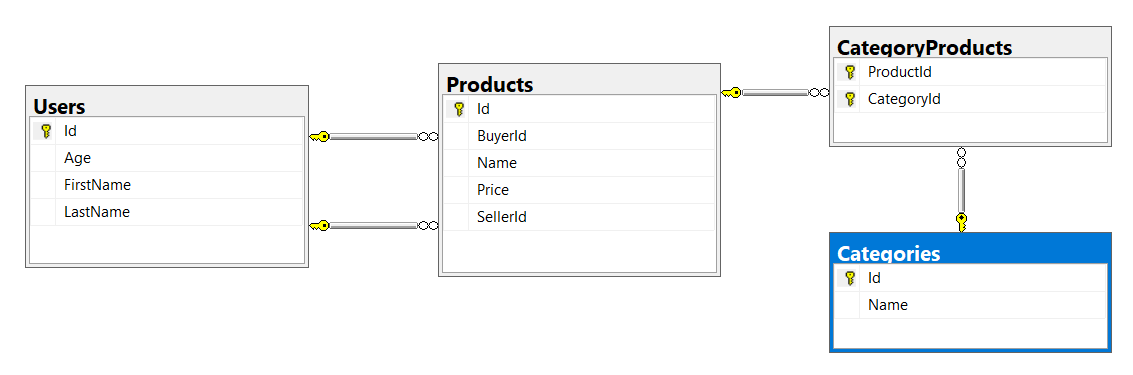
# Exercises: External Format Processing

### Products Shop

A products shop holds **users**, **products** and **categories** for the products. Users can **sell** and **buy** products.

* Users have an **id**, **first** **name** (optional) and **last** **name**, and **age** (optional).
* Products have an **id**, **name**, **price**, **buyerId** (optional) and **sellerId** as IDs of users.
* Categories have an **id** and **name** (from **3** to **15** characters)

Using Entity Framework Code First create a database following the above description.



Configure the following relations in your EF models:

* **Users** should have **many products sold** and **many products bought**.
* **Products** should have **many categories**
* **Categories** should have **many products**

## JSON Processing

### Import data

**Import** the data from the provided files (**users.json**, **products.json**, **categories.json**).

Import the **users** first. When importing products, randomly **select the buyer** and **seller** from the existing users. Leave out some **products** that have **not been sold** (i.e. buyer is null).

Randomly **generate categories** for each product from the existing categories.

### Query and Export Data

Write the below described queries and **export** the returned data to the specified **format**.

Note that because of the random generation of the data output probably will be different.

#### Query 1 - Products In Range

Get all products in a specified **price range:** 500 to 1000 (inclusive). Order them by price (from lowest to highest). Select only the **product name**, **price** and the **full name** **of the seller**. Export the result to JSON.

|  |
| --- |
| **products-in-range.json** |
| [  {  "name": "TRAMADOL HYDROCHLORIDE",  "price": 516.48,  "seller": "Christine Gomez"  },  {  "name": "Allopurinol",  "price": 518.50,  "seller": "Kathy Gilbert"  },  {  "name": "Parsley",  "price": 519.06,  "seller": "Jacqueline Perez"  },  ...  ] |

#### Query 2 - Successfully Sold Products

Get all users who have **at least 1 sold item** with a **buyer**. Order them by **last name**, then by **first name**. Select the person's **first** and **last name**. For each of the **sold products** (products with buyers), select the product's **name**, **price** and the buyer's **first** and **last name**.

|  |
| --- |
| **users-sold-products.json** |
| [  {  "firstName": "Carl",  "lastName": "Daniels",  "soldProducts": [  {  "name": "Peter Island Continous sunscreen kids",  "price": 471.30,  "buyerFirstName": "Anna",  "buyerLastName": "Parker"  },  {  "name": "Warfarin Sodium",  "price": 1379.79,  "buyerFirstName": "Brandon",  "buyerLastName": "Fuller"  }  ]  },  ...  ] |

#### Query 3 - Categories By Products Count

Get **all** **categories**. Order them by the category’s **name**. For each category select its **name**, the **number of products**, the **average price of those products** and the **total revenue** (total price sum) of those products (regardless if they have a buyer or not).

|  |
| --- |
| **categories-by-products.json** |
| [  {  "category": "Sports",  "productsCount": 49,  "averagePrice": 754.327755,  "totalRevenue": 36962.06  },  {  "category": "Adult",  "productsCount": 46,  "averagePrice": 905.283478,  "totalRevenue": 41643.04  },  ...  ] |

#### Query 4 - Users and Products

Get all users who have **at least 1 sold product**. Order them by the **number of sold products** (from highest to lowest), then by **last name** (ascending). Select only their **first** and **last name**, **age** and for each product - **name** and **price**.

Export the results to **JSON**. Follow the format below to better understand how to structure your data.

|  |
| --- |
| **users-and-products.json** |
| {  "usersCount":35,  "users":  [  {  "firstName":"Carl",  "lastName":"Daniels",  "age":59,  "soldProducts":  {  "count":10,  "products":  [  {  "name":"Finasteride",  "price":1374.01  },  {  "name":"Peter Island Continous sunscreen kids",  "price":471.30  },  {  "name":"Warfarin Sodium",  "price":1379.79  },  {  "name":"Gilotrif",  "price":1454.77  },  {  "name":"Cold and Cough",  "price":218.14  },  ...  ]  }  },  {  "firstName": null,  "lastName": "Harris",  "age": 0,  "soldProducts":  {  "count":9,  "products":  [  {  "name":"Clarins Paris Skin Illusion – 114 cappuccino",  "price":811.42  },  ...  ]  }  },  ...  ]  } |

## XML Processing

### Import Data

**Import** the data from the provided files (**users.xml**, **products.xml**, **categories.xml**).

Import the **users** first. When importing products, randomly **select the buyer** and **seller** from the existing users. Leave out some **products** that have **not been sold** (i.e. buyer is null).

Randomly **generate categories** for each product from the existing categories.

### Query and Export Data

Write the below described queries and **export** the returned data to the specified **format**. Make sure that Entity Framework generates only a **single query** for each task.

#### Query 1 - Products In Range

Get all products in a specified **price range** between 1000 and 2000 (inclusive) which have a **buyer**. Order them by price (from lowest to highest). Select only the **product name**, **price** and the **full name** **of the buyer**. Export the result to XML.

|  |
| --- |
| **products-in-range.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <products>  <product name="TYLENOL COLD MULTI-SYMPTOM DAYTIME" price="1010.81" buyer="Sandra Riley" />  <product name="Butalbital, Aspirin and Caffeine" price="1010.98" buyer=" Bennett" />  <product name="SEPHORA Acne-Fighting Mattifying Moisturizer" price="1019.28" buyer="Patricia Cooper" />  </products> |

#### Query 2 - Sold Products

Get all users who have **at least 1 sold item**. Order them by **last name**, then by **first name**. Select the person's **first** and **last name**. For each of the **sold products**, select the product's **name** and **price**.

|  |
| --- |
| **users-sold-products.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <users>  <user first-name="Carl" last-name="Daniels">  <sold-products>  <product>  <name>Peter Island Continous sunscreen kids</name>  <price>471.30</price>  </product>  <product>  <name>Warfarin Sodium</name>  <price>1379.79</price>  </product>  ...  </sold-products>  </user>  ...  </users> |

#### Query 3 - Categories By Products Count

Get **all** **categories**. Order them by the **number of products** in that category (a product can be in many categories). For each category select its **name**, the **number of products**, the **average price of those products** and the **total revenue** (total price sum) of those products (regardless if they have a buyer or not).

|  |
| --- |
| **categories-by-products.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <categories>  <category name="Sports">  <products-count>49</products-count>  <average-price>754.327755</average-price>  <total-revenue>36962.06</total-revenue>  </category>  <category name="Adult">  <products-count>46</products-count>  <average-price>905.283478</average-price>  <total-revenue>41643.04</total-revenue>  </category>  ...  </categories> |

#### Query 4 - Users and Products

Get all users who have **at least 1 sold product**. Order them by the **number of sold products** (from highest to lowest), then by **last name** (ascending). Select only their **first** and **last name**, **age** and for each product - **name** and **price**.

Export the results to **XML**. Follow the format below to better understand how to structure your data.

|  |
| --- |
| **users-and-products.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <users count="35">  <user first-name="Carl" last-name="Daniels" age="59">  <sold-products count="10">  <product name="Finasteride" price="1374.01" />  <product name="Peter Island Continous sunscreen kids" price="471.30" />  <product name="Warfarin Sodium" price="1379.79" />  <product name="Gilotrif" price="1454.77" />  <product name="Cold and Cough" price="218.14" />  ...  </sold-products>  </user>  <user last-name="Harris">  <sold-products count="9">  <product name="Clarins Paris Skin Illusion - 114 cappuccino" price="811.42" />  ...  </sold-products>  </user>  ...  </users> |