🧪 Hands-On Lab: Convert CSV + JSON Data into RDF Triples and Query with SPARQL

This lab will teach you how to:

1. Convert **CSV and JSON** data into **RDF triples**
2. Load that RDF into a **triple store** (we'll use **Apache Jena Fuseki**)
3. Query it using **SPARQL** to validate and explore your knowledge graph

---

🗃️ PART 1: Convert CSV to RDF Using OpenRefine

✅ Step 1: Install OpenRefine with RDF Plugin

1. Download OpenRefine: https://openrefine.org/download.html
2. Download RDF plugin: https://github.com/stkenny/grefine-rdf-extension
3. Install the RDF plugin:
   - Unzip the plugin
   - Copy the folder into `OpenRefine/extensions`
   - Restart OpenRefine

✅ Step 2: Prepare CSV File

Create a file named `products.csv`:

```
id,name,category,price
P001,Laptop,Electronics,1200
P002,Chair,Furniture,200
P003,Smartphone,Electronics,800
```

✅ Step 3: Load CSV in OpenRefine

1. Open OpenRefine in your browser (typically runs at `http://127.0.0.1:3333`)
2. Click **"Create Project" → Upload** `products.csv` **→ Next → Create Project**

✅ Step 4: Create RDF Mapping

1. Click **RDF** in top-right → Configure RDF Mapping
2. Set base namespace: `http://example.org/products#`

Now define mappings:

- Row node: `http://example.org/products#{{cells["id"].value}}`
- Add **rdf:type**: `ex:Product`
- Add properties:
  - `ex:name` → value from `name`
  - `ex:category` → value from `category`
  - `ex:price` → value from `price` (add datatype: `xsd:decimal`)

✅ Step 5: Export as RDF (Turtle)

1. Click **RDF → Export RDF as Turtle**
2. Save as `products.ttl`

🧾 PART 2: Convert JSON to RDF Using Python + RDFLib

✅ Step 1: Install RDFLib

```
pip install rdflib
```

✅ Step 2: Sample JSON File

Create `employees.json`:

```
[
  {
    "id": "E001",
    "name": "Alice",
    "role": "Engineer",
    "department": "R&D"
  },
  {
    "id": "E002",
    "name": "Bob",
    "role": "Manager",
    "department": "HR"
  }
]
```

✅ Step 3: Python Script to Convert to RDF

```
from rdflib import Graph, Literal, RDF, Namespace, URIRef
import json
```

```
g = Graph()
EX = Namespace("http://example.org/employees#")

g.bind("ex", EX)

with open("employees.json") as f:
    data = json.load(f)

for item in data:
    emp_uri = URIRef(EX + item["id"])
    g.add((emp_uri, RDF.type, EX.Employee))
    g.add((emp_uri, EX.name, Literal(item["name"])))
    g.add((emp_uri, EX.role, Literal(item["role"])))
    g.add((emp_uri, EX.department, Literal(item["department"])))

g.serialize("employees.ttl", format="turtle")
```

💾 PART 3: Load RDF Triples into Apache Jena Fuseki

✅ Step 1: Download and Launch Fuseki

1. Download Apache Jena Fuseki: https://jena.apache.org/download/index.cgi

2. Extract folder and run from terminal:

```
cd apache-jena-fuseki-*/fuseki
./fuseki-server
```

By default, Fuseki runs at `http://localhost:3030`

✅ Step 2: Create a Dataset

1. Visit `http://localhost:3030`

2. Click "**Manage datasets**" → "**Add new dataset**"

3. Choose name `enterpriseKG` → Persistent memory → Create

✅ Step 3: Upload RDF Files

1. Go to `http://localhost:3030/enterpriseKG/upload`

2. Upload both:

    - `products.ttl`

    - `employees.ttl`

These are now stored in your knowledge graph.

🔎 PART 4: Query with SPARQL

✅ Example 1: Get All Products

```
PREFIX ex: <http://example.org/products#>
SELECT ?product ?name ?category ?price
WHERE {
  ?product a ex:Product ;
           ex:name ?name ;
           ex:category ?category ;
           ex:price ?price .
}
```

✅ Example 2: Get Employees in R&D

```
PREFIX ex: <http://example.org/employees#>
SELECT ?employee ?name
WHERE {
  ?employee a ex:Employee ;
            ex:department "R&D" ;
            ex:name ?name .
}
```

✅ Example 3: List All Unique Roles

```
PREFIX ex: <http://example.org/employees#>
SELECT DISTINCT ?role
WHERE {
  ?s a ex:Employee ;
     ex:role ?role .
}
```

✅ Bonus Tips

- Add `rdfs:label` to improve readability
- Add `xsd:decimal`, `xsd:string`, `xsd:date` datatypes where appropriate
- Use **SHACL** or **reasoners** to validate consistency
- Add **owl:sameAs** if entities exist in other datasets (e.g., Wikidata)

🎉 You've Completed the Lab!

You now know how to:

- Convert real-world CSV and JSON data into RDF
- Load it into a triple store
- Explore and validate the graph using SPARQL