Bazy danych

Korzystając np. ze strony: http://sqlfiddle.com proszę utworzyć następującą tabelę i zasilić ją danymi (przycisk Build schema):

```
create table product (id integer, category varchar(50), quantity integer, techdata varchar(100), cost_price float);

insert into product

values

(1,'window',2,'<Data w="1000" h="1000"/>',100.56),

(2,'door',1,'<Data w="900" h="1800"/>',96.12),

(3,'window',20,'<Data w="750" h="300"/>',152.5),

(4,'door',100,'<Data w="1046" h="2046"/>',46.74),

(5,'window',1,null,null);
```

, gdzie cost_price jest ceną kosztów w PLN za 1 sztukę,

atrybuty w i h to odpowiednio szerokość i wysokość produktu

Następnie proszę przygotować zapytania SQL:

Zwróć listę unikalnych kategorii produktów

Zwróć łączna ilość produktów o kategorii ‘window’

Zwróć produkt o najwyższym koszcie za sztukę

Zwróć produkt o najwyższej wartości kosztów

Zwróć produkty których wartość kosztów jest większa niż 400 PLN

Zwróć łączną wartość kosztów produktów z podziałem na kategorie

Zwróć wartość obwodu każdego produktu na podstawie danych xml

Programowanie:

Korzystając np. ze strony https://www.onlinegdb.com/online_c++_compiler

Proszę uruchomić poniższy kod:

```
#include <iostream>
using namespace std;
int main()
{
   std::string s1 = "1;10;3;4";
   std::string s2 = "11->12->33->14";
   std::string delimiter = ";";
   size_t pos = 0;
   int sum1 = 0;
   int sum2 = 0;
   int sumAll = 0;
   while ((pos = s1.find(delimiter)) != std::string::npos) {
      sum1 = sum1 + atoi(s1.substr(0, pos).c_str());
      s1.erase(0, pos + delimiter.length());
   }

   std::cout << std::to_string(sum1) << std::endl;
   delimiter = "->";
   while ((pos = s2.find(delimiter)) != std::string::npos) {
      sum2 = sum2 + atoi(s2.substr(0, pos).c_str());
```

```
        s2.erase(0, pos + delimiter.length());
    }

    std::cout << std::to_string(sum2) << std::endl;
    sumAll = sum1 + sum2;
    std::cout << std::to_string(sumAll) << std::endl;
}
```

Poprawić błąd w programie, aby zwracał prawidłową sumę liczb.

Zminimalizować kod programu , poprzez unikanie zdublowanych fragmentów kodu

Napisać dodatkowe metody testujące poprawność algorytmu dla 4 różnych przypadków testowych: „1;2;3;4;5” = 15, „10->20” =30, „100” =100, „”=0,

Zamiast gotowego kodu można również przesłać opisową odpowiedź gdzie jest błąd i jak można zoptymalizować kod.