Database Applications Exam (March 2015) – Dossier System

Your exam consists of several parts, explained below. You may work independently on each exam part. Submit your solutions as a single **ZIP file holding the full source code**, without any libraries and compiled binaries.

Part I – Query Existing Database, Import and Export Data

You are given a **MS SQL Server database "PhotographySystem"** holding photographs, albums, users and equipment available as a **SQL script**. Your task is to write a few data-driven applications in C# for importing data, querying data and exporting data from the database.

Problem 1. Entity Framework Mappings (Database First)

Create an **Entity Framework (EF) data model** of the existing database (map the database tables to C# classes). Use the "**database first**" model in EF. To test your EF data model, **list each photograph's title and link**.

Output

`Fields of Gold -- http://photo-forum.net/i/1919442`
`Best Photo Ever -- http://photo-forum.net/i/1920515`
`Dog -- http://photo-forum.net/i/1920281`
`Tilma Lek -- http://photo-forum.net/i/1920390`
`They are coming -- http://photo-forum.net/i/1920370`
`Sitting on a tree... -- http://photo-forum.net/i/1920004`
`Time of poets -- http://photo-forum.net/i/1920110`
`Angel eyes -- http://photo-forum.net/i/1919347`
`I dont want you to go! -- http://photo-forum.net/i/1919931`

3 score

Problem 2. Export Albums as JSON

Write a program that fetches all **albums** that have **at least 1 photo**. Order them by their photo count, then by id. Select each album's id, name, owner and total count of photos. Export the data in JSON format as shown below:

albums.json

```
[
  {
    "id": 1,
    "name": "Mobile",
    "owner": "Alexandra Svilarova",
    "photoCount": 3
  },
  {
    "id": 2,
    "name": "SoftUni Teambuilding",
    "owner": "Alexandra Svilarova",
    "photoCount": 3
  },
```

```
  {
    "id": 3,
    "name": "The little things in my life",
    "owner": "Alexandra Svilarova",
    "photoCount": 3
  },
  {
    "id": 4,
    "name": "OpenFest 2014",
    "owner": "Svetlin Nakov",
    "photoCount": 5
  }
]
```

Write the output in a JSON file named **albums.json**. The code indentation in the JSON file is not important.

4 score

Order the **albums** by **photo count**, then by **id**.

2 score

For better performance, ensure your program executes a **single DB query** and retrieves from the database only the required data (without unneeded rows and columns).

4 score

Problem 3. Users and Albums as XML

Write a program that exports data for **all users** with at least **1 album** as XML. The users should be ordered alphabetically by **full name**.

users-with-albums.xml

```
<?xml version="1.0" encoding="utf-8"?>
<users>
  <user id="3" birth-date="1982-02-09T00:00:00">
    <albums>
      <album name="Mobile" description="Mobile uploads">
        <photographs>
          <photograph title="Best Photo Ever" />
          <photograph title="Tilma Lek" />
          <photograph title="Sitting on a tree..." />
        </photographs>
      </album>
      <album name="SoftUni Teambuilding" description="Special contributions">
        <photographs>
          <photograph title="They are coming" />
          <photograph title="Sitting on a tree..." />
          <photograph title="Time of poets" />
        </photographs>
      </album>
      <album name="The little things in my life" description="Unforgettable
moments...">
        <photographs>
          <photograph title="Time of poets" />
          <photograph title="Angel eyes" />
          <photograph title="I dont want you to go!" />
        </photographs>
      </album>
    </albums>
    <camera model="XQ1" lens="AF-S Nikkor 600mm f/4G ED VR" megapixels="14" />
  </user>
  <user id="2" birth-date="1988-08-18T00:00:00">
    <albums>
      <album name="OpenFest 2014" description="OpenFest - Sofia, 1-Nov-2014
University education in the IT sector - problems and solutions">
        <photographs>
          <photograph title="Fields of Gold" />
          <photograph title="Best Photo Ever" />
          <photograph title="Tilma Lek" />
          <photograph title="They are coming" />
          <photograph title="Time of poets" />
        </photographs>
      </album>
      <album name="Digital Kids Conference - 1 Nov 2014">
        <photographs />
      </album>
```

```
      <album name="VarnaConf 2013" description="Programming and technology
conferences">
        <photographs />
      </album>
    </albums>
    <camera model="XQ2" lens="AF Nikkor 24-85mm f/2.8-4D IF" megapixels="12" />
  </user>
  <user id="1" birth-date="1983-04-17T00:00:00">
    <albums>
      <album name="New Year 2015">
        <photographs />
      </album>
      <album name="SoftUni Conf May 2014">
        <photographs />
      </album>
      <album name="Software University Team Building GrandFinale">
        <photographs />
      </album>
    </albums>
    <camera model="EOS 750D" lens="EF 17-40mm f/4.0L USM" megapixels="24" />
  </user>
</users>
```

Each user should have an **id**, **birthdate**, **albums** and **camera** (**model**, **lens** and **megapixels**). Each **album** should have a **name** and **description** (may be missing, do not attach attribute if missing). Each **photo** should have a **title**. Use an XML parser by choice.

6 score

For better performance, ensure your program executes a **single DB query** and retrieves from the database only the required data (without unneeded rows and columns).

4 score

Problem 4. Import Manufacturers from XML

Write a **C# application** based on your EF data model for **importing manufacturers and their cameras and lenses**.

The input comes from an XML file **`manufacturers-and-goods.xml`** in the following format:

manufacturers-and-goods.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<manufacturers>
      <manufacturer name="Jazzy">
            <cameras>
                  <camera model="Praktica" year="2008" megapixels="25"/>
                  <camera model="AgfaPhoto" year="2004" price="652"/>
                  <camera model="Pentax" year="2011" price="464" megapixels="39"/>
                  <camera model="Intova" year="2004" price="1626" megapixels="34"/>
                  <camera model="Kodak" year="2015" price="327"/>
```

```
                <camera model="Olympus " year="2015" price="1347"
megapixels="24"/>
                <camera model="Nikon" year="2011" price="982" megapixels="27"/>
            </cameras>
            <lenses>
                <lens model="Axis" type="Telephoto zoom lens"/>
                <lens model="Cambo" type="Telephoto zoom lens" price="361"/>
                <lens model="D-Link" type="Zoom lens" price="182"/>
            </lenses>
      </manufacturer>
      <manufacturer name="Meembee">
            <cameras>
                <camera model="Hunten" year="2011" price="982" megapixels="27"/>
                <camera model="iPUX" year="2014" price="1555" megapixels="27"/>
                <camera model="Kaiser" year="2002" price="1790"/>
            </cameras>
            <lenses>
                <lens model="Leupold II" type="Zoom lens" price="510"/>
                <lens model="Mercury A+" type="Zoom lens"/>
            </lenses>
      </manufacturer>
      ...
</manufacturers>
```

The input XML holds a sequence of requests given in the **<manufacturer>…</ manufacturer >** elements.

The only **manufacturer** attribute is **name**.

The **camera elements** hold **model**, **year**, **price** (optional) and **megapixels** (optional).

The **lens elements** hold **model**, **type** and **price** (optional).

Running the program for the first time should yield the following results:

```
Successfully added manufacturer Jazzy.
Successfully added manufacturer Meembee.
Successfully added manufacturer Demivee.
```

Running the program again should produce the following results:

```
Manufacturer Jazzy already exists.
Manufacturer Meembee already exists.
Manufacturer Demivee already exists.
```

Manufacturers with those **names** have already been imported and the program should not allow duplicate imports.

Your program should correctly **parse the input XML**.

5 score

Your program should correctly **import manufacturers**, **cameras** and **lenses.**

7 score

Your program should correctly **add manufacturers and their lenses and cameras**. If a lens or camera **fails to add**, **nothing related to the current manufacturer** should be imported.

12 score

Part II – EF Code First: Define Data Models, Import and Export Data

You are assigned to define a **code first data model in EF** and write a few data-driven applications in C# for importing data, querying data and exporting data from the database.

Use a new database "**DossierSystem**". Do not modify the "**Photography**" database.

Problem 5. EF Code First: Dossier System

Create an **Entity Framework (EF) code first data model** holding **individual** and their **locations** and **activities**. Users can have **multiple activities** and **multiple locations**. They can also be related to **multiple other individuals**.

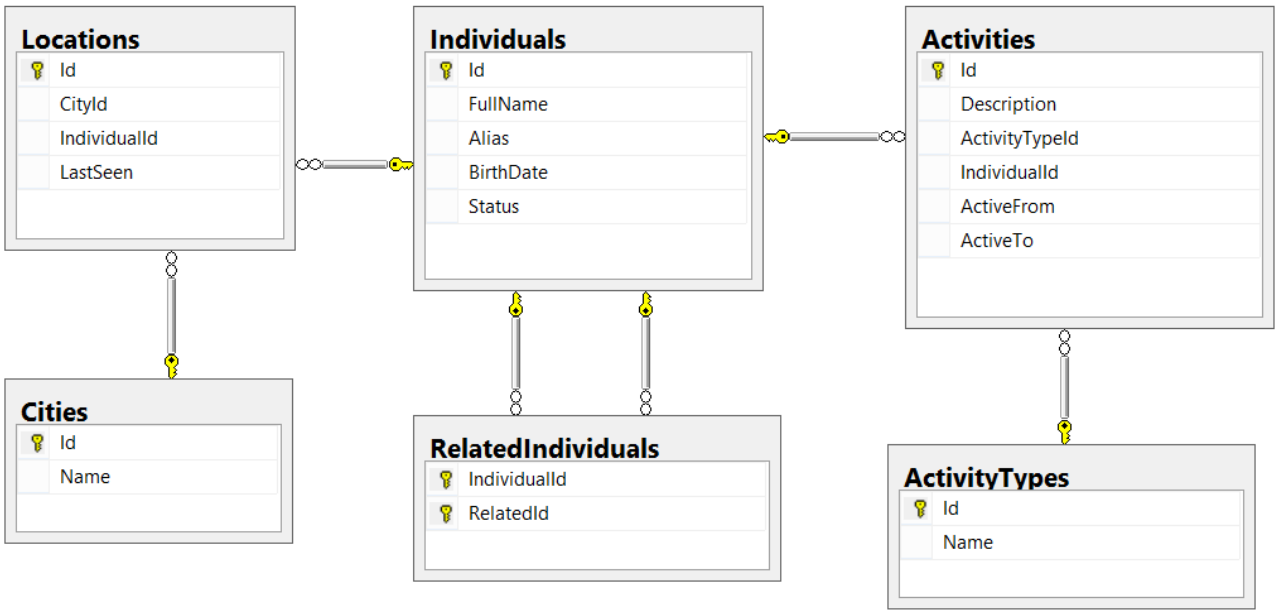
- **Individuals** have **id (unique string)**, **full name**, **alias** (optional), **birth-date** (may be missing) and a **status** (one of the following: **Active**, **Missing**, **Deceased**). Individuals may also have related individuals.
- **Cities** have **id** and **name**
- **Locations** have **id, city** and the **date** of when the individual was last seen
- **Activities** have id, activity type, individual id, **description** (optional) and dates **ActiveFrom** (required) and **ActiveTo** (optional)

Configure the following relations:

- **Individuals** can have **many locations.**
- **Individuals** can have **many activities**.
- **Individuals** can be related to **many other individuals.**

Put **navigation properties** on **both sides** of the relations.

Sample database diagram:



20 score

Seed the database by importing the provided JSON files (**cities.xml, activity-types.json, individuals.json**). Missing data should be considered **opitonal**.

10 score

Problem 6. Query the Database

Query the newly created Code First database using Entity Framework. Export the results as JSON in the format specified.

1. Active Individuals

Export all **individuals** who have an **active** status. Order them by **full name**. Select each individual's **id**, **full name** and alias.

active-individuals.json

```
[
  {
    "id": "#46b",
    "fullName": "Christopher Freeman",
    "alias": "King Pin"
  },
  {
    "id": "#59d",
    "fullName": "Frank Carter",
    "alias": "Fist"
  },
  {
    "id": "#964",
    "fullName": "George Johnston",
    "alias": "The Wolf"
  },
  {
    "id": "#412",
    "fullName": "Kenneth Cole",
    "alias": null
  },
  {
    "id": "#fbd",
    "fullName": "Rachel Diaz",
    "alias": null
  }
]
```

Correctly export **active individuals** to JSON.

4 score

For better performance, ensure your program executes a **single DB query** and retrieves from the database only the required data (without unneeded rows and columns).

4 score

2. Drug Activities

Export **all individuals** who have been involved in **drug trafficking activities.** Order them by their **full name**. Select each individual's name, birth date and **drug trafficking activities** (**description** and **dates**). The activities should be ordered by their **start date**.

drug-activities.json


```
[
  {
    "fullName": "Christopher Freeman",
    "birthDate": null,
    "activities": [
      {
        "description": "Ran a small methamphetamine laboratory
on the outskirts of Bogota.",
        "activeFrom": "1991-10-06T00:00:00",
        "activeTo": "2011-05-12T00:00:00"
      }
    ]
  },
  {
    "fullName": "George Graham",
    "birthDate": "1976-06-10T00:00:00",
    "activities": [
      {
        "description": null,
        "activeFrom": "1999-07-03T00:00:00",
        "activeTo": "2010-03-01T00:00:00"
      }
    ]
  },
  {
    "fullName": "George Johnston",
    "birthDate": "1955-11-27T00:00:00",
    "activities": [
      {
        "description": "Biggest drug import cartel in South
America.",
        "activeFrom": "1994-10-05T00:00:00",
        "activeTo": "2001-08-09T00:00:00"
      },
      {
        "description": "Columnbian drug cartel funded by
religious extremists.",
        "activeFrom": "1996-05-28T00:00:00",
        "activeTo": "2001-10-04T00:00:00"
      },
      {
        "description": "Started shipping to Europe and Africa.",
        "activeFrom": "1998-07-16T00:00:00",
        "activeTo": "2005-09-10T00:00:00"
```

```
      },
      {
        "description": "Founded a huge cocaine production
factory in the middle of the Amazon. Worldwide export.",
        "activeFrom": "1999-12-31T00:00:00",
        "activeTo": "2003-07-06T00:00:00"
      }
    ]
  },
  ...
]
```

Correctly export **all drug activities** by **user** to JSON with the required data.

5 score

For better performance, ensure your program executes a **single DB query** and retrieves from the database only the required data (without unneeded rows and columns).

4 score

3. Jurish Visitors

Export all **individuals** who have visited **Jurish** before the year **2005**. Select their **name** and **locations** (ordered by the **last time they were seen** there).

jurish-visitors.json

```
[
  {
    "fullName": "Rachel Diaz",
    "locations": [
      {
        "seen": "2002-03-28T00:00:00"
      },
      {
        "seen": "2002-10-22T00:00:00"
      },
      {
        "seen": "2012-08-02T00:00:00"
      }
    ]
  }
]
```

Correctly export **all Jurish** visitors.

5 score

For better performance, ensure your program executes a **single DB query** and retrieves from the database only the required data (without unneeded rows and columns).

4 score

4. Updating Missing People

Write a query that updates all **active** individuals' status to **Missing** if they haven't been seen in any location since 2010 (inclusive).

Correctly update **all active individuals'** status.

3 score

For better performance, ensure your program executes a **single DB query** and retrieves from the database only the required data (without unneeded rows and columns).

4 score

Exam Information

You are allowed to use any resources at your disposal, e.g. Internet, software, existing code.

You are not allowed to get help from other people. Skype, ICQ, FB, email, talks, phone calls, etc. are forbidden.

Exam time: **6 hours**.