-- GROUP BY ....HAVING

select \* from employees;

-- find all job id in each department

Select job\_id, department\_id from employees group by job\_id;

-- find max salary under each department

Select max(salary), department\_id from employees group by department\_id;

-- only for one dept filtering

Select max(salary), department\_id from employees group by department\_id having department\_id=40;

-- sub group

-- find all department for each job\_id

Select department\_id, job\_id from employees group by job\_id, department\_id;

Select max(salary), department\_id, job\_id from employees group by department\_id, job\_id;

-- Natural join

select employee\_id, first\_name, last\_name, department\_id, department\_name from employees natural join departments;

SELECT department\_id, department\_name,

location\_id, city

FROM departments

NATURAL JOIN locations ;

-- USING CLAUSE

SELECT employees.employee\_id, employees.last\_name,

departments.location\_id, department\_id

FROM employees JOIN departments

USING (department\_id);

-- with aliases name

SELECT e.employee\_id, e.last\_name, d.location\_id, department\_id

FROM employees e JOIN departments d USING (department\_id);

-- retrieving with ON command

-- When the column names are different then on command can be used

SELECT e.employee\_id, e.last\_name, e.department\_id,

d.department\_id, d.location\_id

FROM employees e JOIN departments d

ON (e.department\_id = d.department\_id);

select \* from employees;

-- Self-Joins Using the ON Clause

SELECT e.last\_name emp, m.last\_name mgr

FROM employees e JOIN employees m

ON (e.manager\_id = m.employee\_id);

-- Applying Additional Conditions to a Join

SELECT e.employee\_id, e.last\_name, e.department\_id,

d.department\_id, d.location\_id

FROM employees e JOIN departments d

ON (e.department\_id = d.department\_id)

AND e.manager\_id = 149 ;

-- Creating Three-Way Joins with the ON Clause

SELECT employee\_id, city, department\_name, l.city

FROM employees e

JOIN departments d

ON d.department\_id = e.department\_id

JOIN locations l

ON d.location\_id = l.location\_id;

-- Retrieving Records with Nonequijoins

select \* from jobs;

alter table jobs add column grade\_level char;

describe jobs;

select \* from jobs order by max\_salary desc;

select \* from jobs where job\_id='AD\_VP';

update jobs set grade\_level='A'

where job\_id='AD\_PRES';

SELECT e.last\_name, e.salary, j.grade\_level

FROM employees e JOIN jobs j

ON e.salary

BETWEEN j.min\_salary AND j.max\_salary;

-- Outer Joins

-- INNER Versus OUTER Joins

-- LEFT OUTER JOIN

SELECT e.last\_name, e.department\_id, d.department\_name

FROM employees e LEFT OUTER JOIN departments d

ON (e.department\_id = d.department\_id) ;

-- RIGHT OUTER JOIN

SELECT e.last\_name, e.department\_id, d.department\_name

FROM employees e RIGHT OUTER JOIN departments d

ON (e.department\_id = d.department\_id) ;

-- [ this is in oracle, it is not available in mysql]

-- FULL OUTER JOIN

-- SELECT e.last\_name, d.department\_id, d.department\_name

-- FROM employees e FULL OUTER JOIN departments d

-- ON (e.department\_id = d.department\_id) ;

-- Creating Cross Joins / Cartesian product

SELECT last\_name, department\_name

FROM employees

CROSS JOIN departments ;