# SQL (Hard)
# 10+1 bonus questions

Practice here: sql-practice

## *Question 1:*

Show all of the patients grouped into weight groups and number of patients in each weight group.

Order the list by weight group descending.

For example, if they weigh from:

1. 100 to 109 (both inclusive), they are placed in the weight group - 100
2. 110 to 119 (both inclusive), they are placed in the weight group - 110

## *Query:*

```
SELECT
-- Step 1: FLOOR() to group patients based on given weight group conditions
-- Step 2: COUNT() to get number of patients in each weight group
FLOOR(weight/10)*10 AS weight_group,COUNT(*) AS no_of_patients
FROM patients
GROUP BY weight_group
-- Step 3: ORDER BY to sort by weight group in descending order
ORDER BY weight_group DESC;
```

## *Question 2:*

Show patient_id, weight, height, isObese. Display isObese as a boolean i.e. 0 or 1.

Obese is defined as weight(kg)/(height(m)^2) >= 30.

NOTE:
weight is in units kg
height is in units cm

## *Query:*

```
SELECT
patient_id, weight, height,
-- Step 2: CASE to identify if patient is obese or not
CASE
-- Step 1: POWER() to find square of height in meters
WHEN (weight/POWER((height*0.01),2))>=30 THEN 1
ELSE 0
END AS isObese
FROM patients;
```

## *Question 3:*

Show patient_id, first_name, last_name, and attending doctor's specialty.

Show only the patients who have a diagnosis as 'Epilepsy' and the doctor's first name is 'Lisa'.

Check patients, admissions, and doctors tables for required information.

## *Query:*

```
SELECT
pat.patient_id,pat.first_name,pat.last_name,doc.specialty
-- Step 1: JOIN patients and admissions table based on common column patient_id
FROM patients AS pat
JOIN admissions AS adm
ON pat.patient_id=adm.patient_id
-- Step 2: JOIN admissions and doctors table based on common column doctor_id
-- Note: As we don't have a common column between patients and doctors table, we bring in
admissions table
JOIN doctors AS doc
ON adm.attending_doctor_id=doc.doctor_id
-- Step 3: WHERE to filter diagnosis and doctor's first name based on given conditions
WHERE adm.diagnosis='Epilepsy' AND doc.first_name='Lisa';
```

## *Question 4:*

All patients who have gone through admissions, can see their medical documents on our site. Those patients are given a temporary password after their first admission. Show patient_id and temp_password.

The temp_password must be the following, in order:
1. patient_id
2. numerical length of patient's last_name
3. year of patient's birth_date

## *Query:*

```
SELECT
--Step 3: LEN() to get numerical length of patient's last_name
--Step 4: YEAR() to get year from patient's birth_date
--Step 5: CONCAT() to concatenate required columns
patient_id, CONCAT(patient_id,LEN(last_name),YEAR(birth_date)) AS temp_password
FROM patients
-- Step 1: WHERE to filter only patients who were admitted i.e. patient_id is present in both
patients and admissions table
WHERE patient_id IN
-- Step 2: subquery to get patient_id from admissions table
(SELECT patient_id FROM admissions);
```

## *Question 5:*

Each admission costs $50 for patients without insurance and $10 for patients with insurance. All patients with an even patient_id have insurance.

Give each patient a 'Yes' if they have and 'No' if they don't have insurance. Add up to show admission_total_cost for each has_insurance group.

## *Query:*

```
-- Step 4: SUM() to add up admission_cost based on has_insurance
SELECT has_insurance,SUM(admission_cost) AS admission_total_cost FROM
(
SELECT
-- Step 2: CASE to identify if patient has insurance or not
CASE
WHEN patient_id%2==0 THEN 'Yes'
ELSE 'No'
END AS has_insurance,
-- Step 3: CASE to assign corresponding admission_cost
CASE
WHEN patient_id%2==0 THEN '10'
ELSE '50'
END AS admission_cost
FROM admissions
-- Step 1: subquery to get has_insurance and admission_cost
) AS subquery_has_insurance_admission_cost
GROUP BY has_insurance;
```

## *Question 6:*

Show provinces that have more patients identified as 'M' than 'F'

Display only the full province_name

## *Query:*

```
SELECT province_name FROM
(
SELECT
province_name,
-- Step 3: SUM() to get number_of_male_patients
SUM(gender='M') AS number_of_male_patients,
-- Step 4: SUM() to get number_of_female_patients
SUM(gender='F') AS number_of_female_patients
FROM patients AS pat
-- Step 2: JOIN patients and province_names tables based on common column province_id
JOIN province_names AS pvn
ON pat.province_id=pvn.province_id
GROUP BY province_name
-- Step 1: subquery to get number of male and female patients in each province
) AS subquery_number_of_patients
-- Step 5: WHERE to filter only province_name that have more male patients than female
patients
WHERE number_of_male_patients>number_of_female_patients;
```

## *Question 7:*

We are looking for a specific patient.

Pull all columns for the patient who matches all criteria:
- First_name contains an 'r' after first two letters
- Identifies their gender as 'F'
- Born in February, May or December
- Their weight is between 60 and 80 kg
- Their patient_id is an odd number
- They are from the city 'Kingston'

## *Query:*

```
-- Step 2: SELECT * to get all columns from patients table
SELECT * FROM patients
-- Step 1: WHERE and AND to find the patient who matches all given criteria
WHERE first_name LIKE '__r%'
AND gender='F'
AND MONTH(birth_date) IN (2,5,12)
AND weight BETWEEN 60 AND 80
AND patient_id%2!=0
AND city='Kingston';
```

## *Question 8:*

Show percent of patients that have 'M' as their gender

Round the answer to nearest hundredth number and represent in percent form

## *Query:*

```
SELECT
-- Step 1: SUM() to get count of male patients
-- Step 2: COUNT() to get count of all patients
-- Step 3: ROUND() to round to nearest hundredth number
-- Step 4: CONCAT() to represent the number in percent form
CONCAT(ROUND((SUM(gender='M')/(COUNT(*)*1.0))*100,2),'%')
AS percentage_of_male_patients
FROM patients;
```

## *Question 9:*

For each day display total amount of admissions on that date

Display amount changed from previous date

## *Query:*

```
SELECT *,
-- Step 3: LAG() to get admissions on previous date
-- Step 4: subtract admissions on current and previous date to get change_from_previous_date
admissions_on_current_date-LAG(admissions_on_current_date,1) OVER(ORDER BY
admission_date)
AS change_from_previous_date
FROM
(
SELECT
-- Step 2: COUNT() to get admissions_on_current_date
admission_date,COUNT(*) AS admissions_on_current_date
FROM admissions
GROUP BY admission_date
-- Step 1: subquery to get admissions_on_current_date
) AS subquery_admissions_on_current_date;
```

## *Question 10:*

Sort province names in ascending order in such a way that the province 'Ontario' is always on top

## *Query:*

```
-- Step 1: display 'Ontario' on top always
SELECT province_name FROM province_names WHERE province_name='Ontario'
-- Step 3: UNION ALL to combine the outputs
UNION ALL
-- Step 2: get all other province_name except 'Ontario'
SELECT province_name FROM province_names WHERE province_name!='Ontario';
```

## *Bonus question 1:*

Show the employee's first_name, last_name, a "num_orders" column with a count of orders taken and a column called "Shipped" that displays "On Time" if the order shipped on time and "Late" if the order shipped late.

Order by employee last_name, then by first_name and then descending by number of orders.

NOTE:
An order is considered to be delivered 'On Time' if required_date>shipped_date

## *Query:*

```
SELECT
first_name,last_name,
-- Step 2: COUNT() to get num_orders
COUNT(*) AS num_orders,
-- Step 3: CASE to identify 'On Time' and 'Late' delivery of orders
CASE
WHEN required_date>shipped_date THEN 'On Time'
ELSE 'Late'
END
AS Shipped
FROM employees AS emp
-- Step 1: JOIN to combine employees and orders table based on common column i.e.
employee_id
JOIN orders AS ord
ON emp.employee_id=ord.employee_id
GROUP BY first_name,last_name,Shipped
ORDER BY last_name,first_name,num_orders DESC;
```