## Hint1 Record type, OrderedList type, UnorderedList type
https://asterixdb.apache.org/docs/0.8.8-incubating/aql/datamodel.html#DerivedTypes

Record type, denoted by {…}
OrderedList type, denoted by […]
UnorderedList type, denoted by {{…}}

**Key:**
For all the collections in FROM clause, the type should be either OrderedList, which is denoted by brackets: "[…]", or UnorderedList, which is denoted by two opening flower braces: "{{…}}".

**Example1:**
SELECT y as country FROM world x, x.mondial.country y;

If we check the mondial.adm data file, search "country", we will see that world.mondial.country is denoted by brackets"[…]", so it is OrderedList type. We give "world.mondial.country" a name called y, actually the database system will make y to be a Record type, which we could think of a Record type as "tables" when we learn SQL, thus we actually create a "table" named y and by using y.name we could retrieve the information we want. We could also think of each object in the "world.mondial.country" OrderedList is a "record" in y "table".

**Example2:**
SELECT z as country FROM world x, x.mondial y, y.country z;

We will get an error message for this query:
"Type mismatch: function scan-collection expects its 1st input parameter to be type orderedlist or unorderedlist, but the actual input type is record [TypeMismatchException]"

If we check the mondial.adm data file, search "mondial", it is denoted by flower braces"{…}", so it is a Record type, thus we will get a Type mismatch error since all the relations in FROME clause should be either OrderedList or UnorderedList type.

## Hint2 Heterogenous Colletions
Test Queries4, 5 on http://courses.cs.washington.edu/courses/cse414/17sp/hw/hw5/hw5.html

**Key:**
We could use "CASE WHEN …END" to incorporate data with different types.

**Example1:**
SELECT z.name as province_name, u.name as city_name
FROM world x, x.mondial.country y, y.province z, z.city u;

We will get an error message:
"Type mismatch: function scan-collection expects its 1st input parameter to be type orderedlist or unorderedlist, but the actual input type is record [TypeMismatchException]"

If we check world.mondial.country.province.city, it can be both denoted by brackets […] and single flower braces {…}, so it can either be an OrderdList type or a Record type, violating the rule that all the collections in FROME clause should be either OrderedList or UnorderedList type.

To solve this problem, we could use "CASE WHEN…END" to generate an OrderedList of the data we want, and then name it as a new record u:

```
SELECT z.name as province_name, u.name as city_name
FROM world x, x.mondial.country y, y.province z,
    (CASE WHEN is_array(z.city) THEN z.city
             ELSE [z.city] END) u;
```

We could see that there are four records or "tables" x, y, z, u. Since they are in FROM clause, we need to make sure that world, world.mondial.country, world.mondial.country.province and the collections before u should all be OrderedList or UnorderedList type.

If we check world.mondial.country.province.city(=z.city), it could be either denoted by brackets […] or flower braces {…}, making it either be an OrderedList type and a Record type. So We split into different cases based on the type of z.city. If z.city is denoted by […](is_array(z.city) will return true), meaning that it is an OrderedList type, then just return z.city. Else if z.city is denoted by {…}, meaning that it is a Record type. Then we add an outer bracket […] to that z.city so that we could change a Record type to an OrderedList type, then we can guarantee that "CASE WHEN …END" will return an OrderedList type.

## Hint3 Nesting

Test Queries6 on http://courses.cs.washington.edu/courses/cse414/17sp/hw/hw5/hw5.html

**Key:**
Using "Let x = (subquery)" or "Let x = (CASE WHEN …END)" to group all the information you want in a list.

**Example1:**

```
SELECT z.name as province_name, (SELECT u.name FROM cities u) as city_namelist
FROM world x, x.mondial.country y, y.province z
LET cities = CASE WHEN z.city is missing THEN []
               WHEN is_array(z.city) THEN z.city
               ELSE  [z.city] END;
```

If a province has multiple cities, the Test Query 5 will return results like:

```
{ "province_name": "Attiki", "city_name": "Athens" }
{ "province_name": "Attiki", "city_name": "Piraeus" }
```

However, if we want to nest our result based on some attribute, such as we want to nest city_name for each province:

```
{ "province_name": "Attiki", "city_namelist": [ { "name": "Athens" }, { "name": "Piraeus" } ]}
```

We could use “Let x = (subquery)” or “Let x = (CASE WHEN …END)” to nest data, x will be an UnOrderedList type. In our case, cities will be an UnOrderedList of city name, this list is generate by the “CASE WHEN … END” clause.

We could see that there are only three records x, y, z in the main query of Test Query 6, compared to Test Query 5, there are four records x, y, z, u.  “cities” is UnOrderedList type, if want to use “cities.name” to retrieve city name information, we will get an error message:
”Type mismatch: function field-access-by-name expects its 1st input parameter to be type record, but the actual input type is unorderedlist”.

Instead, we could just use “cities” or use a subquery“SELECT u.name FROM cities u” to retrieve information we want. As a notice, when we use “cities u”,  the system will make u as a Record type so that we could use u.name to retrieve city name information.

## Hint4  Others

**Key:**
Add back quotes to some illegal attribute, such as `-car_code`.
Use multi-value join if there are multiple values for an attribute.
Make use of Functions such as is_array(), coll_count() etc.