Challenge March 2024

Analyzing Employees

A solution with OPL by Alex Fleischer afleischer@fr.ibm.com

OPL (Optimization Programming Language) is an abstract modeling language that helps model easily optimization problems that can be solved both with IBM CPLEX linear programming and IBM CPLEX constraint programming CPOptimizer (CPO)

In this challenge we won't use any optimization technique but only the data part within OPL.

First we will convert JSON to .dat and then use .dat as input data. .dat is the data format for OPL CPLEX;

We call readjson.py at

https://github.com/AlexFleischerParis/opltipsandtricks/blob/master/readson.py

in order to turn employees.json into employees.dat

```
employees={
<"Robinson",25,"Female","Married",False,"[{'id': 'RobinsonLoc1', 'street': 'Main
Str', 'zipCode': '08831', 'state': 'NJ'}, {'id': 'RobinsonLoc2', 'street': 'Ocean
Drive', 'zipCode': '33019', 'state': 'FL'}]",2,220000.0,>,
<"Warner",45,"Male","Married",False,"[{'id': 'WarnerLoc', 'street': 'Maple
Street', 'zipCode': '08817', 'state': 'NJ'}]",0,150000.0,>,
<"Stevens",24,"Male","Single",False,"[{'id': 'StevensLoc', 'street': 'Regency
Drive', 'zipCode': '08817', 'state': 'NJ'}]",0,135000.0,>,
<"White",32,"Female","Married",False,"[{'id': 'WhiteLoc', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}]",2,195000.0,>,
<"Smith",46,"Male","Single",False,"[{'id': 'SmithLoc', 'street': 'Maple Street',
'zipCode': '90027', 'state': 'CA'}]",1,120000.0,>,
<"Green",28,"Female","Married",False,"[{'id': 'GreenLoc', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}]",3,140000.0,>,
<"Brown",32,"Male","Married",False,"[{'id': 'BrownLoc', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}]",2,65000.0,>,
<"Klaus",54,"Male","Married",False,"[{'id': 'KlausLoc', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}]",1,85000.0,>,
<"Houston",47,"Female","Single",False,"[{'id': 'HoustonLoc', 'street': 'Green
Road', 'zipCode': '33019', 'state': 'FL'}]",2,135000.0,>,
<"Long",29,"Male","Married",False,"[{'id': 'LongLoc', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}]",3,40000.0,>,
<"Short",22,"Male","Single",False,"[{'id': 'ShortLoc', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}]",0,120000.0,>,
<"Doe",21,"Female","Single",False,"[{'id': 'DoeLoc1', 'street': 'Green Road',
'zipCode': '33019', 'state': 'FL'}, {'id': 'DoeLoc2', 'street': 'Morgan Street',
```

```
'zipCode': '33020', 'state': 'FL'}, {'id': 'DoeLoc3', 'street': 'Lyric Ave',
'zipCode': '90027', 'state': 'CA'}]",1,210000.0,>,
};
```

And then we write the OPL model employees.mod

```
tuple employee
{
  key string name;
  int age;
  string gender;
  string maritalstatus;
  string minor;
  string locations;
  int children;
  float salary;
}

{string} maritalstatuses={"Single","Married"};

// tuple set for employees
{employee} employees with maritalstatus in maritalstatuses= ...;

{string} selectedZipCode={"08817","90027"};

// zips and states per employees
{string} zips[employees];
{string} states[employees];

// compute zips and states per employees with javascript in OPL
execute
{
  for(var e in employees)
   {
     var loc=e.locations;

     // Read zips
     var i=0;
     while (i!=-1)
     {
       i=loc.indexOf("zipCode",i);
       if (i!=-1)
       {
         zips[e].add(loc.substring(i+11,i+16));
         i++;
       }
     }

     // Read states
     i=0;
     while (i!=-1)
     {
       i=loc.indexOf("state",i);
       if (i!=-1)
       {
         states[e].add(loc.substring(i+9,i+12));
         i++;
       }
     }
```

```
   }
}

int nbEmployees=card(employees);
int nbChildren=sum(i in employees) i.children;
float averageSalary=sum(i in employees) i.salary/nbEmployees;
float maximalSalary=max(i in employees) i.salary;
float minimalSalary=min(i in employees) i.salary;
int numberOfSingleEmployees=card({i | i in employees:i.maritalstatus=="Single"});
{string} employeesstates=union(e in employees) states[e];

tuple salaryname
{
  float salary;
  string name;
}

reversed {salaryname} salarynames={<e.salary,e.name> | e in employees };

{employee} highpaidemployees={e | e in
employees:(ord(salarynames,<e.salary,e.name>)+1)*5<=nbEmployees};

{employee} employeeslivingincertainzipcode={e | e in employees :
card(selectedZipCode inter zips[e])!=0};

execute
{
  writeln("1. Total number of employees: ",nbEmployees);
  writeln("2. Total children: ",nbChildren);
  writeln("Average children per employee: ",nbChildren/nbEmployees);
  writeln("3. Average salary: ",averageSalary);
  writeln("Maximal salary: ",maximalSalary);
  writeln("Minimal salary: ",minimalSalary);
  writeln("4. Number of single employees: ",numberOfSingleEmployees);
  writeln("5. States where employees have residences: ",employeesstates);
  writeln("6. Number of employees inside 20% of highest paid employees:
",Opl.card(highpaidemployees));
  writeln("Details of high-paid employees:");
  for(var e in highpaidemployees) writeln("Name : ",e.name," Salary : ",e.salary);
  writeln("7. Employees living in selected zip codes:");
  for(var e in employeeslivingincertainzipcode)
  writeln(e.name);
}


/*

gives

1. Total number of employees: 12
2. Total children: 17
Average children per employee: 1.416666667
3. Average salary: 134583.333333333
Maximal salary: 220000
Minimal salary: 40000
4. Number of single employees: 5
5. States where employees have residences:  {"NJ'" "FL'" "CA'"}
6. Number of employees inside 20% of highest paid employees: 2
Details of high-paid employees:
```

```
Name : Robinson Salary : 220000
Name : Doe Salary : 210000
7. Employees living in selected zip codes:
Warner
Stevens
Smith
Doe

*/
```