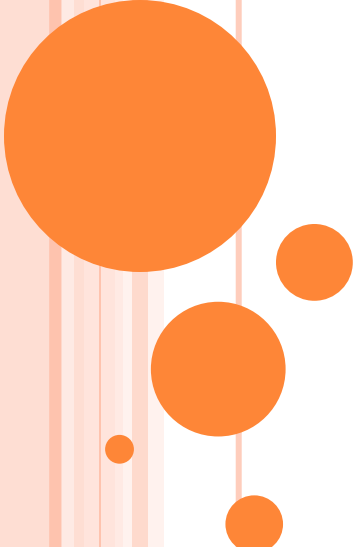


# INTRODUCTION TO STRUCTURED QUERY LANGUAGE

## LAB 5

- **RESTRICTING DATA**



By **Kanokwan Atchariyachanvanich**  
**Faculty of Information Technology**  
**KMITL**  
**Database System Concepts**  
**2/2565**

# OUTLINE ก่อนสอบกลางภาค

| Date               | SQL   |
|--------------------|---|
| 9, 11 JAN 2023     | <ul style="list-style-type: none"><li>• Lab Introduction</li><li>• Introduction to DBLearn (SQL tool)</li></ul> |
| 16, 18 JAN 2023    | LAB 1 - Creating and Managing Tables (DDL)  |
| 23, 25 JAN 2023    | LAB 2 - Including Constraints (DDL)   |
| 30 JAN, 1 FEB 2023 | LAB 3 - Manipulating Data (DML)   |
| 6, 8 FEB 2023      | LAB 4 - SQL SELECT Statements <ul style="list-style-type: none"><li>• Writing Basic</li></ul>                   |
| 13, 15 FEB 2023    | LAB 5 - SQL SELECT Statements <ul style="list-style-type: none"><li>• Restricting Data</li></ul>                |
| 20, 22 FEB 2023    | - ทวนก่อนสอบ Quiz 1   |
| 27 FEB, 1 MAR 2023 | <b>Quiz 1: LAB 1 – LAB 5</b>  |

# SQL STATEMENT

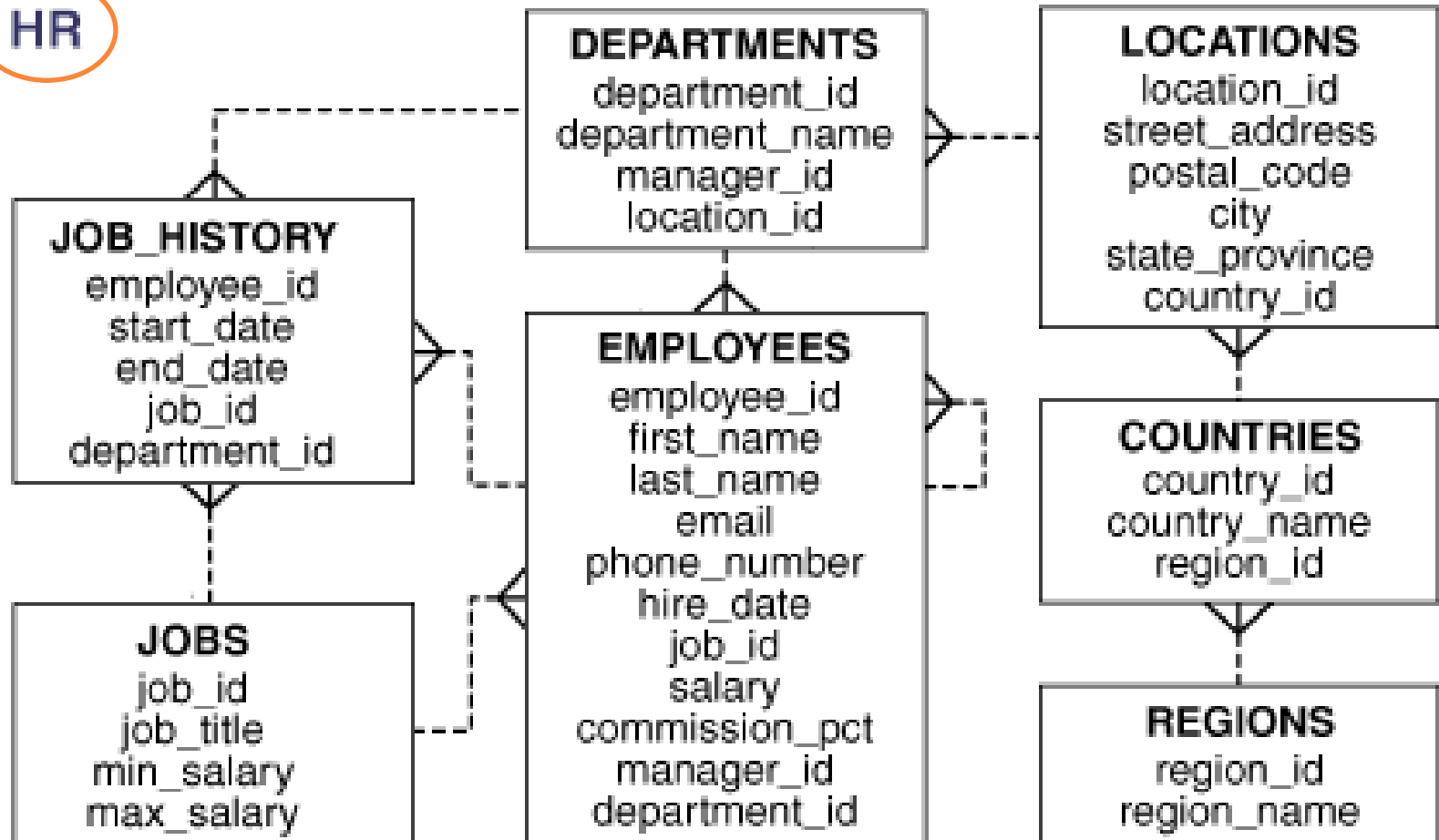
| Type  | SQL Statement  |   |
|---|--|---|
| <b>Data Manipulation Language (DML)</b><br>ภาษาสำหรับการจัดการข้อมูล คือส่วนของประโยค SQL ที่อนุญาตให้คุณ<br>ควบคุมหรือจัดการข้อมูล   | <b>SELECT</b><br><b>INSERT</b><br><b>UPDATE</b><br><b>DELETE</b><br><b>MERGE</b><br><b>CALL</b><br><b>EXPLAIN PLAN</b><br><b>LOCK TABLE</b>  |   |
| <b>Data Definition Language (DDL)</b><br>อธิบายส่วนของ SQL ที่อนุญาตให้สร้าง, เปลี่ยน, และทำลายอ็อบเจกต์<br>ฐานข้อมูล อ็อบเจกต์ฐานข้อมูลเหล่านี้รวมถึงแบบแผน, ตาราง, มุมมอง,<br>ลำดับ, แคตาล็อก, ดัชนี, และ alias | <b>CREATE</b><br><b>ALTER</b><br><b>DROP</b><br><b>RENAME</b><br><b>ANALYZE</b><br><b>AUDIT</b><br><b>COMMENT</b><br><b>ASSOCIATE STATISTICS</b><br><b>DISASSOCIATE STATISTICS</b> | <b>FLASHBACK</b><br><b>GRANT</b><br><b>NOAUDIT</b><br><b>PURGE</b><br><b>REVOKE</b><br><b>TRUNCATE</b><br><b>UNDROP</b> |
| <b>Transaction Control</b><br>จัดการ transaction จากการเปลี่ยนแปลงที่เกิดจาก DML  | <b>COMMIT</b><br><b>ROLLBACK</b><br><b>SAVEPOINT</b><br><b>SET TRANSACTION</b>   |   |

# OBJECTIVES

- After completing this lesson, you should be able to do the following:
  - **Limit** the rows retrieved by a query

# TABLES USED IN THE COURSE

HR



# EXAMPLE: EMPLOYEE TABLE DESCRIPTION

| Column Name    | Null?    | Type           |
|----------------|----------|----------------|
| EMPLOYEE_ID    | NOT NULL | NUMBER (6)     |
| FIRST_NAME     |          | VARCHAR2 (20)  |
| LAST_NAME      | NOT NULL | VARCHAR2 (25)  |
| EMAIL          | NOT NULL | VARCHAR2 (20)  |
| PHONE_NUMBER   |          | VARCHAR2 (20)  |
| HIRE_DATE      | NOT NULL | DATE           |
| JOB_ID         | NOT NULL | VARCHAR2 (10)  |
| SALARY         |          | NUMBER (8 , 2) |
| COMMISSION_PCT |          | NUMBER (2 , 2) |
| MANAGER_ID     |          | NUMBER (6)     |
| DEPARTMENT_ID  |          | NUMBER (4)     |

# LIMITING ROWS USING A SELECTION

| EMPLOYEE_ID | LAST_NAME | JOB_ID  | DEPARTMENT_ID |
|-------------|-----------|---------|---------------|
| 100         | King      | AD_PRES | 90            |
| 101         | Kochhar   | AD_VP   | 90            |
| 102         | De Haan   | AD_VP   | 90            |
| 103         | Hunold    | IT_PROG | 60            |
| 104         | Ernst     | IT_PROG | 60            |
| 107         | Lorentz   | IT_PROG | 60            |
| 124         | Mourgos   | ST_MAN  | 50            |

....

20 rows selected.

**“retrieve all employees in department 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            |

# LIMITING THE ROWS SELECTED

- Restrict the rows returned by using the **WHERE** clause.

```
SELECT    *|{[DISTINCT] column_name | expression [alias],...}  
FROM      table_name  
WHERE     condition(s);
```

- Where restricts the query to rows that meet a condition  
*condition* is composed of column names, expressions, constants, literal values, and a comparison operator



# COMPARISON CONDITIONS

| Operator | Meaning                  |
|----------|--------------------------|
| =        | Equal to                 |
| >        | Greater than             |
| >=       | Greater than or equal to |
| <        | Less than                |
| <=       | Less than or equal to    |
| !=       | Not qual to              |

....WHERE hire\_date= '1998-05-04'

....WHERE salary>=6000

....WHERE last\_name='Smite'

# USING THE WHERE CLAUSE

```
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            |

## EXERCISE # 1

- จงแสดงรหัสพนักงาน ชื่อจริง และนามสกุล ของพนักงานที่มีรหัส 100

```
SELECT  employee_id, first_name, last_name
FROM    employees
WHERE    employee_id = 100;
```

# CHARACTER STRINGS AND DATES

- Character strings and date values are enclosed in single quotation marks (' ').
- Character values are not case sensitive.
- The default date format is YYYY-MM-DD.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'wHalen';
```

| LAST_NAME | JOB_ID  | DEPARTMENT_ID |
|-----------|---------|---------------|
| Whalen    | AD_ASST | 10            |

## EXERCISE # 2

- จงแสดงรหัสสถานที่ตั้งสำนักงาน ชื่อเมือง และรหัสไปรษณีย์ ของสถานที่ตั้งสำนักงานที่ไม่ได้ตั้งอยู่ในประเทศรหัส US

```
SELECT  location_id, city, postal_code
FROM    locations
WHERE   country_id != "US";
```

# USING COMPARISON CONDITIONS

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

| LAST_NAME | SALARY |
|-----------|--------|
| Matos     | 2600   |
| Vargas    | 2500   |

....

| LAST_NAME | SALARY |
|-----------|--------|
| OConnell  | 2600   |
| Grant     | 2600   |

26 rows selected.

## EXERCISE # 3

- จงแสดงชื่อจริง นามสกุล และวันที่เริ่มจ้างงาน ของพนักงานที่ถูกจ้างงานตั้งแต่วันที่ 1 มกราคม 2000 เป็นต้นไป โดยเรียงลำดับข้อมูลเงินเดือนจากมากไปน้อย

```
SELECT      first_name, last_name, hire_date
FROM        employees
WHERE       hire_date >= "2000-01-01"
ORDER BY    salary DESC;
```

## OTHER COMPARISON CONDITIONS

| Operator             | Meaning                         |
|----------------------|---------------------------------|
| BETWEEN<br>...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                 |



# USING THE BETWEEN CONDITION

- Use the **BETWEEN** condition to display rows based on a range of values.

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

Lower limit      Upper limit

| LAST_NAME   | SALARY |
|-------------|--------|
| Khoo        | 3100   |
| Baida       | 2900   |
| Tobias      | 2800   |
| Himuro      | 2600   |
| Colmenares  | 2500   |
| Nayer       | 3200   |
| Mikkilineni | 2700   |

17

## EXERCISE # 4

- จงแสดงรหัสพนักงาน อีเมล และเบอร์โทรศัพท์ ของพนักงานที่มีรหัสตั้งแต่ 110 ถึง 120 (ใช้ BETWEEN)

```
SELECT  employee_id, email, phone_number
FROM    employees
WHERE   employee_id BETWEEN 110 AND 120;
```

## EXERCISE # 5

- จงแสดงรหัสตำแหน่งงาน ชื่อตำแหน่งงาน และเงินเดือนต่ำที่สุด ของตำแหน่งงานที่มีเงินเดือนต่ำที่สุดที่ไม่ได้อยู่ในช่วง 3000 ถึง 5000 (ใช้ BETWEEN)

```
SELECT  job_id, job_title, min_salary
FROM    jobs
WHERE   min_salary NOT BETWEEN 3000 AND 5000;
```

# USING THE IN CONDITION

- Use the **IN** membership condition to test for values in a list.

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

| EMPLOYEE_ID | LAST_NAME | SALARY | MANAGER_ID |
|-------------|-----------|--------|------------|
| 101         | Kochhar   | 17000  | 100        |
| 102         | De Haan   | 17000  | 100        |
| 114         | Raphaely  | 11000  | 100        |
| 120         | Weiss     | 8000   | 100        |
| 121         | Fripp     | 8200   | 100        |
| 122         | Kaufling  | 7900   | 100        |
| 123         | Vollman   | 6500   | 100        |
| 124         | Mourgos   | 5800   | 100        |
| 145         | Russell   | 14000  | 100        |
| 146         | Partners  | 13500  | 100        |
| 147         | Errazuriz | 12000  | 100        |
| 148         | Cambrault | 11000  | 100        |
| 149         | Zlotkey   | 10500  | 100        |
| 201         | Hartstein | 13000  | 100        |
| 108         | Greenberg | 12000  | 101        |
| 200         | Whalen    | 4400   | 101        |
| 203         | Mavris    | 6500   | 101        |
| 204         | Baer      | 10000  | 101        |
| 205         | Higgins   | 12000  | 101        |
| 202         | Fay       | 6000   | 201        |

**Remark: if characters or dates are used in the list, they must be enclosed in single quotation marks**

## EXERCISE # 6

- จงแสดงนามสกุล รหัสงาน และรหัสแผนก ของพนักงานที่ทำงานอยู่ในแผนก รหัส 10 20 30 หรือ 40 (ใช้ IN)

```
SELECT last_name, job_id, department_id
FROM employees
WHERE department_id IN (10,20,30,40);
```

## EXERCISE # 7

- จงแสดงรหัสสถานที่ตั้งสำนักงาน ชื่อเมือง และรหัสไปรษณีย์ ของสถานที่ตั้งสำนักงานที่ไม่ได้ตั้งอยู่ในเมือง Venice, Tokyo, Toronto และ Bern (ใช้ IN)

```
SELECT location_id, city, postal_code
FROM locations
WHERE city NOT IN ("Venice", "Tokyo", "Toronto", "Bern");
```

# USING THE LIKE CONDITION

- Use the **LIKE** condition to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
  - % denotes zero or many characters.
  - \_ denotes one character.

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

13 rows selected.

# USING THE LIKE CONDITION

- You can combine pattern-matching characters.
- Example: displays the names of all employees whose last names have an o as the second character

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

| LAST_NAME  |
|------------|
| Colmenares |
| Doran      |
| Fox        |
| Johnson    |
| Jones      |
| Kochhar    |
| Lorentz    |
| Mourgos    |
| Popp       |
| Rogers     |
| Tobias     |
| Vollman    |

12 rows selected.



## EXERCISE # 8

- จงแสดง รหัสพนักงาน และรหัสงาน ของพนักงานที่เริ่มทำงานในเดือนมกราคมของปีใดก็ได้ กำหนดให้ใช้ LIKE

```
SELECT  employee_id, job_id
FROM    employees
WHERE   hire_date LIKE '%-01-%';
```

## EXERCISE # 9

- จงแสดง ชื่อเต็ม (คั่นชื่อจริงและนามสกุลด้วย whitespace เช่น David Austin ตั้งชื่อว่า name) และเงินเดือน โดยแสดงเฉพาะพนักงานที่ชื่อจริงมี 5 ตัวอักษร และอักษรตัวที่สองคือตัว a

```
SELECT  CONCAT(first_name, ' ', last_name) name, salary
FROM    employees
WHERE   first_name LIKE '_a_____';
```

# USING THE LIKE CONDITION (MySQL)

- **ESCAPE** identifier default : \
- **ESCAPE** identifier to search for the **actual % and \_ symbols**, or have special “**and & or |**”
- Example: search for job\_id that contains 'SA\_'

```
SELECT  employee_id, last_name, job_id
FROM    employees
WHERE   job_id LIKE '%SA\__%' ;
```

| employee_id | last_name | job_id |
|-------------|-----------|--------|
| 145         | Russell   | SA_MAN |
| 146         | Partners  | SA_MAN |
| 147         | Errazuriz | SA_MAN |
| 148         | Cambrault | SA_MAN |
| 149         | Zlotkey   | SA_MAN |

# USING THE LIKE CONDITION (MySQL)

- **ESCAPE** identifier default : \
- **ESCAPE** identifier ex: \$ to search for the **actual % and \_ symbols**, or have special “**and & or |**”
- Example: search for job\_id that contains 'SA\_'

```
SELECT  employee_id, last_name, job_id
FROM    employees
WHERE   job_id LIKE '%SA$__%' ESCAPE '$'
```

| employee_id | last_name | job_id |
|-------------|-----------|--------|
| 145         | Russell   | SA_MAN |
| 146         | Partners  | SA_MAN |
| 147         | Errazuriz | SA_MAN |
| 148         | Cambrault | SA_MAN |
| 149         | Zlotkey   | SA_MAN |

35 rows selected.

# USING THE NULL CONDITION

- **Test for nulls with the IS NULL operator.**
- Example: retrieve the last names and managers of all employee who do not have a manager

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL ;
```

| LAST_NAME | MANAGER_ID |
|-----------|------------|
| King      |            |

# LOGICAL CONDITIONS

| 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          |

# USING THE AND OPERATOR

- **AND** requires both conditions to be true.
- Example: only employees who have a job title that contains the string MAN *and* earn \$10,000 or more are selected

```
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 |
|-------------|-----------|-----------|--------|--------|
| 114         | Raphaely  | PU_MAN    | 11000  |        |
| 145         | Russell   | SA_MAN    | 14000  |        |
| 146         | Partners  | SA_MAN    | 13500  |        |
| 147         | Errazuriz | SA_MAN    | 12000  |        |
| 148         | Cambrault | SA_MAN    | 11000  |        |
| 149         | Zlotkey   | SA_MAN    | 10500  |        |
| 201         | Hartstein | MK_MAN    | 13000  |        |

# USING THE OR OPERATOR

- **OR** requires either conditions to be true.
- Example: any employee who has a job ID containing MAN or earn \$10,000 or more are selected

```
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  |
| 108         | Greenberg | FI_MGR  | 12000  |
| 114         | Raphaely  | PU_MAN  | 11000  |
| 120         | Weiss     | ST_MAN  | 8000   |
| 121         | Fripp     | ST_MAN  | 8200   |
| 122         | Kaufling  | ST_MAN  | 7900   |
| 123         | Vollman   | ST_MAN  | 6500   |
| 124         | Mourgos   | ST_MAN  | 5800   |
| 145         | Russell   | SA_MAN  | 14000  |
| 146         | Partners  | SA_MAN  | 13500  |
| 147         | Errazuriz | SA_MAN  | 12000  |
| 148         | Cambrault | SA_MAN  | 11000  |
| 149         | Zlotkey   | SA_MAN  | 10500  |
| 150         | Tucker    | SA_REP  | 10000  |
| 156         | King      | SA_REP  | 10000  |
| 162         | Vishney   | SA_REP  | 10500  |
| 168         | Ozer      | SA_REP  | 11500  |
| 169         | Bloom     | SA_REP  | 10000  |
| 174         | Abel      | SA_REP  | 11000  |
| 201         | Hartstein | MK_MAN  | 13000  |
| 204         | Baer      | PR_REP  | 10000  |
| 205         | Higgins   | AC_MGR  | 12000  |

24 rows selected.



## EXERCISE # 10

- จงแสดงชื่อแผนก รหัสผู้จัดการ และรหัสสถานที่ตั้งสำนักงาน ของแผนกที่มีชื่อขึ้นต้นด้วย IT หรือมีผู้จัดการดูแลอยู่

```
SELECT  department_name, manager_id, location_id
FROM    departments
WHERE   department_name LIKE "IT%"
OR      manager_id IS NOT NULL;
```

## EXERCISE # 11

- จงแสดงชื่อจริง นามสกุล และเงินเดือน ของพนักงานที่อยู่ในแผนกรหัส 80 และมีเงินเดือนสูงกว่า 10000

```
SELECT      first_name, last_name, salary
FROM        employees
WHERE       department_id = 80
AND         salary > 10000;
```

## EXERCISE # 12

- จงแสดงชื่อจริง และอีเมลของพนักงานที่มีเงินเดือนอยู่ระหว่าง \$2000 และ \$3000 และรหัสงานขึ้นต้นด้วย ST\_ (ต้องใช้ BETWEEN, LIKE)

| first_name | email    |
|------------|----------|
| Irene      | IMIKKILI |
| James      | JLANDRY  |

```
SELECT first_name, email
FROM employees
WHERE salary between 2000 and 3000
AND job_id LIKE 'ST\_%';
```

# USING THE NOT OPERATOR

- Example: displays the last names and job ID of all employees whose job ID *is not* IT\_PROG, ST\_CLERK, or SA\_REP

```
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 |

52 rows selected.

## EXERCISE # 13

- จงแสดงชื่อถนนและที่อยู่ และ รัฐ/จังหวัด ของสถานที่ตั้งสำนักงาน (คั่นด้วย comma ตามด้วย whitespace เช่น 2017 Shinjuku-ku, Tokyo Prefecture ตั้งชื่อใหม่ว่า Address) ที่มีรหัสที่ตั้ง ที่ไม่ได้อยู่ระหว่าง 1500 และ 1800 และ ข้อมูล รัฐ/จังหวัด ต้องไม่เป็นค่าว่าง

| Address                            |
|------------------------------------|
| 2017 Shinjuku-ku, Tokyo Prefecture |
| 2014 Jabberwocky Rd, Texas         |

```
SELECT  concat(street_address, ', ', state_province) Address
FROM    locations
WHERE   location_id NOT BETWEEN 1500 AND 1800
AND     state_province IS NOT NULL ;
```

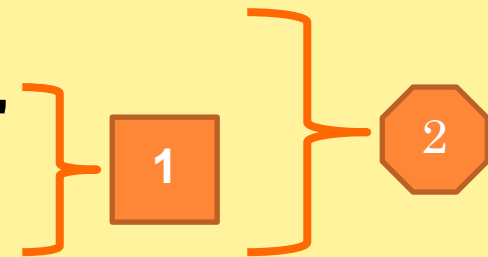
# RULES OF PRECEDENCE

| Order Evaluated       | Operator                      |
|-----------------------|-------------------------------|
| 1      *   /   +   -  | Arithmetic operators          |
| 2                     | Concatenation operator        |
| 3 =, >, >=, <, <=, <> | Comparison conditions         |
| 4                     | IS [NOT] NULL, LIKE, [NOT] IN |
| 5                     | [NOT] BETWEEN                 |
| 6                     | NOT logical condition         |
| 7                     | AND logical condition         |
| 8                     | OR logical condition          |

Override rules of precedence by using parentheses.

# RULES OF PRECEDENCE

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id= 'SA_REP'
OR job_id= 'AD_PRES'
AND salary > 15000;
```



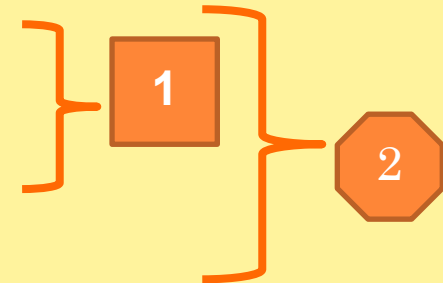
- Select the row if an employee is a president *and* earns more than \$15,000, *or* if the employee is a sales representative

| LAST_NAME | JOB_ID  | SALARY |
|-----------|---------|--------|
| King      | AD_PRES | 24000  |
| Abel      | SA_REP  | 11000  |
| Taylor    | SA_REP  | 8600   |
| Grant     | SA_REP  | 7000   |

# RULES OF PRECEDENCE

- Use parentheses to force priority

```
SELECT    last_name, job_id, salary
FROM      employees
WHERE     ( job_id= 'SA_REP'
OR        job_id= 'AD_PRES')
AND       salary > 15000;
```



- Select the row if an employee is a president *or* a sales representative, *and* if the employee earns more than \$15,000

| LAST_NAME | JOB_ID  | SALARY |
|-----------|---------|--------|
| King      | AD_PRES | 24000  |



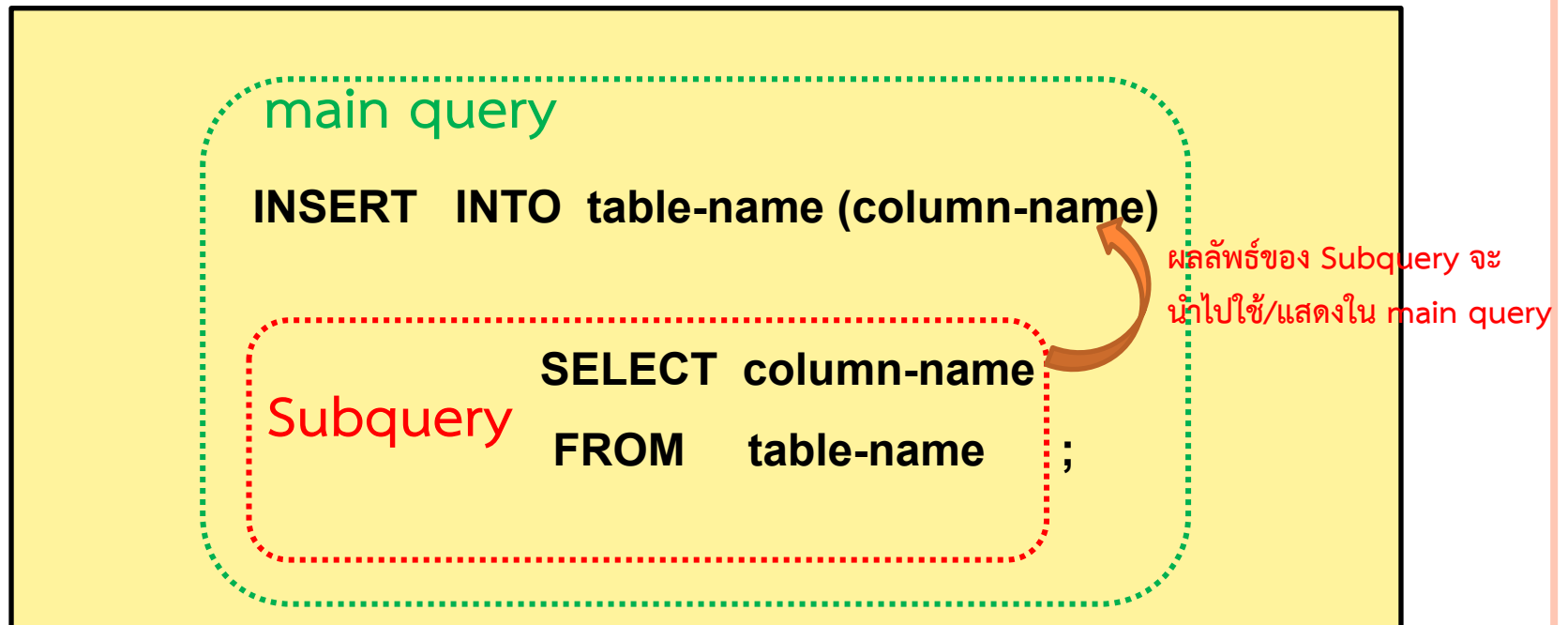
# SUMMARY

- In this lesson, you should have learned how to:
- Use the WHERE clause to restrict rows of output
  - Use the comparison conditions
  - Use the BETWEEN, IN, LIKE, and NULL conditions
  - Apply the logical AND, OR, and NOT operators
- Use the ORDER BY clause to sort rows of output

```
SELECT    *|{[DISTINCT] column_name | expression [alias],...}  
FROM      table_name  
[ WHERE   condition(s) ]  
[ ORDER BY {column_name, expr, alias} [ASC | DESC] ] ;
```

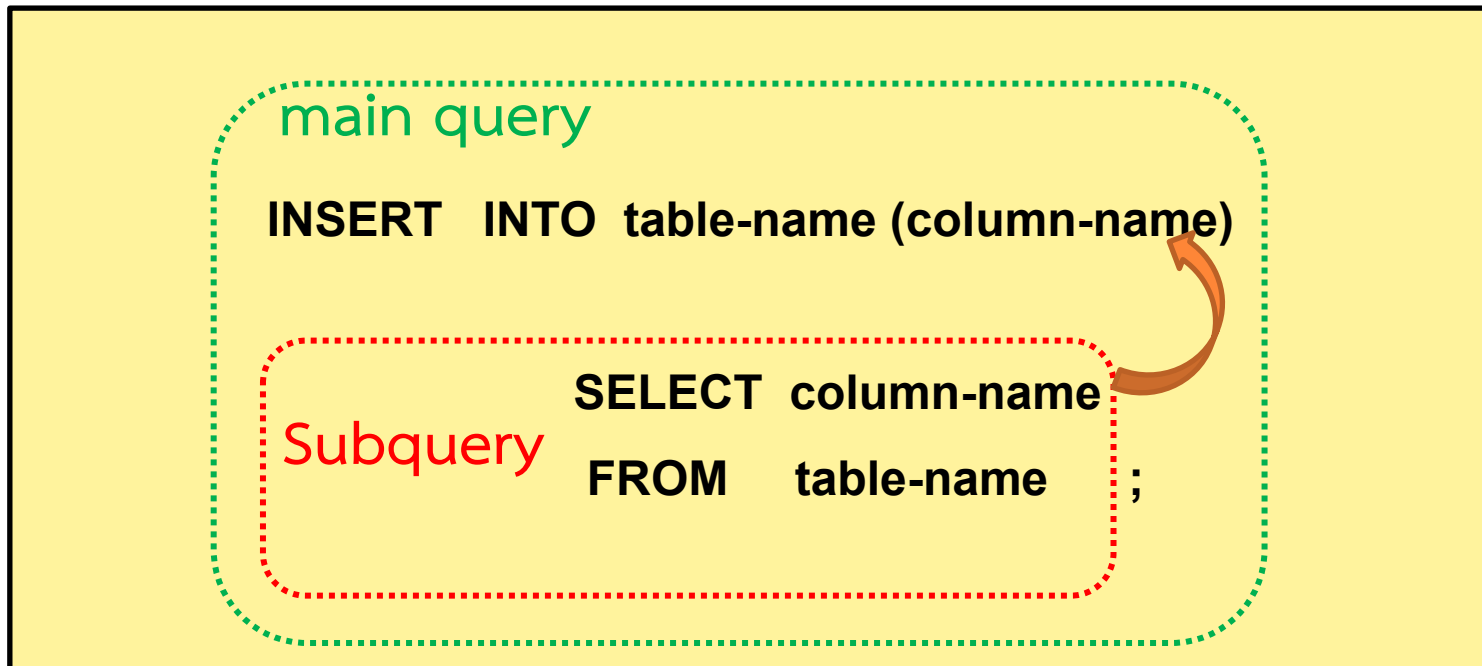
# SELECT & DATA MANIPULATION LANGUAGE

## ○ INSERT



# INSERTING MULTIPLE NEW ROWS

- COPYING ROWS FROM ANOTHER TABLE ต้องการนำข้อมูลจากตารางอื่นมาใส่ในตารางใหม่ ดังนั้นข้อมูลที่เราไม่รู้จำเป็นต้องใช้ subquery ดึงข้อมูลนั้นมา
- เขียน INSERT statement ด้วย subquery ห้ามใช้บรรทัด VALUES clause.
- จับคู่คอลัมน์ใน INSERT clause กับคอลัมน์จาก subquery.



# INSERTING MULTIPLE NEW ROWS

- COPYING ROWS FROM ANOTHER TABLE
- เขียน INSERT statement ด้วย subquery
- ห้ามใช้บรรทัด VALUES clause.
- จับคู่คอลัมน์ใน INSERT clause กับคอลัมน์จาก subquery.

```
INSERT INTO sales_reps (reps_id, name, salary, commission_pct)
      SELECT employee_id, last_name, salary, commission_pct
      FROM   employees ;
```

Subquery  
returns  
rows into  
the table

# INSERTING MULTIPLE NEW ROWS

- COPYING ROWS FROM ANOTHER TABLE
- เขียน INSERT statement ด้วย subquery
- ห้ามใช้บรรทัด VALUES clause.
- จับคู่คอลัมน์ใน INSERT clause กับคอลัมน์จาก subquery.

```
INSERT INTO sales_reps (reps_id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
 FROM   employees
 WHERE  job_id > 'R' ;
```

Subquery  
returns  
rows into  
the table

# INSERT INTO + Subquery

## Exercise # 14

- จงเพิ่มข้อมูลให้กับตาราง sales\_reps โดยนำข้อมูลจากตาราง employees ซึ่งเอาเฉพาะพนักงานที่มีเงินเดือนมากกว่า 5000 (ระบุคอลัมน์)

| Table      | Column1     | Column2   | Column3 | Column4        |
|------------|-------------|-----------|---------|----------------|
| sales_reps | reps_id     | name      | salary  | commission_pct |
| employees  | employee_id | last_name | salary  | commission_pct |

```
INSERT INTO sales_reps (reps_id, name, salary, commission_pct)
  SELECT  employee_id, last_name, salary, commission_pct
  FROM    employees
  WHERE   salary > 5000 ;
```

# SELECT & DATA MANIPULATION LANGUAGE

- **UPDATE** แก้ไขค่าในคอลัมน์ ด้วยการใช้ defined value (ตัวแปร) ใน UPDATE statements เพื่อแก้ไขค่าต่างๆ ในคอลัมน์ตามค่าที่ได้จาก SELECT statement ซึ่งเป็นตารางเดียวกับที่ต้องการแก้ไข

SET @ชื่อตัวแปร := (

SELECT column\_name

FROM table\_name

WHERE condition

);

ผลลัพธ์ของ ตัวแปร จะนำไปใช้  
ใน SET statement

**UPDATE** table-name (column-name)

**SET** column\_name = @ชื่อตัวแปร

**WHERE** condition

## UPDATING ONE COLUMN WITH A SUBQUERY (MySQL)

- แก้ไข 1 คอลัมน์ในตารางเดียวกัน ด้วยการใช้นิยามตัวแปร user defined variable
- ต้องกำหนด user defined variable เพื่อเก็บ result set ที่ได้จาก subquery
- ตัวอย่าง : แก้ไขเงินเดือนของพนักงานรหัส 150 ให้เหมือนกับเงินเดือนของพนักงานรหัส 153.

```
SET    @value_id := (  
  
                SELECT    salary  
                FROM      employees  
                WHERE     employee_id = 153  
  
);
```

```
UPDATE  employees  
SET     salary      = @value_id  
WHERE   employee_id = 150 ;
```



# UPDATE + Defined value

## Exercise # 15

- จงเขียน SQL statement เพื่อแก้ไขอัตราคอมมิชชั่น (commission\_pct) ของพนักงานรหัส 180 ให้เหมือนกับอัตราคอมมิชชั่นของพนักงานรหัส 170 โดยตั้งชื่อตัวแปรคือ comm

```
SET      @comm := (  
                                SELECT  commission_pct  
                                FROM      employees  
                                WHERE     employee_id = 170  
                                );  
UPDATE    employees  
SET        commission_pct      = @comm  
WHERE      employee_id = 180 ;
```

# UPDATING ROWS BASED ON ANOTHER TABLE

- แก้ไขด้วยการใช้ Subquery ใน UPDATE statements เพื่ออัปเดตแถวต่างๆ ในตารางตามค่าที่อยู่ในอีกตารางหนึ่ง

```
UPDATE  sales_reps
SET      salary          = (SELECT  salary
                             FROM      employees
                             WHERE     employee_id = 151) ,
        commission_pct  = (SELECT  commission_pct
                             FROM      employees
                             WHERE     employee_id = 151)
WHERE    reps_id = 151 ;
```

# SELECT & DATA MANIPULATION LANGUAGE

- DELETE: ลบแถวข้อมูลในตาราง ซึ่งมีเงื่อนไขตามค่าที่ได้จากการใช้ SELECT statement โดยที่ตารางที่ถูกลบแถวข้อมูลกับตารางที่ได้ค่าเงื่อนไขคือตารางเดียวกัน

SET @ชื่อตัวแปร := (

SELECT column\_name

FROM table\_name

WHERE condition

);

ผลลัพธ์ของ ตัวแปร จะนำไปใช้  
ใน where statement

**DELETE FROM table-name (column-name)**

**WHERE column\_name = @ชื่อตัวแปร**

# DELETE + Defined value

## Exercise # 16

- ลบแถวในตาราง sales\_reps ที่มีเงินเดือนน้อยกว่า เงินเดือนของ sales\_reps รหัส 105 โดยตั้งชื่อตัวแปรคือ sal

```
SET      @sal := (  
                select salary  
                from  sales_reps  
                where reps_id = 105);  
  
DELETE FROM  sales_reps  
          WHERE salary < @sal ;
```

## DELETING ROWS BASED ON ANOTHER TABLE

- ลบคอลัมน์ ด้วยการ ใช้ Subquery ใน DELETE FROM statements เพื่อลบแถวต่างๆ ในตารางตามค่าที่อยู่ในอีกตารางหนึ่ง
- ตัวอย่าง : ลบแถวในตาราง sales\_reps ที่มี reps\_id เหมือนกับ รหัสพนักงานที่มีนามสกุลคือ Banda จากตาราง employees

```
DELETE FROM sales_reps
WHERE reps_id =
        (SELECT employee_id
         FROM employees
         WHERE last_name='Banda') ;
```

- *Note: Don't forget to verify your deletion.*

# DELETE + Subquery

## Exercise # 17

- ลบแถวในตาราง sales\_reps ที่มีเงินเดือนน้อยกว่า เงินเดือนของ employee รหัส 105 ในตาราง employees

```
DELETE FROM sales_reps
        WHERE salary < (select salary
                        from employees
                        where employee_id = 105);
```