Le SQL aux concours : Corrigé.

1. X 2016

Question 1.

```
SELECT id1 FROM LIENS WHERE id2 = x
```

Question 2.

```
SELECT nom, prenom FROM INDIVIDUS JOIN LIENS ON id=id1 WHERE id2 = x
```

Question 3.

```
SELECT DISTINCT a.id1 FROM liens a JOIN liens b ON a.id2=b.id1 WHERE b.id2 = x AND a.id1 != x
```

Remarque : DISCTINCT pour n'avoir que des résultats différents, et la dernière condition de sélection pour ne pas retrouver x dans les résultats.

2. Centrale 2016

Question 1.

```
SELECT COUNT(*) FROM vol WHERE jour = "2016-05-02" AND heure <= "12:00"
```

Question 2.

```
SELECT id_vol FROM vol JOIN aeroport ON depart = id_aero WHERE ville = "Paris" AND jour = "2016-05-02"
```

Question 3. Elle sélectionne les identifiants des vols internes à la France, décollant le 5 mai 2016.

Question 4.

```
SELECT a.id_vol, b.id_vol
FROM vol a JOIN vol b
ON a.depart = b.arrivee AND a.arrivee = b.depart
WHERE a.jour = b.jour AND a.niveau = b.niveau
```

Remarque : on aurait pu mettre toutes les conditions dans ON, ou dans WHERE... C'est un peu au choix !

3. Mines 2016

Question 1. Aucun attribut seul ne peut servir de clé primaire ici, car pour chacun d'entre eux, il y a au moins deux enregistrements avec des valeurs égales. *A priori*, le couple (`iso`, `annee`) est un bon candidat pour servir de clé primaire.

Question 2.

```
SELECT * FROM palu WHERE annee = 2010 AND deces >= 1000
```

Question 3.

```
SELECT iso, 100000*cas/pop AS taux_ind
FROM palu JOIN demographie ON iso = pays AND periode = annee
WHERE annee = 2011
```

Question 4. Si on connaît LIMIT et OFFSET

```
SELECT nom FROM palu WHERE annee = 2010 ORDER BY cas DESC LIMIT 1 OFFSET 1
```

LIMIT va limiter le nombre de résultats à un seul. OFFSET permet d'ignorer le premier résultat. Ainsi, on ne récupère que le deuxième enregistrement par ordre de nombre de cas décroissant. Si on ne connaît pas, il faut faire des requêtes imbriquées...

```
SELECT nom FROM palu WHERE annee = 2010 AND cas = (
   SELECT MAX(cas) FROM palu WHERE annee = 2010 AND cas != (
      SELECT MAX(cas) FROM palu WHERE annee = 2010
   )
)
```

Question 5. On rappelle que π est une projection, σ est une sélection. On garde les noms des pays et nombre de décès dus au paludisme en 2010. L'instruction suivante trie par ordre croissant de nombre de décès :

```
deces2010.sort(key=lambda x:x[1])
```

4. X 2017

Question 1.

```
SELECT idensemble FROM POINTS JOIN MEMBRE ON id_point = id WHERE x=a AND y=b
```

Question 2.

```
SELECT x,y
FROM POINTS JOIN MEMBRE a JOIN MEMBRE b ON id = a.idpoint AND id=b.idpoint
WHERE a.idensemble = i AND b.idensemble = j
```

Question 3. Avec DISTINCT pour ne récupérer qu'une fois un point qui appartient à plusieurs ensembles en commun avec (a, b).

```
SELECT DISTINCT p1.x, p1.y
FROM POINTS p1 JOIN POINTS p2 JOIN MEMBRE m1 JOIN MEMBRE m2
ON m1.id_point = p1.id AND m2.id_point = p2.id AND m1.idensemble = m2.idensemble
WHERE p2.x = a AND p2.y = b
```

5. Centrale 2017

Question 1. Les considérations sur les champs non renseignés (NULL) sont hors programme. Pour répondre à cette question, il faut savoir qu'il ne faut jamais tester l'égalité (ou l'inégalité) avec NULL (qui est une absence de valeur). La bonne solution est d'utiliser IS NULL / IS NOT NULL.

```
SELECT EX_NUM FROM EXPLO WHERE EX_FIN IS NULL AND EX_DEB IS NOT NULL
```

Question 2. Avec `nb` le numéro de l'exploration.

```
SELECT PI_NUM, PI_X, PI_Y FROM PI WHERE EX_NUM = nb
```

Question 3. On ne considère ici que les zones déja explorées en entier (EX_FIN non nul), ce qui oblige à faire une jointure. Pour le calcul d'une zone, il faut naturellement utiliser les fonctions d'agrégation MIN et MAX, et effectuer un regroupement sur EX_NUM. On n'oublie pas de convertir les millimètres carrés en mètres carrés...

```
SELECT EX_NUM, (MAX(PI_X)-MIN(PI_X))*(MAX(PI_Y)-MIN(PI_Y))/1000000
FROM PI NATURAL JOIN EXPLO
WHERE EX_FIN IS NOT NULL
GROUP BY EX_NUM
```

Remarque : une jointure naturelle était adaptée ici.

Question 4. Cette question est assez floue car on n'a aucune information sur la manière dont sont représentées les données : on sait simplement que l'on travaille avec des entiers en millimètres pour les coordonnées d'exploration. En supposant qu'ils sont représentés sur 32 bits en complément à 2, le plus grand entier représentable est (à 1 près) 2^{31}. On obtient donc une zone d'exploration maximale de 2^{62} millimètres carrés car les entiers sont positifs. Sachant qu'un kilomètre carré fait $10^{12} = 10^{3\times4} \simeq 2^{40}$ millimètres carrés, on obtient une zone maximale de 2^{22} kilomètres carrés, soit 4 millions de kilomètres carrés (de quoi fatiguer le robot : cela fait 8 fois la France...)

Question 5. Toutes les tables ne sont pas utiles. On utilise deux fonctions d'agrégation : comptage (COUNT) et somme (SUM), en regroupant sur IN_NUM. On utilise la requête de la question 1 pour obtenir le numéro de l'exploration en cours.

```
SELECT IN_NUM, COUNT(*), SUM(IT_DUR)
FROM EXPLO NATURAL JOIN ANALY NATURAL JOIN INTYP
WHERE EX_FIN IS NULL AND EX_DEB IS NOT NULL
GROUP BY IN_NUM
```

6. Mines 2017

Question 1.

```
SELECT id_croisement_fin FROM Voie WHERE id_croisement_debut = c
```

Question 2.

```
SELECT longitude, latitude
FROM Voie v JOIN Croisement cr ON v.id_croisement_fin = cr.id
WHERE v.id_croisement_debut = c
```

Question 3. Les identifiants des croisements atteignables en deux voies exactement à partir de c.