- Afficher le nombre de pays en Asie.

Requêter le nombre de pays en Asie revient à appliquer un filtre avec la méthode *Where* puis à appeler la méthode *Count* sur la collection filtrée. Le prédicat qui définit le filtre est une méthode anonyme précisant quel contenu rendra l'enregistrement éligible.

```
// Requête retournant le nombre de pays en Asie.
int nbPaysEnAsie = tabPays.Where(p => p.Continent == "Asie").Count();
Console.WriteLine($"Il y a {nbPaysEnAsie} pays en Asie");
```

La méthode anonyme qui précise le filtre utilise directement des types *string*. On dispose donc de ses méthodes pour notamment traiter des problèmes de casse potentiels dans la saisie : ASIE, asie ou Asie. Une solution consiste à passer les deux côtés en majuscules.

```
nbPaysEnAsie = tabPays.Where(p => p.Continent.ToUpper() == "ASIE").Count();
```

- Afficher en ordre alphabétique les noms et les superficies des pays asiatiques.

Pour afficher les noms et superficies de tous les pays asiatiques, on va appliquer le même filtre, demander un ordonnancement sur les noms des pays puis utiliser la méthode *Select* pour définir ce que l'on souhaite garder. Dans cette première version, on décide de garder les instances "au complet".

```
var paysAsiatiques = from pays
                     in tabPays
                     where pays.Continent == "Asie"
                     orderby pays.Nom
                     select pays;
```

Dans la boucle d'affichage du résultat, on accède aux propriétés de la classe *Pays*.

Si maintenant on ne souhaite sélectionner que les propriétés qui nous intéressent, on peut utiliser une classe anonyme comme ceci :

```
var paysAsiatiques_ = from pays
                      in tabPays
                      where pays.Continent == "Asie"
                      orderby pays.Nom
                      select new
                         { nomPays=pays.Nom,
                           superf =pays.Superficie};
```

Les noms des propriétés de la classe anonyme sont utilisés dans la boucle d'affichage qui suit : plus rien à voir avec ceux de la classe *Pays*. Attention, la portée de cette classe anonyme est la méthode elle-même.

- Calculer et afficher la surface totale de l'Asie.

La question suivante revient à faire le cumul de toutes les surfaces retournées par la requête. Pour cela, à la suite du filtre, on utilise *Sum* et une expression lambda désignant la propriété intervenant dans le calcul.

```
int surfaceAsie = tabPays.Where(p => p.Continent == "Asie")
                         .Sum(p => p.Superficie);
```

La base de données n'ayant pas intégré la Russie et ses 17 075 200 km2 comme faisant partie du continent asiatique, le résultat retourné peut différer de celui affiché sur d'autres sites.

- Afficher en ordre alphabétique tous les pays d'Asie dont le nom commence par A.

Récupérer la liste des pays d'Asie dont le nom commence par la lettre A revient à compléter la clause *Where* avec une nouvelle condition sur le premier caractère de la propriété *Nom*. Rappelons que nous travaillons toujours avec des objets de type *string* et que nous pouvons utiliser toutes les méthodes supportées !

```
var paysAsiatiquesWithA =   from pays
                        in tabPays
                        where pays.Continent == "Asie"
                        && pays.Nom[0] == 'A'
                        orderby pays.Nom
                        select pays;
```

Ce premier code se concentre sur l'essentiel. Pour éviter une exception et la perte d'enregistrements, un code plus élaboré vérifiera que la propriété *Nom* est renseignée et s'affranchira d'un éventuel problème de casse comme ceci :

```
paysAsiatiquesWithA = from pays
         in tabPays
    where pays.Continent == "Asie"
       && !String.IsNullOrEmpty(pays.Nom)
       && pays.Nom.ToUpper()[0] == 'A'
    orderby pays.Nom
    select pays;
```

- Calculer et afficher la surface totale des pays d'Asie dont le nom commence par A.

On reprend en partie la requête précédente et on la complète avec la méthode *Sum*.

```
long surfaceTotalePaysAsiatiquesWithA = tabPays.Where(p => p.Continent == "Asie" &&
p.Nom[0] == 'A')
                                              .Sum(p_ => p_.Superficie);
```

- Afficher le nom du pays d'Asie ayant la plus grande superficie et celui ayant la plus petite superficie.

Pour afficher le plus grand et le plus petit pays d'Asie, on peut simplement demander à la collection filtrée un classement par superficie et afficher le premier et le dernier.

```
var lePlusGrandPays = tabPays.Where(p => p.Continent == "Asie")
                             .OrderBy(p => p.Superficie).Last().Nom;
Console.WriteLine($"Le plus grand pays d'Asie est {lePlusGrandPays}");

var lePlusPetitPays = tabPays.Where(p=>p.Continent == "Asie")
                             .OrderBy(p => p.Superficie).First().Nom;
Console.WriteLine($"Le plus petit pays d'Asie est {lePlusPetitPays}");
```

- Afficher les noms de chaque continent suivis par les noms des pays qui les constituent.

Nous allons utiliser l'opérateur *Group By* pour regrouper les enregistrements par continent et lister les noms de chaque pays le constituant. Le retour d'un *Group By* est un objet de type *IEnumerable<IGrouping<K, T>>*.

La lettre *K* de l'interface *IGrouping* représente un template qui est la clé du dictionnaire. Dans notre cas, ce sera un objet de type *string* qui contiendra le nom du continent. La lettre *T* représente un *IEnumerable* de template qui recevra la collection rattachée à la clé. Dans notre cas, cette liste contiendra les noms des pays du continent clé. Et tout ça va se faire en une ligne…

```
var groupes = tabPays.GroupBy(p => p.Continent, p => p.Nom);
```

Le premier paramètre du *GroupBy* est une expression lambda créant la clé de regroupement. Cette clé est la propriété *Continent*. Le second paramètre est également une expression lambda permettant de sélectionner les propriétés. Pour l'instant, seul le nom du pays nous importe.

L'affichage de chaque continent suivi de tous ses pays s'effectue à l'aide de deux boucles d'itération imbriquées.

```
foreach (var groupe in groupes)
{
    Console.WriteLine($"{groupe.Key}");

    foreach (var nomPays in groupe)
    {
        Console.WriteLine($"\t{nomPays}");
    }
}
```

- Compléter l'affichage précédent par les surfaces de chaque continent et de chacun de ces pays.

Le nom du continent est directement accessible par la propriété *Key* de l'interface *IGrouping*. Les noms de chaque pays sont accessibles directement, car le second paramètre de l'opérateur *GroupBy* n'a sélectionné que la propriété *Nom*, mais nous devons compléter la requête en récupérant également les superficies. Pour cela, on va simplement modifier ce second paramètre en créant un objet anonyme contenant la propriété *Nom* et la propriété *Superficie*.

```
var groupes2 = tabPays.GroupBy(p => p.Continent,
    p => new { nomPays = p.Nom, superficie = p.Superficie });
```

L'itération sur le résultat est complétée par l'affichage de la somme totale des superficies de chaque pays et du détail par pays.

```
foreach (var groupe2 in groupes2)
{
    Console.WriteLine($"{groupe2.Key} : {groupe2.Sum(d => d.superficie)} km2 au
total");

    foreach (var paysSurface in groupe2)
    {
        Console.WriteLine($"\t{paysSurface.nomPays} = {paysSurface.superficie}
km2");
    }
}
```

Dans la seconde itération, contrairement à la requête précédente, il faut préciser les noms des propriétés de l'objet anonyme.

- Afficher le nom du plus grand continent avec sa superficie.
- Afficher le nom du plus petit continent avec sa superficie.

Enfin, pour récupérer le plus grand et le plus petit continent, nous allons encore utiliser l'opérateur *GroupBy*. Cette fois-ci, on va le faire suivre par l'opérateur *Select* pour travailler directement le résultat du regroupement en créant un objet anonyme contenant la clé : le nom du continent et la somme de toutes les surfaces trouvées dans ce regroupement. Le résultat du *Select* est un objet de type *IEnumerable* que l'on peut directement classer en utilisant *OrderByDescending* sur la propriété contenant la somme des surfaces. Le résultat de *OrderByDescending* est une collection de type *IOrderedEnumerable* sur laquelle nous pouvons utiliser *First* ou *Last* pour récupérer le premier et le dernier de la liste. Cet élément de la liste est un objet anonyme contenant le nom du continent et sa superficie. On peut donc directement accéder à ses propriétés.

```
var lePlusGrandContinent = tabPays
    .GroupBy(p => p.Continent, p => p.Superficie)
    .Select(p => new {continent = p.Key, surfaceTotale = p.Sum()})
    .OrderByDescending(p=>p.surfaceTotale)
    .First();

Console.WriteLine($"Le plus grand continent est {lePlusGrandContinent.continent}");
```