# Exercises: JSON Processing

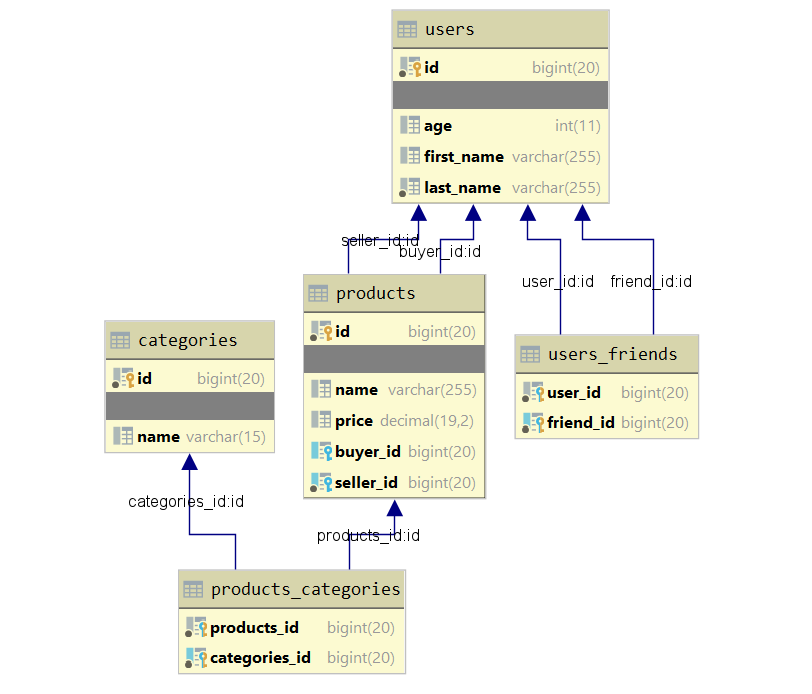
This document defines the exercise assignments for the ["Spring Data" course @ SoftUni](https://softuni.bg/trainings/4366/spring-data-february-2024).

## 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** (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 Code First approach create a database following the above description.



Configure the following relations in your models:

* **Users** should have **many products sold** and **many products bought**.
* **Products** should have **many categories**.
* **Categories** should have **many products**.
* **Users** should have **many friends** (i.e. users).

## Seed the Database

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

Import the **users** first. When importing the 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**.

### Query 1 – Products in Range

Get all products in a specified **price range** (e.g. 500 to 1000), which have **no buyer**. 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 item sold** with a **buyer**. Order them by **last name**, then by **first name**. Select the person's **first** and **last name**. For each of the **products** **sold** (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 **number of products** in each 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.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 product sold**. Order them by the **number of products sold** (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  },  ...  ]  }  },  ...  ]  } |

## Car Dealer

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

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

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

Using Code First approach create a database following the above description.



Configure the following relations in your models:

* 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.**

## Car Dealer Import Data

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

First, import the **suppliers**. When importing the **parts**, set to each part a **random supplier** from the already imported suppliers. Then, when importing the cars add **between 3 and 5 random parts** to each car. Then import **all customers**. Finally, import the **sales records** by **randomly** selecting a **car, customer** and the amount of **discount to be applied** (discounts can be 0%, 5%, 10%, 15%, 20%, 30%, 40% or 50%).

## Car Dealer Query and Export Data

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

### Query 1 – Ordered Customers

Get all **customers**, ordered by their **birthdate in ascending order**. 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** |
| [  {  "Id": 29,  "Name": "Louann Holzworth",  "BirthDate": " 1960-10-01T00:00:00",  "IsYoungDriver": false,  "Sales": [],  },  {  "Id": 28,  "Name": "Donnetta Soliz",  "BirthDate": " 1963-10-01T00:00:00",  "IsYoungDriver": true,  "Sales": [],  },  ...  ] |

### Query 2 – Cars from Make Toyota

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

|  |
| --- |
| **toyota-cars.json** |
| [  {  "Id": 117,  "Make": "Toyota",  "Model": "Camry Hybrid",  "TravelledDistance": 954775807,  },  {  "Id": 112,  "Make": "Toyota",  "Model": "Camry Hybrid",  "TravelledDistance": 92275807,  },  ...  ] |

### Query 3 – Local Suppliers

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": 6  },  {  "Id": 4,  "Name": "Airgas, Inc.",  "partsCount": 5  },  ...  ] |

### Query 4 – Cars with Their List of Parts

Get all **cars along with their list of parts**. For the **car** get only **make, model** and **traveled distance**. For the **parts** get only the **name** and the **price**. **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": 2147483647,  },  "parts": [  {  "Name": "Front Left Side Outer door handle",  "Price": 999.99  },  {  "Name": "Gudgeon pin",  "Price": 44.99  },  {  "Name": "Oil pump",  "Price": 100.19  },  {  "Name": "Transmission pan",  "Price": 106.99  }  ]  },  {  "car": {  "Make": "Opel",  "Model": "Astra",  "TravelledDistance": 9223372036854775807  },  "parts": [  {  "Name": "Overflow tank",  "Price": 1200.99  },  ...  ]  },  ...  ] |

### Query 5 – Total Sales by Customer

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

|  |
| --- |
| **customers-total-sales.json** |
| [  {  "fullName": "Hipolito Lamoreaux",  "boughtCars": 5,  "spentMoney": 8360.48  },  {  "fullName": "Francis Mckim",  "boughtCars": 4,  "spentMoney": 7115.50  },  {  "fullName": "Johnette Derryberry",  "boughtCars": 4,  "spentMoney": 5337.72  },  ...  ] |

### Query 6 – Sales with Applied Discount

Get all **sales** with information about the **car**, the **customer** and the **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": "Peugeot",  "Model": "405",  "TravelledDistance": 92036854775807  },  "customerName": "Donnetta Soliz",  "Discount": 0.3,  "price": 1402.53,  "priceWithDiscount": 981.771  },  {  "car": {  "Make": "Mercedes",  "Model": "W124",  "TravelledDistance": 2147647  },  "customerName": "Carri Knapik",  "Discount": 0.2,  "price": 254.96999999999997,  "priceWithDiscount": 203.97599999999997  },  ...  ] |