เรียกดูข้อมูลทั้งหมดของพนักงานผู้หญิงที่ถูกจ้างตั้งแต่ปี 2000 เป็นต้นไป

```
use employees;
select * from employees
where hire_date >= '2000-01-01' and gender = 'F';
```

เรียกดูข้อมูลทั้งหมดของพนักงานทุกคนที่มีรายได้สูงกว่า 150,000 ดอลลาร์ต่อปี

```
use employees;
select * from salaries
where salary > 150000;
```

SELECT DISTINCT

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยทำการเลือกข้อมูลที่ซ้ำกันมาเพียงแค่ Record เดียว

เรียกดูข้อมูลทั้งหมดของวันที่จ้างงานที่แตกต่างกันจากตาราง employees

```
use employees;
select distinct hire_date from employees;
```

Aggregate Functions

COUNT(A): นับแถวที่มีค่าใน Column A
SUM(A): ผลรวมของค่าทั้งของในแถวของ Column A
AVG(A): หาค่าเฉลี่ยของ Column A
MAX(A): หาค่ามากที่สุดของ Column A
MIN(A): หาค่าน้อยที่สุดของ Column A

เรียกดูข้อมูลทั้งหมดของสัญญาจ้างที่มีมูลค่าสูงกว่า 100,000 ดอลลาร์ต่อปี ขึ้นไปจากตาราง salaries

```
use employees;
select count(*)
from salaries
where salary > 100000;
```

เรียกดูข้อมูลทั้งหมดของ managers ที่มีอยู่จากตาราง employees

```
use employees;
select count(*)
from dept_manager;
```

ORDER BY

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดย จัดเรียงข้อมูลตามต้องการ

ASC (Default) เรียงจาก น้อยไปมาก และ DESC เรียงจาก มากไปน้อย

เรียกดูข้อมูลทั้งหมดของวันที่จ้างทำงานตั้งแต่ปัจจุบันไปยังอดีตที่มีอยู่จากตาราง employees

```
use employees;
select *
from employees
order by hire_date desc;
```

GROUP BY

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยใช้หาผลรวมของคอลัมน์ จากแถวใน Column ที่ระบุและทำการรวม Group ภายใต้ Column ที่อยู่หลัง GROUP BY

ALIAS (ชื่อจำลอง)
เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดย ALIAS คือการสร้างชื่อจำลองขึ้นมาใหม่ โดยสามารถจำลองชื่อได้ทั้งชื่อ Field และชื่อ Table

เรียกดูข้อมูลทั้งหมดของคอลัมน์แรกจะต้องมีเงินเดือนต่อปีสูงกว่า 80,000 ดอลลาร์ ในส่วนคอลัมน์ที่สองเปลี่ยนชื่อเป็น“ emps_with_same_salary” ต้องแสดงจำนวนสัญญาว่าจ้างของพนักงานในตาราง salary

```
use employees;
select salary, count(emp_no) as emps_with_same_salary
from salaries
where salary > 80000
group by salary
order by salary;
```

HAVING
เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดย HAVING ใช้กำหนดเงื่อนไข การเลือกข้อมูล ที่อยู่ภายหลังคำสั่ง GROUP BY
เรียกดูข้อมูลทั้งหมดของค่าเฉลี่ยเงินพนักงานที่สูงกว่า 120,000 ดอลลาร์ต่อปี

```
use employees;
select emp_no, avg(salary)
from salaries
group by emp_no
having avg(salary) > 120000
order by emp_no;
```

เรียกดูข้อมูลหมายเลขพนักงานทุกคนที่ลงนามมากกว่า 1 สัญญาหลังจากวันที่ 1 มกราคม 2000

```
use employees;
select emp_no
from dept_emp
where from_date > '2000-01-01'
group by emp_no
having count(from_date) > 1
order by emp_no;
```

LIMIT
เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) ที่สามารถกำหนดจำนวน Record ที่แสดงผลออกมาได้

เรียกดูข้อมูล 100 บรรทัดแรกจากตาราง dept_emp

```
use employees;
select *
from dept_emp
limit 100;
```

Aggregate Function

COUNT()
เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยทำการนับ จำนวน Count Record ที่ค้นพบ (numeric and non-numeric data)

เรียกดูจำนวนข้อมูลทั้งหมดแผนกจากตาราง dept_emp

```
use employees;
select count(distinct dept_no) as count_dept
from dept_emp;
```

SUM()

- เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยหาค่าผลรวมของฟิวด์
- numeric data เท่านั้น

เรียกดูจำนวนเงินทั้งหมดที่ใช้ในการจ่ายเงินเดือนสำหรับทุกสัญญาจ้างเริ่มตั้งแต่วันที่ 1 มกราคม 1997

```
use employees;
select sum(salary)
from salaries
where from_date > '1997-01-01';
```

MAX() และ MIN()

- เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยหาค่าสูงสุด (max) หรือ ค่าต่ำสุด(min) ในฟิวด์
- numeric data เท่านั้น

เรียกดูหมายเลขของพนักงานที่ต่ำสุด

```
use employees;
select min(emp_no)
from employees;
```

เรียกดูหมายเลขของพนักงานที่สูงสุด

```
use employees;
select max(emp_no)
from employees;
```

AVG()

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยหาค่าเฉลี่ยผลรวมของฟิวด์

numeric data เท่านั้น

เรียกดูค่าเฉลี่ยเงินทั้งหมดที่ใช้ในการจ่ายเงินเดือนสำหรับทุกสัญญาจ้างเริ่มตั้งแต่วันที่ 1 มกราคม 1997

```
use employees;
select avg(salary)
from salaries
where from_date = '1997-01-01';
```

ROUND()
เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยทำการปัดเศษขึ้นในกรณีที่มากกว่า >= .5 และปัดเศษลงในกรณีที่น้อยกว่า < .5
numeric data เท่านั้น
เรียกดูค่าเฉลี่ยเงินทั้งหมดที่ใช้ในการจ่ายเงินเดือนสำหรับทุกสัญญาจ้างเริ่มตั้งแต่วันที่ 1 มกราคม 1997 ในหน่วยเซนต์ (cents)

```
use employees;
select round(avg(salary), 2)
from salaries
where from_date > '1997-01-01';
```

SQL JOINs
เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (Table) โดยเงื่อนไขการ JOIN จะกระทำเมื่อมีข้อมูลตั้งแต่ 2 Table ขึ้นไป โดยข้อมูลเหล่านั้นเป็นข้อมูลที่มีความสัมพันธ์และเชื่อมโยงกับข้อมูลหลัก

สร้างและเติมข้อมูลลงไปในตาราง departments_dup

```
use employees;
DROP TABLE IF EXISTS departments_dup;
CREATE TABLE departments_dup ( dept_no CHAR(4) NULL , dept_name VARCHAR(40) NULL);
INSERT INTO departments_dup ( dept_no, dept_name ) SELECT * FROM departments;
INSERT INTO departments_dup (dept_name) VALUES  ('Public Relations');
SET SQL_SAFE_UPDATES = 0;
DELETE FROM departments_dup WHERE dept_no = 'd002';
INSERT INTO departments_dup(dept_no) VALUES ('d010'), ('d011');
```

```
use employees;

DROP TABLE IF EXISTS dept_manager_dup;

CREATE TABLE dept_manager_dup ( emp_no int(11) NOT NULL,
dept_no char(4) NULL, from_date date NOT NULL, to_date date NULL );

INSERT INTO dept_manager_dup select * from dept_manager;

INSERT INTO dept_manager_dup (emp_no, from_date) VALUES (999904,
'2017-01-01'), (999905,'2017-01-01'),(999906, '2017-01-01'),(999907,
'2017-01-01');

SET SQL_SAFE_UPDATES = 0;

DELETE FROM dept_manager_dup WHERE dept_no = 'd001';
```

เรียกดูข้อมูลหมายเลขพนักงาน,ชื่อ - นามสกุล, หมายเลขของแผนก และ วันที่ว่าจ้างของผู้จัดการโดยใช้ where และ join

```
use employees;

select employees.emp_no, employees.first_name, employees.last_name, dept_manager.dept_no, employees.hire_date

from employees, dept_manager

where employees.emp_no = dept_manager.emp_no;
```

เราสามารถประกาศตัวแปรได้ใน From;

```
use employees;

select e.emp_no, e.first_name, e.last_name, dm.dept_no, e.hire_date

from employees e, dept_manager dm

where e.emp_no = dm.emp_no;
```

Join in:

```
use employees;

select e.emp_no, e.first_name, e.last_name, dm.dept_no, e.hire_date

from employees e join dept_manager dm

on e.emp_no = dm.emp_no;
```

เรียกดูเลือกชื่อและนามสกุล,วันที่ว่าจ้างและตำแหน่งงานของพนักงานทุกคนที่มีชื่อ "Margareta" และมีนามสกุล "Markovitch"โดยใช้ where และ join ร่วมกัน

```
use employees;

select e.first_name, e.last_name, e.hire_date, t.title

from employees e

join titles t

on e.emp_no = t.emp_no

where first_name = 'Margareta' and last_name = 'Markovitch';

order by e.emp_no;
```

INNER JOINs

ให้นำเฉพาะแถวที่มีสมาชิกเหมือนกัน (Intersection)

เรียกดูข้อมูลหมายเลขพนักงาน,ชื่อ - นามสกุล, หมายเลขของแผนก และ วันที่ว่าจ้างของผู้จัดการ

```
use employees;
select e.emp_no, e.first_name, e.last_name, dm.dept_no, e.hire_date
from employees e
join dept_manager dm on e.emp_no = dm.emp_no;
```

LEFT JOINs / LEFT OUTER JOINs

เอาตารางซ้ายเป็นหลัก

เชื่อมตาราง 'employees' และ 'dept_manager' เพื่อเรียกดูพนักงานที่เป็นผู้จัดการทุกคนที่มีนามสกุล Markovitch

```
use employees;
select e.emp_no, e.first_name, e.last_name, e.hire_date, dm.dept_no
from employees e
left join dept_manager dm on e.emp_no = dm.emp_no
where last_name = 'Markovitch'
order by emp_no;
```

RIGHT JOINs / RIGHT OUTER JOINs

เอาตารางขวาเป็นหลัก

เชื่อมตาราง 'employees' และ 'dept_manager' เพื่อเรียกดูพนักงานที่เป็นผู้จัดการทุกคนที่มีนามสกุล Markovitch

```
use employees;
select e.emp_no, e.first_name, e.last_name, e.hire_date, dm.dept_no
from employees e
right join dept_manager dm on e.emp_no = dm.emp_no
where last_name = 'Markovitch'
order by emp_no;
```

CROSS JOIN / CARTESIAN JOIN

เป็นการรวม row ระหว่าง 2 Tables โดยจะวนจนครบตามจำนวนของข้อมูลแบบ multiple set หรือเรียกว่า Cartesian product โดยจะนำข้อมูลบน Table หลักแต่ละแถวไปรวมกับข้อมูลของ Table รองทั้งหมด โดยไล่ไปจนถึง row สุดท้ายของ Table หลัก

ใช้คำสั่ง cross join เพื่อเรียกดูความเกี่ยวข้องกันของผู้จัดการจากตาราง dept_manager และ department หมายเลข 9

```
use employees;
select dm.*, d.*
from departments d
cross join dept_manager dm
where d.dept_no = 'd009'
order by dept_name;
```

ใช้คำสั่ง where เพื่อเรียกดูพนักงาน 10 คนแรกจากทุกแผนก

```
use employees;
select e.*, d.*
from employees e
cross join departments d
group by e.emp_no
order by e.emp_no
limit 10;
```

UNION / UNION ALL

เป็นคำสั่งที่ใช้สำหรับการเลือกข้อมูลโดยทำการรวมจำนวนแถวระหว่าง Table เข้าด้วยกัน
สำหรับ UNION และ UNION ALL ต่างกันตรงที่
UNION จะเลือกข้อมูลที่ระหว่าง 2 ตาราง ตามกฎ DISTINCT ของ Table นั้น ๆ คือข้อมูลใน Table หนึ่ง ๆ จะไม่ซ้ำกัน
UNION ALL เลือกข้อมูลโดยไม่สนใจ คือเลือกเอาทั้งหมด

ความหมายของเครื่องหมายลบซับเซตของ a ที่บรรทัดสุดท้าย (ORDER BY -a.emp_no DESC)

```
use employees;
select *
from (
select e.emp_no, e.first_name, e.last_name, null as dept_no, null as
from_date
from employees e
where last_name = 'Denis' union select null as emp_no, null as first_name,
null as last_name, dm.dept_no, dm.from_date
from dept_manager dm ) as a
order by - a.emp_no desc;
```

SQL Subqueries

Subquery หรือ Inner query หรือ Nested query เป็นการดึงข้อมูลใน table จากผลลัพธ์ของการทำ select query ก่อนหน้านี้อีกที ซึ่งการทำ subquery สามารถใช้งาน Where Clause ไม่ว่าจะเป็น main query หรือ ส่วนของ subquery ส่วนมากแล้วมักถูกใช้ในเงื่อนไขที่ main query ไม่สามารถดึงข้อมูลได้ตามเงื่อนไขปกติที่สามารถทำได้ หรือ ติดข้อจำกัดภายใต้กฎของ SQL

กฎการใช้งาน Subquery

1. Subquery ต้องอยู่ภายใต้ วงเล็บ
2. Subquery สามารถมีได้อย่างน้อย 1 column ใน Select Clause นอกจาก main query จะมีหลาย column ซึ่งทั้งหมดต้องมีใน subquery เพื่ออ้างอิงด้วย
3. ORDER BYไม่สามารถใช้ใน subquery ได้ แต่ main query สามารถใช้ได้
4. GROUP BYไม่สามารถใช้ใน subquery ได้ แต่ main query สามารถใช้ได้
5. ไม่สามารถใช้ BETWEEN Operator กับ subquery ได้ แต่สามารถใช้ภาย

ใต้ subquery ได้

เรียกดูผู้จัดการแผนกทั้งหมดที่ได้รับการว่าจ้างระหว่าง 1 มกราคม 2533 และ 1 มกราคม 2538

```
use employees;

SELECT * FROM dept_manager

WHERE emp_no IN (

    SELECT emp_no FROM employees

    WHERE hire_date BETWEEN '1990-01-01' AND '1995-01-01');
```