CRYPTO VIZ

BOOTSTRAP

CRYPTO VIZ

For this activity, you'll need the files available on the SharePoint site related to this module.

pandas: fast, flexible, and expressive data structures

Check **pandas** out, install it and use `pandas.read_csv` function with or without the `chunksize` feature to open and describe the huge *Liquor_Sales.csv* file...

```
> pip install pandas
```

duckdb: in-process dbms

Check **duckdb** out and install it.

```
> pip install duckdb==0.8.0
```

Get inspiration from the following chunk of code to :

- ✓ explore the data (using *pandas*) ;
- ✓ import from AND export to a *parquet* file ;

```
import duckdb as ddb
con = ddb.connect("my-working-database.db")
con.sql("""create table liquor as (select * from read_csv("Liquor_Sales.csv", types={"Item
    Number": "VARCHAR"}))""")
con.sql("select * from liquor")
con.close()
```

Play around with duckdb for more integration (pandas, postgresql, etc), or to load several files with the same schema at once: csv, parquet.

Apache Spark: unified engine for large-scale data analytics

Check the general documentation of Spark.

Then create your custom network and your Spark cluster master (use the *bitnami docker image*)

```
> docker network create -d bridge spark-experiment
> docker run -v "<working-dir-path>:/opt/bitnami/spark/work" -e SPARK_MODE=master -p
    8080:8080 --network spark-experiment bitnami/spark:latest
```

Do not use these commands without understanding them. It would make no sense.

Now you can visit Spark web UI at `http://localhost:8080` to check the state of your cluster.

You might need to set write permission on your working directory for `others`...

Copy the master URL from the logs (`spark://MASTERURL:PORT`).
Then, on separated terminals, create as much workers as you want (ex: 4 workers) with the following command:

```
> docker run -v "<working-dir-path>:/opt/bitnami/spark/work" -e SPARK_MODE=worker -e
    SPARK_MASTER_URL="spark://MASTERURL:PORT" --network spark-experiment bitnami/spark:
    latest
```

Open a dedicated console and run your application on your distributed environment (check the web ui during your experiments):

```
> docker run -itv "<working-dir-path>:/opt/bitnami/spark/work" -e SPARK_MASTER_URL="spark
    ://MASTERURL:PORT"  --network spark-experiment bitnami/spark:latest bash
```

In the container...

```
> pip install pyspark ipython && export PATH=$PATH:/.local/bin/ && ipython
```

Here is a simple code that you can distribute on a Spark cluster: Wordcount.

As it says, it counts occurrence of every words in a given text. You can use Spark map/reduce paradigm to take advantage of your platform:

```
Python 3.9.16 (main, May 19 2023, 23:53:35)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.13.2 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from operator import add
In [2]: from pyspark.sql import SparkSession
In [3]: import os
In [4]: spark = SparkSession.builder.appName("WordCount").master(os.environ.get("
    SPARK_MASTER_URL")).getOrCreate()

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel
    ).
23/05/26 03:23:39 WARN NativeCodeLoader: Unable to load native-hadoop library for your
    platform... using builtin-java classes where applicable

In [5]: lines = spark.read.text("work/discours-macron-savines-le-lac-30-mars-2023.txt").
    rdd.map(lambda r: r[0])
In [6]: counts = lines.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1)).reduceByKey(
    add)
In [7]: output = counts.collect()
In [9]: for (word, count) in output:
    ...:     print("%s: %i" % (word, count))
    ...:
Merci: 2
beaucoup,: 1
Monsieur: 8
le: 111
Ministre,: 1
et: 160
merci: 4
pour: 69
travail: 6
fait: 11
par: 31
...
```

To go further, build a mini cluster with your mates for your project.

v 0.9.1