-- 1> Write a SQL query to find the nth highest salary from employee table.

-- Example: finding 3rd highest salary from employee table

select \* from employee order by salary desc;

--- Limit N-1,1

select distinct salary from employee order by salary desc limit 2, 1;

-- 2> Write a SQL query to find top n records?

-- Example: finding top 5 records from employee table

select \* from employee order by salary desc limit 5;

-- 3> Write a SQL query to find the count of employees working in department 'Admin'

select count(\*) from employee where department = 'Admin';

-- 4> Write a SQL query to fetch department wise count employees sorted by department count in desc order.

select \* from employee;

select department, count(\*) as employeecount

from employee

group by department

order by employeecount desc;

-- 5> Write a query to fetch only the first name(string before space) from the FullName column of user\_name table.

select distinct(substring\_index(full\_names, ' ', 1)) first\_name from user\_name;

-- 6> Write a SQL query to find all the employees from employee table who are also managers

select e1.first\_name, e2.last\_name from employee e1

join employee e2

on e1.employee\_id = e2.manager\_id;

-- 7> Write a SQL query to find all employees who have bonus record in bonus table

select \* from employee;

select \* from bonus;

select \* from employee where employee\_id in (select employee\_ref\_id from bonus where employee.employee\_id = bonus.employee\_ref\_id);

-- 8> Write a SQL query to find only odd rows from employee table

select \* from employee where MOD(employee\_id,2)<>0;

-- 9> Write a SQL query to fetch first\_name from employee table in upper case

select upper(first\_name) as First\_Name from employee;

-- 10> Write a SQL query to get combine name (first name and last name) of employees from employee table

select concat(first\_name, ' ' ,last\_name) as Name from employee;

-- 11> Write a SQL query to print details of employee of employee 'Jennifer' and 'James'.

select \* from employee where first\_name in ('Jennifer', 'James');

-- 12> Write a SQL query to fetch records of employee whose salary lies between

select first\_name, last\_name, salary from employee where salary between 100000 and 500000;

-- 13> Write a SQL query to get records of employe who have joined in Jan 2017

select \* from employee;

select first\_name, last\_name, joining\_date from employee where year(joining\_date)=2017 and month(joining\_date) = 1;

-- 14> Write a SQL query to get the list of employees with the same salary

select e1.first\_name, e2.last\_name from employee e1, employee e2 where e1.salary = e2.salary and e1.employee\_id != e2.employee\_id;

-- 15> Write a SQL query to show all departments along with the number of people working there.

select \* from employee;

select department, count(\*) as 'Number of employees' from employee

group by department

order by count(department);

-- 16> Write a SQL query to show the last record from a table.

select \* from employee where employee\_id = (select max(employee\_id) from employee);

-- 17> Write a SQL query to show the first record from a table.

select \* from employee where employee\_id = (select min(employee\_id) from employee);

-- 18> Write a SQL query to get last five records from a employee table.

(select \* from employee order by employee\_id desc limit 5) order by employee\_id;

-- 19> Write a SQL query to find employees having the highest salary in each department.

select first\_name, last\_name, department, max(salary) as 'Max Salary'from employee group by department order by max(salary);

-- 20> Write a SQL query to fetch three max salaries from employee table.

select distinct salary from employee order by salary desc limit 3 ;

-- OR-----

select distinct Salary from employee e1 WHERE 3 >= (SELECT count(distinct Salary) from employee e2 WHERE e1.Salary <= e2.Salary) order by e1.Salary desc;

-- 21> Write a SQL query to fetch departments along with the total salaries paid for each of them.

select department, sum(salary) as 'Total Salary' from employee group by department order by sum(salary);

-- 22> Write a SQL query to find employee with highest salary in an organization from employee table.

select first\_name, last\_name from employee where salary = (select max(salary) from employee);

-- 23> Write an SQL query that makes recommendations using the pages that your friends liked.

-- Assume you have two tables: a two-column table of users and their friends, and a two-column table of

-- users and the pages they liked. It should not recommend pages you already like.

-- 24> write a SQL query to find employee (first name, last name, department and bonus) with highest bonus.

select first\_name, last\_name, department, max(bonus\_amount) from employee e

join bonus b

on e.employee\_id = b.employee\_ref\_id

group by department

order by max(bonus\_amount) desc limit 1;

-- 25> write a SQL query to find employees with same salary

select e1.first\_name, e1.last\_name, e1.salary from employee e1, employee e2

where e1.salary = e2.salary

and e1.employee\_id != e2.employee\_id;

-- 26> Write SQL to find out what percent of students attend school on their birthday from attendance\_events and all\_students tables?

select \* from all\_students;

select \* from attendance\_events;

select (count(attendance\_events.student\_id) \* 100 / (select count(student\_id) from attendance\_events)) as Percent

from attendance\_events

join all\_students

on all\_students.student\_id = attendance\_events.student\_id

where month(attendance\_events.date\_event) = month(all\_students.date\_of\_birth)

and day(attendance\_events.date\_event) = day(all\_students.date\_of\_birth);

-- 27> Given timestamps of logins, figure out how many people on Facebook were active all seven days

-- of a week on a mobile phone from login info table?

select \* from login\_info;

select a.login\_time, count(distinct a.user\_id) from

login\_info a

Left join login\_info b

on a.user\_id = b.user\_id

where a.login\_time = b.login\_time - interval 1 day

group by 1;

-- 28> Write a SQL query to find out the overall friend acceptance rate for a given date from user\_action table.

select \* from user\_action;

select count(a.user\_id\_who\_sent)\*100 / (select count(user\_id\_who\_sent) from user\_action) as percent

from user\_action a

join user\_action b

on a.user\_id\_who\_sent = b.user\_id\_who\_sent and a.user\_id\_to\_whom = b.user\_id\_to\_whom

where a.date\_action = '2018-05-24' and b.action = "accepted";

-- 29> How many total users follow sport accounts from tables all\_users, sport\_accounts, follow\_relation?

select \* from all\_users;

select \* from sport\_accounts;

select \* from follow\_relation;

select count(distinct c.follower\_id) as count\_all\_sports\_followers

from sport\_accounts a

join all\_users b

on a.sport\_player\_id = b.user\_id

join follow\_relation c

on b.user\_id = c.target\_id;

-- 30> How many active users follow each type of sport?

select b.sport\_category, count(a.user\_id)

from all\_users a

join sport\_accounts b

on a.user\_id = b.sport\_player\_id

join follow\_relation c

on a.user\_id = c.follower\_id

where a.active\_last\_month =1

group by b.sport\_category;

-- 31> What percent of active accounts are fraud from ad\_accounts table?

select \* from ad\_accounts;

select count(distinct a.account\_id)/(select count(account\_id) from ad\_accounts where account\_status= "active") as 'percent'

from ad\_accounts a

join ad\_accounts b

on a.account\_id = b.account\_id

where a.account\_status = 'fraud' and b.account\_status='active';

-- 32> How many accounts became fraud today for the first time from ad\_accounts table?

select count(account\_id) 'First time fraud accounts' from (

select distinct a.account\_id, count(a.account\_status)

from ad\_accounts a

join ad\_accounts b

on a.account\_id = b.account\_id

where b.date = curdate() and a.account\_status = 'fraud'

group by account\_id

having count(a.account\_status) = 1) ad\_accnt;

-- 33> Write a SQL query to determine avg time spent per user per day from user\_details and event\_session\_details

select \* from event\_session\_details;

select \* from user\_details;

select date, user\_id, sum(timespend\_sec)/count(\*) as 'avg time spent per user per day'

from event\_session\_details

group by 1,2

order by 1;

-- or --

select date, user\_id, avg(timespend\_sec)

from event\_session\_details

group by 1,2

order by 1;

-- 34> write a SQL query to find top 10 users that sent the most messages from messages\_detail table.

select \* from messages\_detail;

select user\_id, messages\_sent

from messages\_detail

order by messages\_sent desc

limit 10;

-- 35> Write a SQL query to find disctinct first name from full user name from usere\_name table

select \* from user\_name;

select distinct(substring\_index(full\_names, ' ', 1)) first\_name from user\_name;

-- 36> You have a table with userID, appID, type and timestamp. type is either 'click' or 'impression'.

-- Calculate the click through rate from dialoglog table. Now do it in for each app.

-- click through rate is defined as (number of clicks)/(number of impressions)

select \* from dialoglog;

select app\_id

, ifnull(sum(case when type = 'click' then 1 else 0 end)\*1.0

/ sum(case when type = 'impression' then 1 else 0 end), 0 )AS 'CTR(click through rate)'

from dialoglog

group by app\_id;

-- 37> Given two tables Friend\_request (requestor\_id, sent\_to\_id, time),

-- Request\_accepted (acceptor\_id, requestor\_id, time). Find the overall acceptance rate of requests.

-- Overall acceptate rate of requests = total number of acceptance / total number of requests.

select \* from friend\_request;

select \* from request\_accepted;

select ifnull(round(

(select count(\*) from (select distinct acceptor\_id, requestor\_id from request\_accepted) as A)

/

(select count(\*) from (select distinct requestor\_id, sent\_to\_id from friend\_request ) as B), 2),0

) as basic;

-- 38> from a table of new\_request\_accepted, find a user with the most friends.

select \* from new\_request\_accepted;

select id from

(

select id, count(\*) as count

from (

select requestor\_id as id from new\_request\_accepted

union all

select acceptor\_id as id from new\_request\_accepted) as a

group by 1

order by count desc

limit 1) as table1;

-- 39> from the table count\_request, find total count of requests sent and total count of requests sent failed

-- per country

select \* from count\_request;

select country\_code, Total\_request\_sent, Total\_percent\_of\_request\_sent\_failed,

cast((Total\_request\_sent\*Total\_percent\_of\_request\_sent\_failed)/100 as decimal) as Total\_request\_sent\_failed

from

(

select country\_code, sum(count\_of\_requests\_sent) as Total\_request\_sent,

cast(replace(ifnull(sum(percent\_of\_request\_sent\_failed),0), '%','') as decimal(2,1)) as Total\_percent\_of\_request\_sent\_failed

from count\_request

group by country\_code

) as Table1;

-- 40> create a histogram of duration on x axis, no of users on y axis which is populated by volume in each bucket

-- from event\_session\_details

select \* from event\_session\_details;

select floor(timespend\_sec/500)\*500 as bucket,

count(distinct user\_id) as count\_of\_users

from event\_session\_details

group by 1;

-- 41> Write SQL query to calculate percentage of confirmed messages from two tables :

-- confirmation\_no (phone numbers that facebook sends the confirmation messages to) and

-- confirmed\_no (phone numbers that confirmed the verification)

select round((count(confirmed\_no.phone\_number)/count(confirmation\_no.phone\_number))\*100, 2)

from confirmation\_no

left join confirmed\_no

on confirmed\_no.phone\_number= confirmation\_no.phone\_number;

-- 42> Write SQL query to find number of users who had 4 or more than 4 interactions on 2013-03-23 date

-- from user\_interaction table (user\_1, user\_2, date).

-- assume there is only one unique interaction between a pair of users per day

select \* from user\_interaction;

select table1.user\_id, sum(number\_of\_interactions) as Number\_of\_interactions

from

(

select user\_1 as user\_id, count(user\_1) as number\_of\_interactions from user\_interaction

group by user\_1

union all

select user\_2 as user\_id, count(user\_2) as number\_of\_interactions from user\_interaction

group by user\_2) table1

group by table1.user\_id

having sum(number\_of\_interactions) >= 4;

-- 43> write a sql query to find the names of all salesperson that have order with samsonic from

-- the table: salesperson, customer, orders

select s.name

from salesperson s

join orders o on s.id = o.salesperson\_id

join customer c on o.cust\_id = c.id

where c.name = 'Samsonic';

-- 44> write a sql query to find the names of all salesperson that do not have any order with Samsonic from the table: salesperson, customer, orders

select s.Name

from Salesperson s

where s.ID NOT IN(

select o.salesperson\_id from Orders o, Customer c

where o.cust\_id = c.ID

and c.Name = 'Samsonic');

-- 45> Wrie a sql query to find the names of salespeople that have 2 or more orders.

select s.name as 'salesperson', count(o.number) as 'number of orders'

from salesperson s

join orders o on s.id = o.salesperson\_id

group by s.name

having count(o.number)>=2;

-- 46> Given two tables: User(user\_id, name, phone\_num) and UserHistory(user\_id, date, action),

-- write a sql query that returns the name, phone number and most recent date for any user that has logged in

-- over the last 30 days

-- (you can tell a user has logged in if action field in UserHistory is set to 'logged\_on')

select user.name, user.phone\_num, max(userhistory.date)

from user,userhistory

where user.user\_id = userhistory.user\_id

and userhistory.action = 'logged\_on'

and userhistory.date >= date\_sub(curdate(), interval 30 day)

group by user.name;

-- 47> Given two tables: User(user\_id, name, phone\_num) and UserHistory(user\_id, date, action),

-- Write a SQL query to determine which user\_ids in the User table are not contained in the UserHistory table

-- (assume the UserHistory table has a subset of the user\_ids in User table). Do not use the SQL MINUS statement.

-- Note: the UserHistory table can have multiple entries for each user\_id.

select user.user\_id

from user

left join userhistory

on user.user\_id = userhistory.user\_id

where userhistory.user\_id is null;

-- 48> from a given table compare(numbers int(4)), write a sql query that will return the maximum value

-- from the numbers without using

-- sql aggregate like max or min

select numbers

from compare

order by numbers desc

limit 1;

-- 49> Write a SQL query to find out how many users inserted more than 1000 but less than 2000 images in their presentations from event\_log table

-- There is a startup company that makes an online presentation software and they have event\_log table that records every time a user inserted

-- an image into a presentation. one user can insert multiple images

select count(\*) from

(select user\_id, count(event\_date\_time) as image\_per\_user

from event\_log

group by user\_id) as image\_per\_user

where image\_per\_user <2000 and image\_per\_user>1000;

-- 50> select the most recent login time by values from the login\_info table

select \* from login\_info

where login\_time in (select max(login\_time) from login\_info

group by user\_id)

order by login\_time desc limit 1;