Contents

[Information\_schema: 2](#_Toc143176020)

[Practice: 2](#_Toc143176021)

[Sub-querying: 3](#_Toc143176022)

[Selecting 2nd highest salary: 4](#_Toc143176023)

[Update with select: 4](#_Toc143176024)

[Referential Integrity: 4](#_Toc143176025)

# Information\_schema:

For tables:

* SELECT \* FROM information\_schema.tables
* SHOW FULL TABLES FROM `schema\_name` WHERE table\_type = 'BASE TABLE';
* Important columns in information\_schema.tables
  1. TABLE\_SCHEMA
  2. TABLE\_NAME
  3. TABLE\_ROWS

For columns

* SELECT \* FROM information\_schema.COLUMNS
* Important columns in information\_schema.columns
  1. TABLE\_SCHEMA
  2. TABLE\_NAME
  3. COLUMN\_NAME
  4. ORDINAL\_POSITION
  5. DATA\_TYPE

For views:

* SELECT \* FROM INFORMATION\_SCHEMA.views
* Important columns in information\_schema.views
  1. TABLE\_SCHEMA
  2. TABLE\_NAME
  3. VIEW\_DEFINITION
  4. IS\_UPDATEABLE
  5. DEFINER

For constraints:

* Select \* from information\_schema.table\_constraints
* Important columns in information\_schema.table\_constraints
  1. CONSTRAINT\_SCHEMA
  2. TABLE\_SCHEMA
  3. TABLE\_NAME
  4. CONSTRAINT\_TYPE

# Practice:

* We can group by on the basis of primary key, to get unique rows
* We cannot use alias name for ambiguous column in joins as filter on where clause because filter is executed first than select statement. Rather we have to specify the full column name (employees.name)  
    
  E.g.: we have 2 tables employees and department both containing name column and we have to select filter by department name.  
  we cannot write:   
  select e.name as employee\_name, d.name as department\_name from employees e inner join departments on e.department\_name =d.name where employee\_name=”ram”;  
    
  we write:  
  select e.name as employee\_name, d.name as department\_name from employees e inner join departments on e.department\_name =d.name where e.name=”ram”;
* Outer join is not available in mysql, instead union the result of left join and right join.
* Except(set operation) : A-B operation
* Semi join: return values of t1 for a condition met in t2. (by sub-querying)
* Anti-Join: return values of t1 for a condition not met in t2. (by sub-querying)

# Sub-querying:

* Inside SELECT:   
  SELECT NAME, (SELECT COUNT(\*) FROM departments) AS counting FROM employees;  
    
    
  select distinct department\_name, (select count(\*) from employees where employees.department\_name=departments.department\_name)   
  from departments;  
    
  SELECT DISTINCT DepartmentName,(SELECT COUNT(\*) FROM employees WHERE departments.departmentid=employees.departmentid) AS counting FROM departments ORDER BY departmentname;
* Inside where:  
    
  select \* from employees where departmentid in (Select departmentid from deparments where departmentname=’HR’);

select distinct countries.name as country,(select count(\*) from cities where cities.country\_code=countries.code) as cities\_num

from countries

left join cities

on countries.code=cities.country\_code

order by cities\_num desc, country asc

limit 9;

Inside from:

-- Select local\_name and lang\_num from appropriate tables

SELECT local\_name, sub.lang\_num

FROM countries,

    (SELECT code, COUNT(\*) AS lang\_num

     FROM languages

     GROUP BY code) AS sub

-- Where codes match

WHERE countries.code = sub.code

ORDER BY lang\_num DESC;

# Selecting 2nd highest salary:

* SELECT \* FROM (SELECT \*,DENSE\_RANK() OVER(ORDER BY salary DESC) AS ranking FROM salaries) testing WHERE ranking=2;
* WITH cc AS (  
   SELECT \*,DENSE\_RANK() OVER(ORDER BY salary) AS ranking FROM salaries)  
    
  SELECT \* FROM cc WHERE ranking=(SELECT MAX(ranking)-1 FROM cc);  
    
  or::  
    
    
   WITH cc AS (  
  SELECT \*,DENSE\_RANK() OVER(ORDER BY salary DESC) AS ranking FROM salaries)   
  SELECT \* FROM cc WHERE ranking=2;
* SELECT \* FROM salaries WHERE salary =(SELECT salary FROM salaries GROUP BY salary ORDER BY salary DESC LIMIT 1,1);

# Update with select:

**UPDATE test\_table**

**SET managerid =  
(SELECT managerid FROM employees WHERE employeeid=testing.employeeid);**

# Referential Integrity:

Alter table table\_name

Add foreign key(company\_id) references company(id) { **on update|delete restrict|cascade|set null|no action }**

* **Restrict or no action: Throw an error when deleting or changing**
* **Set null: on delete or update on the child table set the respective value on the parent column as null.**
* **Cascade: if deleted on the child table, parent table values are deleted. If updated, the parent values are updated**

# Remove duplicates from a table:

  
  




