SQL PRACTICE INTERVIEW QUESTIONS:

by Akarshan Kapoor

Write a SQL query to fetch all the duplicate records from a table.

```
--Tables Structure:

drop table users;
create table users
(
user_id int primary key,
user_name varchar(30) not null,
email varchar(50));

insert into users values
(1, 'Sumit', 'sumit@gmail.com'),
(2, 'Reshma', 'reshma@gmail.com'),
(3, 'Farhana', 'farhana@gmail.com'),
(4, 'Robin', 'robin@gmail.com'),
(5, 'Robin', 'robin@gmail.com');

select * from users;

-- Solution 1:

-- Replace ctid with rowid for Oracle, MySQL and Microsoft SQLServer

select *
from users u
where u.ctid not in (
select min(ctid) as ctid
from users
group by user_name
order by ctid);


-- Solution 2: Using window function.

select user_id, user_name, email
from (
select *,
row_number() over (partition by user_name order by user_id) as rn
from users u
order by user_id) x
where x.rn <> 1;
```

Write a SQL query to fetch the second last record from an employee table.

```
--Tables Structure:

drop table employee;
create table employee
( emp_ID int primary key
, emp_NAME varchar(50) not null
```

```
, DEPT_NAME varchar(50)
, SALARY int);

insert into employee values(101, 'Mohan', 'Admin', 4000);
insert into employee values(102, 'Rajkumar', 'HR', 3000);
insert into employee values(103, 'Akbar', 'IT', 4000);
insert into employee values(104, 'Dorvin', 'Finance', 6500);
insert into employee values(105, 'Rohit', 'HR', 3000);
insert into employee values(106, 'Rajesh',  'Finance', 5000);
insert into employee values(107, 'Preet', 'HR', 7000);
insert into employee values(108, 'Maryam', 'Admin', 4000);
insert into employee values(109, 'Sanjay', 'IT', 6500);
insert into employee values(110, 'Vasudha', 'IT', 7000);
insert into employee values(111, 'Melinda', 'IT', 8000);
insert into employee values(112, 'Komal', 'IT', 10000);
insert into employee values(113, 'Gautham', 'Admin', 2000);
insert into employee values(114, 'Manisha', 'HR', 3000);
insert into employee values(115, 'Chandni', 'IT', 4500);
insert into employee values(116, 'Satya', 'Finance', 6500);
insert into employee values(117, 'Adarsh', 'HR', 3500);
insert into employee values(118, 'Tejaswi', 'Finance', 5500);
insert into employee values(119, 'Cory', 'HR', 8000);
insert into employee values(120, 'Monica', 'Admin', 5000);
insert into employee values(121, 'Rosalin', 'IT', 6000);
insert into employee values(122, 'Ibrahim', 'IT', 8000);
insert into employee values(123, 'Vikram', 'IT', 8000);
insert into employee values(124, 'Dheeraj', 'IT', 11000);

select * from employee;

-- Solution:

select emp_id, emp_name, dept_name, salary
from (
select *,
row_number() over (order by emp_id desc) as rn
from employee e) x
where x.rn = 2;
```

Write a SQL query to display only the details of employees who either earn the highest salary or the lowest salary in each department from the employee table.

```
--Tables Structure:

drop table employee;
create table employee
( emp_ID int primary key
, emp_NAME varchar(50) not null
, DEPT_NAME varchar(50)
, SALARY int);

insert into employee values(101, 'Mohan', 'Admin', 4000);
insert into employee values(102, 'Rajkumar', 'HR', 3000);
insert into employee values(103, 'Akbar', 'IT', 4000);
insert into employee values(104, 'Dorvin', 'Finance', 6500);
insert into employee values(105, 'Rohit', 'HR', 3000);
```

```
insert into employee values(106, 'Rajesh',  'Finance', 5000);
insert into employee values(107, 'Preet', 'HR', 7000);
insert into employee values(108, 'Maryam', 'Admin', 4000);
insert into employee values(109, 'Sanjay', 'IT', 6500);
insert into employee values(110, 'Vasudha', 'IT', 7000);
insert into employee values(111, 'Melinda', 'IT', 8000);
insert into employee values(112, 'Komal', 'IT', 10000);
insert into employee values(113, 'Gautham', 'Admin', 2000);
insert into employee values(114, 'Manisha', 'HR', 3000);
insert into employee values(115, 'Chandni', 'IT', 4500);
insert into employee values(116, 'Satya', 'Finance', 6500);
insert into employee values(117, 'Adarsh', 'HR', 3500);
insert into employee values(118, 'Tejaswi', 'Finance', 5500);
insert into employee values(119, 'Cory', 'HR', 8000);
insert into employee values(120, 'Monica', 'Admin', 5000);
insert into employee values(121, 'Rosalin', 'IT', 6000);
insert into employee values(122, 'Ibrahim', 'IT', 8000);
insert into employee values(123, 'Vikram', 'IT', 8000);
insert into employee values(124, 'Dheeraj', 'IT', 11000);

select * from employee;

-- Solution:

select x.*
from employee e
join (select *,
max(salary) over (partition by dept_name) as max_salary,
min(salary) over (partition by dept_name) as min_salary
from employee) x
on e.emp_id = x.emp_id
and (e.salary = x.max_salary or e.salary = x.min_salary)
order by x.dept_name, x.salary;
```

From the weather table, fetch all the records when London had extremely cold temperature for 3 consecutive days or more.

Note: Weather is considered to be extremely cold then its temperature is less than zero.

```
--Table Structure:

drop table weather;
create table weather
(
id int,
city varchar(50),
temperature int,
day date
);
delete from weather;
insert into weather values
(1, 'London', -1, to_date('2021-01-01','yyyy-mm-dd')),
(2, 'London', -2, to_date('2021-01-02','yyyy-mm-dd')),
(3, 'London', 4, to_date('2021-01-03','yyyy-mm-dd')),
```

```
(4, 'London', 1, to_date('2021-01-04','yyyy-mm-dd')),
(5, 'London', -2, to_date('2021-01-05','yyyy-mm-dd')),
(6, 'London', -5, to_date('2021-01-06','yyyy-mm-dd')),
(7, 'London', -7, to_date('2021-01-07','yyyy-mm-dd')),
(8, 'London', 5, to_date('2021-01-08','yyyy-mm-dd'));

select * from weather;

--Solution:

select id, city, temperature, day
from (
    select *,
        case when temperature < 0
              and lead(temperature) over(order by day) < 0
              and lead(temperature,2) over(order by day) < 0
        then 'Y'
        when temperature < 0
              and lead(temperature) over(order by day) < 0
              and lag(temperature) over(order by day) < 0
        then 'Y'
        when temperature < 0
              and lag(temperature) over(order by day) < 0
              and lag(temperature,2) over(order by day) < 0
        then 'Y'
        end as flag
    from weather) x
where x.flag = 'Y';
```

Finding n consecutive records where temperature is below zero. And table has a primary key.

```
--Table Structure:
drop table if exists weather cascade;
create table if not exists weather
        (
               id                                      int
        primary key,
               city                            varchar(50) not null,
               temperature int                                 not null,
               day                             date
        not null
        );

delete from weather;
insert into weather values
        (1, 'London', -1, to_date('2021-01-01','yyyy-mm-dd')),
        (2, 'London', -2, to_date('2021-01-02','yyyy-mm-dd')),
        (3, 'London', 4, to_date('2021-01-03','yyyy-mm-dd')),
        (4, 'London', 1, to_date('2021-01-04','yyyy-mm-dd')),
        (5, 'London', -2, to_date('2021-01-05','yyyy-mm-dd')),
        (6, 'London', -5, to_date('2021-01-06','yyyy-mm-dd')),
        (7, 'London', -7, to_date('2021-01-07','yyyy-mm-dd')),
        (8, 'London', 5, to_date('2021-01-08','yyyy-mm-dd')),
        (9, 'London', -20, to_date('2021-01-09','yyyy-mm-dd')),
        (10, 'London', 20, to_date('2021-01-10','yyyy-mm-dd')),
        (11, 'London', 22, to_date('2021-01-11','yyyy-mm-dd')),
```

```
        (12, 'London', -1, to_date('2021-01-12','yyyy-mm-dd')),
        (13, 'London', -2, to_date('2021-01-13','yyyy-mm-dd')),
        (14, 'London', -2, to_date('2021-01-14','yyyy-mm-dd')),
        (15, 'London', -4, to_date('2021-01-15','yyyy-mm-dd')),
        (16, 'London', -9, to_date('2021-01-16','yyyy-mm-dd')),
        (17, 'London', 0, to_date('2021-01-17','yyyy-mm-dd')),
        (18, 'London', -10, to_date('2021-01-18','yyyy-mm-dd')),
        (19, 'London', -11, to_date('2021-01-19','yyyy-mm-dd')),
        (20, 'London', -12, to_date('2021-01-20','yyyy-mm-dd')),
        (21, 'London', -11, to_date('2021-01-21','yyyy-mm-dd'));
COMMIT;

select * from weather;


-- solution:
with
        t1 as
                (select *,      id - row_number() over (order by id) as diff
                from weather w
                where w.temperature < 0),
        t2 as
                (select *,
                count(*) over (partition by diff order by diff) as cnt
                from t1)
select id, city, temperature, day
from t2
where t2.cnt = 3;
```

Finding n consecutive records where temperature is below zero. And table does not have primary key.

```
create or replace view vw_weather as
select city, temperature from weather;

select * from vw_weather ;

-- solution:

with
        w as
                (select *, row_number() over () as id
                from vw_weather),
        t1 as
                (select *,      id - row_number() over (order by id) as diff
                from w
                where w.temperature < 0),
        t2 as
                (select *,
                count(*) over (partition by diff order by diff) as cnt
                from t1)
select city, temperature, id
from t2
where t2.cnt = 5;
```

Finding n consecutive records with consecutive date value.

```
--Table Structure:
drop table if exists orders cascade;
create table if not exists orders
  (
    order_id    varchar(20) primary key,
    order_date  date        not null
);

delete from orders;
insert into orders values
  ('ORD1001', to_date('2021-Jan-01','yyyy-mon-dd')),
  ('ORD1002', to_date('2021-Feb-01','yyyy-mon-dd')),
  ('ORD1003', to_date('2021-Feb-02','yyyy-mon-dd')),
  ('ORD1004', to_date('2021-Feb-03','yyyy-mon-dd')),
  ('ORD1005', to_date('2021-Mar-01','yyyy-mon-dd')),
  ('ORD1006', to_date('2021-Jun-01','yyyy-mon-dd')),
  ('ORD1007', to_date('2021-Dec-25','yyyy-mon-dd')),
  ('ORD1008', to_date('2021-Dec-26','yyyy-mon-dd'));
COMMIT;

select * from orders;


-- Solution
with
  t1 as
               (select *, row_number() over(order by order_date) as rn,
                order_date - cast(row_number() over(order by
order_date)::numeric as int) as diff
               from orders),
        t2 as
               (select *, count(1) over (partition by diff) as cnt
               from t1)
select order_id, order_date
from t2
where cnt >= 3;
```
