# Exercises: External Format Processing

This document defines the **exercise assignments** for the ["Databases Advanced – EF Core" course @ Software University.](https://softuni.bg/trainings/3221/entity-framework-core-february-2021)

# Products Shop Database

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** (at least 3 characters) and **age** (optional).
* Products have an **id**, **name** (at least 3 characters), **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.



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

### Import Data

### Import Users

**NOTE**: You will need method **public static string ImportUsers(ProductShopContext context, string inputJson)** and public StartUp class.

Import the users from the provided file **users.json**.

Your method should return string with message $"Successfully imported {Users.Count}";

### Import Products

**NOTE**: You will need method **public static string ImportProducts(ProductShopContext context, string inputJson)** and public StartUp class.

Import the users from the provided file **products.json**.

Your method should return string with message $"Successfully imported {Products.Count}";

### Import Categories

**NOTE**: You will need method **public static string ImportCategories(ProductShopContext context, string inputJson)** and public StartUp class.

Import the users from the provided file **categories.json**. Some of the names will be null, so you don’t have to add them in the database. Just skip the record and continue.

Your method should return string with message $"Successfully imported {Categories.Count}";

### Import Categories and Products

**NOTE**: You will need method **public static string ImportCategoryProducts(ProductShopContext context, string inputJson)** and public StartUp class.

Import the users from the provided file **categories-products.json**.

Your method should return string with message $"Successfully imported {CategoryProducts.Count}";

## 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.

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

### Export Products in Range

**NOTE**: You will need method **public static string GetProductsInRange(ProductShopContext context)** and public StartUp class.

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"  },  ...  ] |

### Export Successfully Sold Products

**NOTE**: You will need method **public static string GetSoldProducts(ProductShopContext context)** and public StartUp class.

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": "Gloria",  "lastName": "Alexander",  "soldProducts": [  {  "name": " Metoprolol Tartrate",  "price": 1405.74,  "buyerFirstName": "Bonnie",  "buyerLastName": "Fox"  }  ]  },  ...  ] |

### Export Categories by Products Count

**NOTE**: You will need method **public static string GetCategoriesByProductsCount(ProductShopContext context)** and public StartUp class.

Get **all** **categories**. Order them in descending order by the category’s **products count**. For each category select its **name**, the **number of products**, the **average price of those products** (rounded to second digit after the decimal separator) and the **total revenue** (total price sum and rounded to second digit after the decimal separator) of those products (regardless if they have a buyer or not).

|  |
| --- |
| **categories-by-products.json** |
| [  {  "category": "Garden",  "productsCount": 23,  "averagePrice": "800.15",  "totalRevenue": "18403.47",  },  {  "category": "Drugs",  "productsCount": 22,  "averagePrice": "882.20",  "totalRevenue": "19408.43"  },  ...  ] |

### Export Users and Products

**NOTE**: You will need method **public static string GetUsersWithProducts(ProductShopContext context)** and public StartUp class.

Get all users who have **at least 1 sold product with a buyer**. Order them in descending order by the **number of sold products with a buyer**. Select only their **first** and **last name**, **age** and for each product - **name** and **price**. Ignore all null values.

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

|  |
| --- |
| **users-and-products.json** |
| {  "usersCount":54,  "users":  [  {  "lastName": "Stewart",  "age": 39,  "soldProducts":  {  "count": 9,  "products":  [  {  "name": "Finasteride",  "price": 1374.01  },  {  "name": "Glyburide",  "price": 95.1  },  {  "name": "GOONG SECRET CALMING BATH ",  "price": 742.47  },  {  "name": "EMEND",  "price": 1365.51  },  {  "name": "Allergena",  "price": 109.32  },  ...  ]  }  },  ...  ]  } |

# Car Dealer

## Setup Database

A car dealer needs information about cars, their parts, parts suppliers, customers and sales.

* **Cars** have **make, model**, travelled distance in kilometers
* **Parts** have **name**, **price** and **quantity**
* Part **supplier** have **name** and info whether he **uses imported parts**
* **Customer** has **name**, **date of birth** and info whether he **is young driver** (Young driver is a driver that has **less than 2 years of experience**. Those customers get **additional 5% off** for the sale.)
* **Sale** has **car**, **customer** and **discount percentage**

A **price of a car** is formed by **total price of its parts**.



* A **car** has **many parts** and **one part** can be placed **in many cars**
* **One supplier** can supply **many parts** and each **part** can be delivered by **only one supplier**
* In **one sale**, only **one car** can be sold
* **Each sale** has **one customer** and **a customer** can buy **many cars**

## Import Data

Import data from the provided files (**suppliers.json, parts.json, cars.json, customers.json**)

### Import Suppliers

**NOTE**: You will need method **public static string ImportSuppliers(CarDealerContext context, string inputJson)** and public StartUp class.

Import the suppliers from the provided file **suppliers.json**.

Your method should return string with message **$"Successfully imported {Suppliers.Count}.";**

### Import Parts

**NOTE**: You will need method **public static string ImportParts(CarDealerContext context, string inputJson)** and public StartUp class.

Import the parts from the provided file **parts.json**. **If the supplierId doesn’t exists, skip the record**.

Your method should return string with message **$"Successfully imported {Parts.Count}.";**

### Import Cars

**NOTE**: You will need method **public static string ImportCars(CarDealerContext context, string inputJson)** and public StartUp class.

Import the cars from the provided file **cars.json**.

Your method should return string with message **$"Successfully imported {Cars.Count}.";**

### Import Customers

**NOTE**: You will need method **public static string ImportCustomers(CarDealerContext context, string inputJson)** and public StartUp class.

Import the customers from the provided file **customers.json**.

Your method should return string with message **$"Successfully imported {Customers.Count}.";**

### Import Sales

**NOTE**: You will need method **public static string ImportSales(CarDealerContext context, string inputJson)** and public StartUp class.

Import the sales from the provided file **sales.json**.

Your method should return string with message **$"Successfully imported {Sales.Count}.";**

## 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.

### Export Ordered Customers

**NOTE**: You will need method **public static string GetOrderedCustomers(CarDealerContext context)** and public StartUp class.

Get all **customers** ordered by their **birth date ascending**. If two customers are born on the same date **first print those who are not young drivers** (e.g. print experienced drivers first). **Export** the list of customers **to JSON** in the format provided below.

|  |
| --- |
| **ordered-customers.json** |
| [  {  "Name": "Louann Holzworth",  "BirthDate": " 01/10/1960",  "IsYoungDriver": false  },  {  "Name": "Donnetta Soliz",  "BirthDate": "01/10/1963",  "IsYoungDriver": true  },  ...  ] |

### Export Cars from Make Toyota

**NOTE**: You will need method **public static string GetCarsFromMakeToyota(CarDealerContext context)** and public StartUp class.

Get all **cars** from make **Toyota** and **order them by model alphabetically** and by **travelled distance descending**. **Export** the list of **cars to JSON** in the format provided below.

|  |
| --- |
| **toyota-cars.json** |
| [  {  "Id": 134,  "Make": "Toyota",  "Model": "Camry Hybrid",  "TravelledDistance": 486872832,  },  {  "Id": 139,  "Make": "Toyota",  "Model": "Camry Hybrid",  "TravelledDistance": 397831570,  },  ...  ] |

### Export Local Suppliers

**NOTE**: You will need method **public static string GetLocalSuppliers(CarDealerContext context)** and public StartUp class.

Get all suppliers that **do not import** parts from abroad. Get their id, name and the number of parts they can offer to supply. Export the list of suppliers to JSON in the format provided below.

|  |
| --- |
| **local-suppliers.json** |
| [  {  "Id": 2,  "Name": "Agway Inc.",  "PartsCount": 3  },  {  "Id": 4,  "Name": "Airgas, Inc.",  "PartsCount": 2  },  ...  ] |

### Export Cars with Their List of Parts

**NOTE**: You will need method **public static string GetCarsWithTheirListOfParts(CarDealerContext context)** and public StartUp class.

Get all **cars along with their list of parts**. For the **car** get only **make, model** and **travelled distance** and for the **parts** get only **name** and **price** (formatted to 2nd digit after the decimal point). **Export** the list of **cars and their parts to JSON** in the format provided below.

|  |
| --- |
| **cars-and-parts.json** |
| [  {  "car": {  "Make": "Opel",  "Model": "Omega",  "TravelledDistance": 176664996  },  "parts": []  },  {  "car": {  "Make": "Opel",  "Model": "Astra",  "TravelledDistance": 516628215  },  "parts": []  },  {  "car": {  "Make": "Opel",  "Model": "Astra",  "TravelledDistance": 156191509  },  "parts": []  },  {  "car": {  "Make": "Opel",  "Model": "Corsa",  "TravelledDistance": 347259126  },  "parts": [  {  "Name": "Pillar",  "Price": "100.99"  },  {  "Name": "Valance",  "Price": "1002.99"  },  {  "Name": "Front clip",  "Price": "100.00"  }  ]  },...  ] |

### Export Total Sales by Customer

**NOTE**: You will need method **public static string GetTotalSalesByCustomer(CarDealerContext context)** and public StartUp class.

**Get all customers** that have **bought at least 1 car** and get their names, bought cars count and total spent money on cars**. Order the result** list by total **spent money descending** then by **total bought cars again in descending** order. Export the list of customers to JSON in the format provided below.

|  |
| --- |
| **customers-total-sales.json** |
| [  {  "fullName": " Johnette Derryberry",  "boughtCars": 5,  "spentMoney": 13529.25  },  {  "fullName": " Zada Attwood",  "boughtCars": 6,  "spentMoney": 13474.31  },  {  "fullName": " Donnetta Soliz",  "boughtCars": 3,  "spentMoney": 8922.22  },  ...  ] |

### Export Sales with Applied Discount

**NOTE**: You will need method **public static string GetSalesWithAppliedDiscount(CarDealerContext context)** and public StartUp class.

**Get first 10** **sales** with information about the **car**, **customer** and **price** of the sale **with and without discount**. **Export** the list of sales **to JSON** in the format provided below.

|  |
| --- |
| **sales-discounts.json** |
| [  {  "car": {  "Make": "Seat",  "Model": "Mii",  "TravelledDistance": 473519569  },  "customerName": "Ann Mcenaney",  "Discount": "30.00",  "price": "2176.37",  "priceWithDiscount": "1523.46"  },  {  "car": {  "Make": "Renault",  "Model": "Alaskan",  "TravelledDistance": 303853081  },  "customerName": "Taina Achenbach",  "Discount": "10.00",  "price": "808.76",  "priceWithDiscount": "727.88"  },  ...  ] |