_ONHAND * P_PRICE) DESC;
```

การแสดงผลจากหลายตารางโดยการ Join (เชื่อม)

ตัวอย่าง

- ต้องการดูรายละเอียดสินค้า และต้องการทราบชื่อผู้ค้าส่ง พนักงานที่เราติดต่อ และที่อยู่ของผู้ค้าส่ง
- SELECT PRODUCT.P_DESCRIPT, PRODUCT.P_PRICE,
  VENDOR.V_NAME, VENDOR.V_CONTACT,
  VENDOR.V_AREACODE, VENDOR.V_PHONE
  FROM PRODUCT, VENDOR
  WHERE PRODUCT.V_CODE = VENDOR.V_CODE;

การแสดงผลจากหลายตารางโดยการ Join (เชื่อม)

ตัวอย่าง

- ต้องการดูรายละเอียดสินค้าและผู้ค้าส่ง ที่ส่งสินค้า หลังวันที่ 15-JAN-2004
- SELECT PRODUCT.P_DESCRIPT, PRODUCT.P_INDATE,
  PRODUCT.P_PRICE, VENDOR.V_NAME,
  VENDOR.V_AREACODE, VENDOR.V_PHONE
  FROM PRODUCT, VENDOR
  WHERE PRODUCT.V_CODE = VENDOR.V_CODE
  AND P_INDATE > #15-JAN-2004#;

คำสั่ง SQL : DML

INSERT : เพิ่มแถว (Row) ลงใน Table

- รูปแบบการใช้คำสั่ง

  INSERT INTO *ชื่อตาราง* VALUES (*ข้อมูลทุกคอลัมน์*)

- ตัวอย่าง

```
INSERT INTO VENDOR
    VALUES ('V0001', 'บริษัทไบรซัน', 'สมัย', '615', '223-3234', 'TN', 'Y');
INSERT INTO PRODUCT
    VALUES ('P0000001', 'หัวพ่นสี', '03-NOV-2003', 8, 5, 1090.99, 0.00, 'V0011');
INSERT INTO PRODUCT
    VALUES ('P0000010', 'ค้อน 12 ปอนด์', '03-JAN-2004', 8, 5, 140.40, 0.05, NULL);
```

ตัวอย่าง: INSERT

```
INSERT INTO Orders(ProdID, Product, EmpID)
VALUES (999, “Ram”, “02”)
```

Orders

ProdID	Product	EmpID
234	Printer	01
657	Table	03
865	Chair	03

Orders

ProdID	Product	EmpID
234	Printer	01
657	Table	03
865	Chair	03
999	Ram	02

คำสั่ง SQL : DML

❖ UPDATE : แก้ไขข้อมูลในตาราง

- รูปแบบการใช้คำสั่ง

UPDATE *ชื่อตาราง*

SET *ชื่อคอลัมน์* = *ข้อมูลใหม่*

WHERE *เงื่อนไข*;

คำสั่ง SQL : DML

UPDATE : แก้ไขข้อมูลใน Table

- ตัวอย่าง
  - **UPDATE PRODUCT**

    SET P_DATE = ’18-JAN-2004’

    WHERE P_CODE = ‘P0000007’;
  - **แก้ไขข้อมูลวันที่สินค้าเข้าคลัง (P_DATE) โดยให้แก้ไขเฉพาะสินค้าที่มีรหัสสินค้า (P_CODE) เป็น P0000007 ให้เป็นวันที่สินค้าเข้าคลังเป็น 18-JAN-2004**

คำสั่ง SQL : DML

❖ UPDATE : แก้ไขข้อมูลใน Table

- ตัวอย่าง
  - **UPDATE Book**

    SET Author = 'สมโชค'

    WHERE Book_name = 'ระบบฐานข้อมูล';
  - **แก้ไขข้อมูลผู้แต่ง (Author) โดยให้แก้ไขเฉพาะหนังสือที่ชื่อระบบฐานข้อมูล (Book_name) ให้เป็นผู้แต่งชื่อสมโชค**

คำสั่ง SQL : DML

UPDATE : แก้ไขข้อมูลใน Table

- ตัวอย่าง
  - **UPDATE Book**

    SET Author = 'สมควร', publisher='ดอกหญ้า'

    WHERE Book_id = '00015';

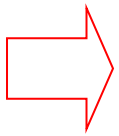
    แก้ไขข้อมูลผู้แต่ง (Author) โดยให้แก้ไขเฉพาะหนังสือที่มีรหัส 00015 (Book_id) ให้เป็นผู้แต่งชื่อสมโชค และสำนักพิมพ์ดอกหญ้า

ตัวอย่าง: UPDATE แบบไม่ระบุแถว

```
UPDATE Employees
SET Name = “FOX”
```

Employees

EmpID	Name
01	ANT
02	BAT
03	CAT
04	DOG



Employees

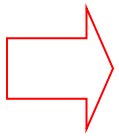
EmpID	Name
01	FOX
02	FOX
03	FOX
04	FOX

ตัวอย่าง: UPDATE แบบระบุแถว (WHERE)

```
UPDATE Employees
SET Name = “FOX”
WHERE EmpID = “04”
```

Employees

EmpID	Name
01	ANT
02	BAT
03	CAT
04	DOG



Employees

EmpID	Name
01	ANT
02	BAT
03	CAT
04	FOX

คำสั่ง SQL : DML

DELETE : ลบแถว (Row) ออกจากตาราง

- รูปแบบการใช้คำสั่ง

```
DELETE FROM ชื่อตาราง
WHERE เงื่อนไข;
```

- ตัวอย่าง

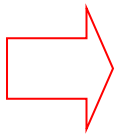
```
DELETE FROM PRODUCT
        WHERE P_CODE = 'P0000016';
DELETE FROM PRODUCT
        WHERE P_MIN = 5;
```

ตัวอย่าง: DELETE แบบระบุแถว (WHERE)

```
DELETE
FROM Employees
WHERE EmpID = “04”
```

Employees

EmpID	Name
01	ANT
02	BAT
03	CAT
04	FOX



Employees

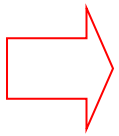
EmpID	Name
01	ANT
02	BAT
03	CAT

ตัวอย่าง: DELETE แบบไม่ระบุแถว

```
DELETE
FROM Employees
```

Employees

EmpID	Name
01	ANT
02	BAT
03	CAT
04	FOX



Employees

EmpID	Name

3. Data Control Language : DCL

- เป็นคำสั่งที่ใช้ในการกำหนดสิทธิการใช้งานในฐานข้อมูล
- ตัวอย่างคำสั่ง เช่น
  - GRANT เป็นคำสั่งให้สิทธิ์ผู้ใช้ (User)

อ้างอิง

- Database system : Design, Implementation & Management 6th edition, Rob & Coronel
- ระบบฐานข้อมูล รศ. ดร. วิเชียร เปรมชัยสวัสดิ์
- ระบบฐานข้อมูล รศ. ศิริลักษณ์ โรจนกิจอำนวย
- http://www.thaicreate.com/tutorial/sql.html