Logic Programming

Exercise

Write the decision system for fraud detection of credit cards.

The system checks a concrete transaction where the transaction record contains at least

- the country where the credit card is used
- the residence address of the credit card holder
- the amount of the transaction

The system might have access to a database where all past transactions are stored.

A fraud is assumed if it is satisfying the following conditions:

- Credit card is used in Africa, the residence of the card holder is in Germany, and the amount is higher than 10'000 € (with the likelihood of 60%)
- Credit card is used outside of EU but not USA, the residence of card holder is in EU, and the amount deviate by 30% from the average (with the likelihood of 70%)

If you make assumptions, please document them!

1. Define the fraud detection as PROLOG rules

One possible solution (one transaction predicate):

```
residence_of_customer(meyer, germany).
average_amount_of_customer(meyer, 5000).

transaction(tr12212, meyer, marroko, 10000).


fraud(NR, 60) :- transaction(NR, Customer, Country_Used, Amount),
                 africa(Country_Used),
                 residence_of_customer(Customer, germany),
                 Amount > 10000.

fraud(NR, 70) :- transaction(NR, Customer, Country_Used, Amount),
                 not(eu(Country_Used)),
                 not(Country_Used = usa),
                 residence_of_customer(Customer, Residence),
                 eu(Residence),
                 average_amount_of_customer(Customer, Average),
                 Deviation is abs(Amount-Average)/Average,
                 Deviation > 1.3.


eu(belgium).
eu(france).
eu(italy).
eu(luxembourg).
eu(netherlands).
eu(germany).
eu(denmark).
eu(ireland).
eu(united_kingdom).
eu(greece).
eu(spain).
eu(portugal).
eu(austria).
eu(sweden).
eu(finland).

africa(marroko).
africa(westSahara).
africa(algeria).
```

Another possible solution (several facts for transaction):

```
residence_of_customer(meyer, germany).
average_amount_of_customer(meyer, 5000).

customer_in_transaction(tr12212, meyer).
used_in_country(tr12212, marroko)
amount_in_transaction(tr12212, 10000).


fraud(NR, 60) :- used_in_country(NR, Country_Used),
                 africa(Country_Used),
                 customer_in_transaction(NR, Customer),
                 residence_of_customer(Customer, Residence),
                 Residence = Germany,
                 amount_in_transaction(NR, Amount),
                 Amount > 10000.

fraud(NR, 70) :- used_in_country(NR, Country_Used),
                 not(eu(Country_Used)),
                 not(Country_Used = usa),
                 customer_in_transaction(NR, Customer),
                 residence_of_customer(Customer, Residence),
                 eu(Residence),
                 amount_in_transaction(NR, Amount),
                 average_amount_of_customer(Customer, Average),
                 Deviation is abs(Amount-Average)/Average,
                 Deviation > 1.3.


eu(belgium).
eu(france).
eu(italy).
eu(luxembourg).
eu(netherlands).
eu(germany).
eu(denmark).
eu(ireland).
eu(united_kingdom).
eu(greece).
eu(spain).
eu(portugal).
eu(austria).
eu(sweden).
eu(finland).

africa(marroko).
africa(westSahara).
africa(algeria).
```

Another possible solution (all arguments in the fraud predicate):

```
residence_of_customer(meyer, germany).
average_amount_of_customer(meyer, 5000).


fraud(Country_Used, Customer, Amount, 60) :-
                  africa(Country_Used),
                  residence_of_customer(Customer, Residence),
                  Residence = Germany,
                  Amount > 10000.

fraud(Country_Used, Customer, Amount, 70) :-
                  not(eu(Country_Used)),
                  not(Country_Used = usa),
                  residence_of_customer(Customer, Residence),
                  eu(Residence),
                  average_amount_of_customer(Customer, Average),
                  Deviation is abs(Amount-Average)/Average,
                  Deviation > 1.3.


eu(belgium).
eu(france).
eu(italy).
eu(luxembourg).
eu(netherlands).
eu(germany).
eu(denmark).
eu(ireland).
eu(united_kingdom).
eu(greece).
eu(spain).
eu(portugal).
eu(austria).
eu(sweden).
eu(finland).

africa(marroko).
africa(westSahara).
africa(algeria).
```

One possible solution (one transaction predicate; belongs):

```
residence_of_customer(meyer, germany).
average_amount_of_customer (meyer, 5000).

transaction(tr12212, meyer, marroko, 10000).


fraud(NR, 60) :- transaction(NR, Customer, Country_Used, Amount),
                 belongs_to(Country_Used, africa),
                 residence_of_customer(Customer, germany),
                 Amount > 10000.

fraud(NR, 70) :- transaction(NR, Customer, Country_Used, Amount),
                 not(belongs_to(Country_Used, eu)),
                 not(Country_Used = usa),
                 residence_of_customer(Customer, Residence),
                 eu(Residence),
                 average_amount_of_customer(Customer, Average),
                 Deviation is abs(Amount-Average)/Average,
                 Deviation > 1.3.


belongs_to(belgium, eu).
belongs_to(france, eu).
belongs_to(italy, eu).
belongs_to(luxembourg, eu).
belongs_to(netherlands, eu).
belongs_to(germany, eu).
belongs_to(denmark, eu).
belongs_to(ireland, eu).
belongs_to(united_kingdom, eu).
belongs_to(greece, eu).
belongs_to(spain, eu).
belongs_to(portugal, eu).
belongs_to(austria, eu).
belongs_to(sweden, eu).
belongs_to(finland, eu).

belongs_to(marroko, africa).
belongs_to(westSahara, africa).
belongs_to(algeria, africa).
```