SMARCO

SMARt COmmunities Skills Development in Europe

Urban Data Analytics

Dr. Assoc. Prof. Sotiris Kotsiantis

HELLENIC OPEN UNIVERSITY

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

SMARCO

SMARt COmmunities Skills Development in Europe

Unit 2 – Big Data Analytics Tools

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Unit 2 - Aim and objectives

- This unit introduces trainees to big data tools with a specific emphasis on Spark.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Unit 2 - Learning outcomes

- Explain data analytics tools.
- Apply PySpark and Collab to perform basic data processing techniques.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Table of contents

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Terms and keywords

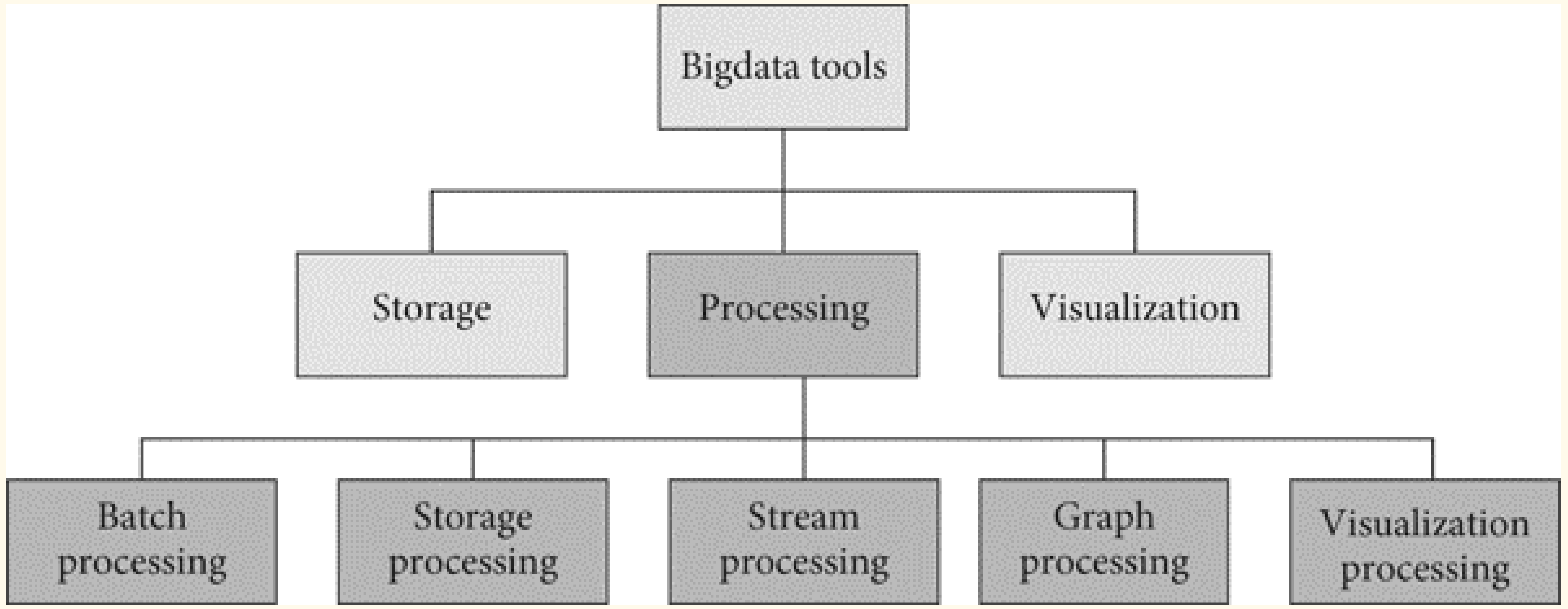
- Storage Processing- Spanner and MemSQL
- Batch Processing - Apache Hadoop and Spark
- Streaming processing- Apache Flink, Storm, Kafka and Samza
- Graph processing- Apache Giraph, GraphX, GraphLab and Neo4j
- PySpark

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Big Data Tools



Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Storage Processing-Popular tools

- Both Spanner and MemSQL are ACID-compliant relational database systems, meaning that any changes to the data are reliably committed and not lost in the event of unexpected or unwelcome scenarios. Both solutions offer large scalability of data storage - Spanner offering up to petabytes of data while MemSQL offering up to terabytes.
- The main difference between Spanner and MemSQL is in their respective capabilities for horizontal scalability - Spanner is able to dynamically scale across clouds or data centers, with no limit, while MemSQL's scaling is limited to a single cluster.
- Even though MemSQL performs better than Spanner in terms of performance, Spanner still offers a higher degree of data consistency across its distributed clusters.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Storage Processing

- The best option for a particular application depends on the specific requirements for data consistency, performance, scalability, and data model.
- If the application requires large scalability, a high degree of consistency, and generalized query performance, then Spanner is the better option.
- On the other hand, if the application requires fast, ad-hoc query performance with limited scalability, then MemSQL is the better choice.

Co-funded by the European Union

smarco

Batch Processing

- In the context of big data, batch processing refers to the process of running long-running jobs such as data mapping, aggregation, and analytics against large data sets. Common tools used for batch processing include:
  - Apache Hadoop: Apache Hadoop is an open source distributed computing platform that enables the processing of large data sets across clusters of computers using simple programming models. Hadoop is well-suited for batch processing as it can scale up to hundreds or thousands of machines, providing a robust platform for data processing.
  - Apache Spark: Apache Spark is a fast and general-purpose engine for large-scale data processing. It can be used to implement batch processing functions in a distributed and fault tolerant manner. Spark also provides powerful APIs for working with structured and semi-structured data, making data integration and analysis easier.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Streaming processing

- Apache Flink is a distributed streaming processing framework for handling large-scale data processing and streaming analytics. It allows for real-time processing of data from multiple sources such as IoT devices, sensors, and websites.
- Apache Storm is a distributed real-time computation system for streaming data. It allows for rapid processing of data from web services, sensors, and social media networks.
- Apache Kafka is an open source stream processing platform. It can be used for both batch and streaming processing of data from distributed and heterogeneous sources.
- Apache Samza is a distributed stream processing framework. It is suitable for real-time processing of data streams from applications such as web services, sensors, and databases.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Streaming processing

- Storm provides higher availability for data-processing operations compared to Flink. It also has better latency than Flink, which makes it suitable for short response-time applications like real-time analytics. On the other hand, Flink is a better choice for applications that require high throughput.
- Kafka is usually used in combination with Apache Spark or Apache Storm for stream processing jobs.
- Samza is built on top of Apache Kafka and is used to process and analyze streamed data in real-time.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Graph processing

- Apache Giraph: It is an Apache Hadoop-based system for processing extremely large-scale graph data in the form of billions of nodes and edges. It uses a master-worker architecture.
- GraphLab: This is a large-scale parallel processing system for performing computations on large graph-structured data. It provides an extensive set of APIs for graph analysis and machine learning and supports Python, Java and Scala languages.
- Apache Spark GraphX: It is a distributed in-memory graph processing engine built on top of Apache Spark. It provides a set of APIs for developers to perform graph analytics on large-scale graph datasets.
- Neo4j: It is an open-source NoSQL database with support for Java, Python, and other programming languages. It is designed to manage and query connected data with ease.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Graph processing

- Apache Giraph is written in Java and works on Apache Hadoop.
- Apache GraphX is a graph computing framework built on top of Apache Spark.
- Neo4j is a native graph database platform, which utilizes nodes, edges, and labels to design, store, and manage its data. Apache GraphX is an open source graph processing framework designed to build and execute distributed algorithms on graphs.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

PySpark

- PySpark is easy to use due to its Python API, which makes it more attractive to developers.
- PySpark is faster than traditional large-scale data processing frameworks like Hadoop MapReduce.
- PySpark allows developers to perform complex computations on large datasets by using its in-memory cluster computing capabilities.
- With the use of DataFrames, PySpark allows for structured query analysis which helps improve performance.
- PySpark also provides a multitude of libraries for analytics, machine learning, graph processing, and streaming.
- PySpark also integrates with a number of other big data tools and cloud services including Apache Hive, Apache Cassandra, and Amazon S3.
- PySpark is open source and free to use.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Pandas vs Pyspark library

- **Use Pandas When:**
  - Your dataset fits into the memory of a single machine.
  - You are working on exploratory data analysis (EDA), data cleaning, or small to medium-sized data manipulation tasks.
- **Use PySpark When:**
  - You are dealing with very large datasets that need to be distributed across a cluster.
  - You need horizontal scalability to handle big data workloads.
  - You want to leverage the full capabilities of the Apache Spark ecosystem for distributed computing, including machine learning and graph processing.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Run Jupyter Online with Colab

- Google Colab is a cloud-based notebook environment that excels in collaborative work, data analysis, and machine learning tasks.
- You can write and execute python code, save and share your analyses, and access powerful computing resources, all for free from your browser.
- To start working with Colab you first need to log in to your google/gmail account, then go to this link https://colab.research.google.com.

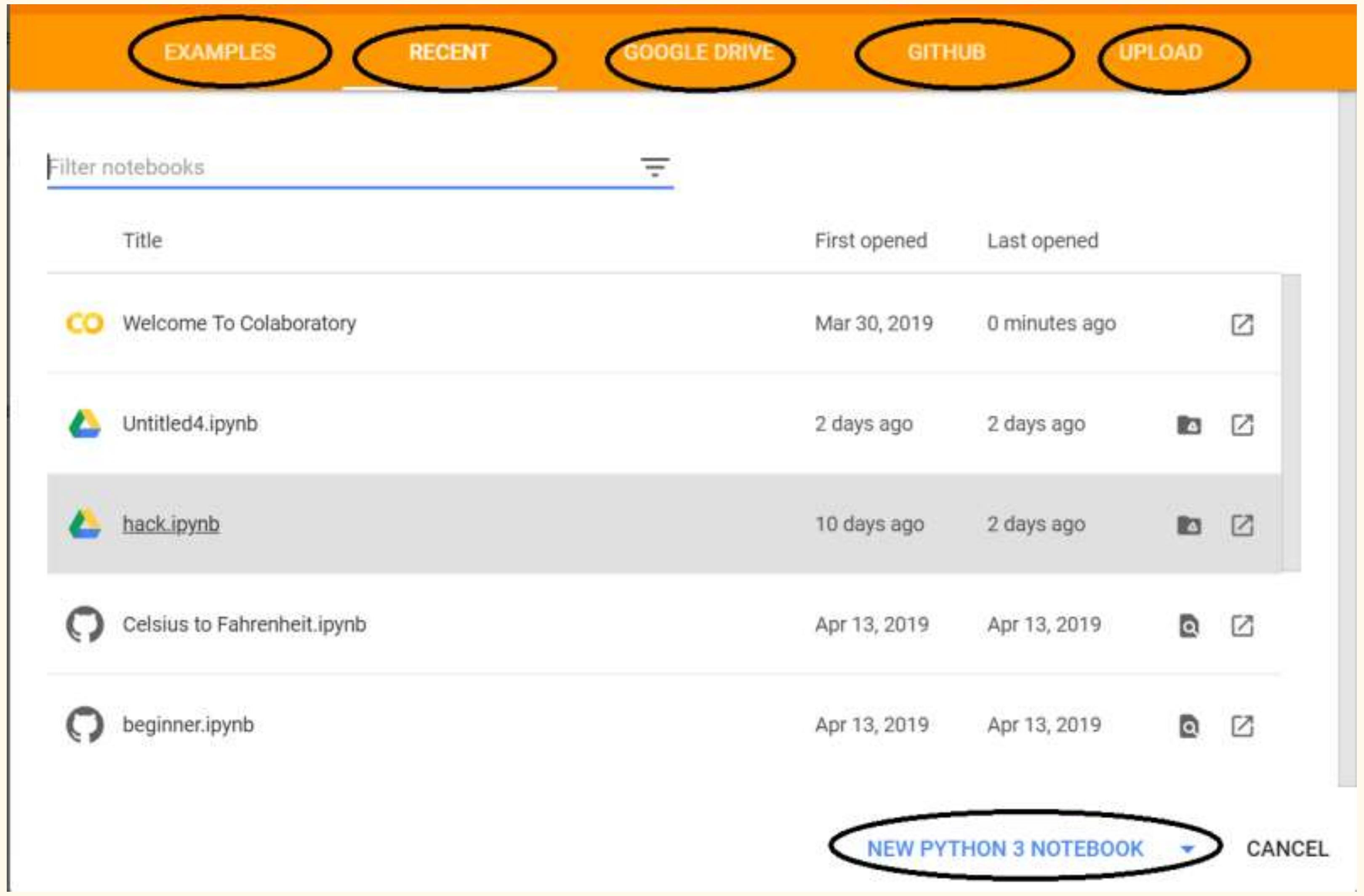
Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Opening Jupyter Notebooks in Colab

- EXAMPLES: Contain a number of Jupyter notebooks of various examples.
- RECENT: Jupyter notebook you have recently worked with.
- GOOGLE DRIVE: Jupyter notebook in your google drive.
- GITHUB: You can add Jupyter notebook from your GitHub but you first need to connect Colab with GitHub.
- UPLOAD: Upload from your local directory.



Title	First opened	Last opened
Welcome To Colaboratory	Mar 30, 2019	0 minutes ago
Untitled4.ipynb	2 days ago	2 days ago
hack.ipynb	10 days ago	2 days ago
Celsius to Fahrenheit.ipynb	Apr 13, 2019	Apr 13, 2019
beginner.ipynb	Apr 13, 2019	Apr 13, 2019

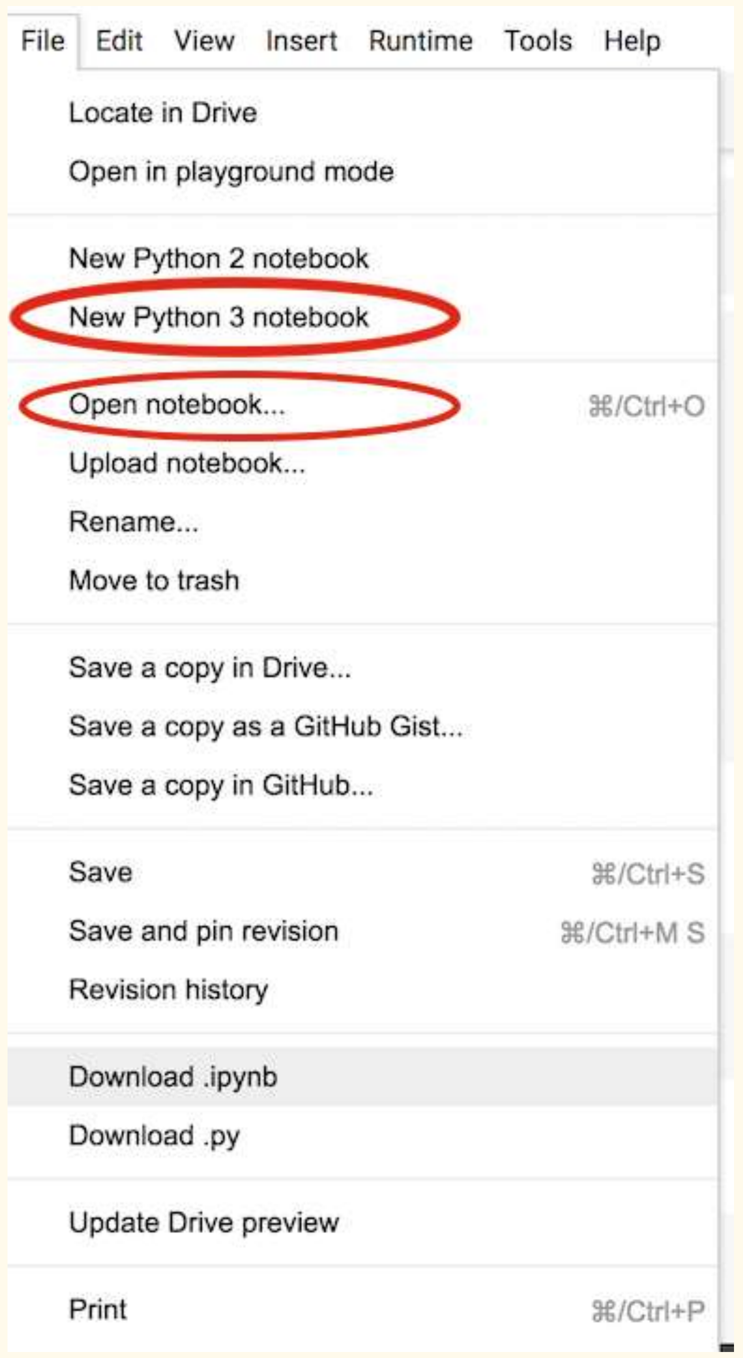
Co-funded by the European Union

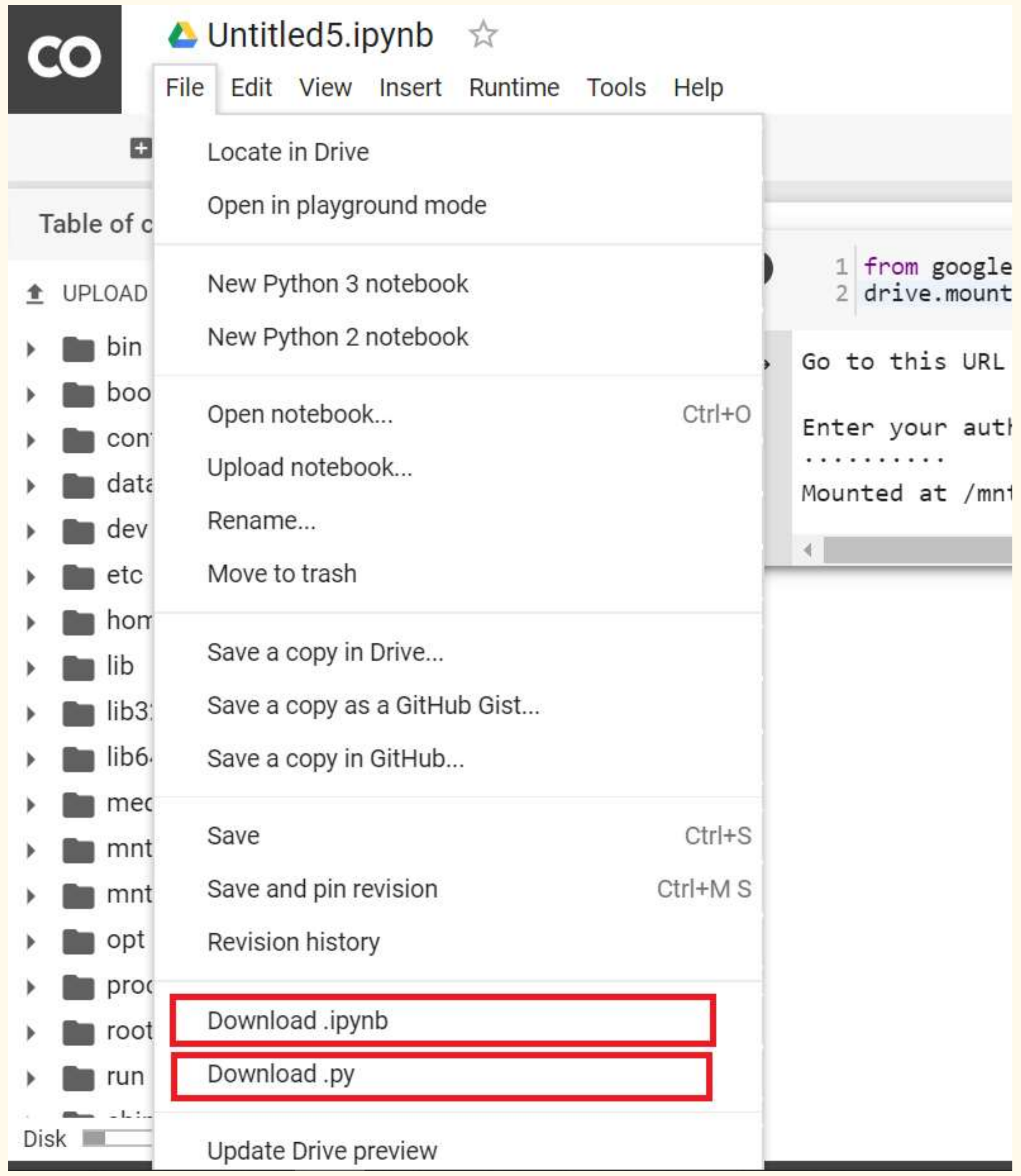
smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Create/Open/Download a Notebook

- If you want to open something specific, drop the “File” menu down to “Open Notebook…”





```
from google
drive.mount
```

```
Go to this URL

Enter your auth
..........
Mounted at /mnt
```

Co-funded by
the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

CSV, basic sharing

- A basic approach to share data is the comma separated value (CSV) format
  - it is a text format, accessible to all apps
  - each line (even if blank) is a row
  - in each row, each value is separated from the others by a comma (even if it is blank)
  - cannot capture complex things like formula

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Using PySpark in Google Colab can offer several advantages

- **Distributed Computing**: PySpark is designed for distributed computing. Google Colab provides free access to a limited amount of cloud-based resources, which allows you to leverage distributed computing capabilities for processing large datasets.
- **Scalability**: PySpark scales horizontally, meaning it can distribute tasks across multiple nodes in a cluster. This is particularly useful when dealing with big data or computationally intensive tasks, where the processing load can be distributed for faster execution.
- **Access to Cloud Resources**: Google Colab allows you to access Google Cloud resources, making it easier to integrate with other Google Cloud services and perform distributed computations using Google's infrastructure.
- **Free GPU Acceleration**: Google Colab provides free access to Graphics Processing Units (GPUs). While PySpark is not primarily GPU-accelerated, you can still leverage GPU acceleration for specific tasks by using libraries that support GPU computations (e.g., TensorFlow, PyTorch).

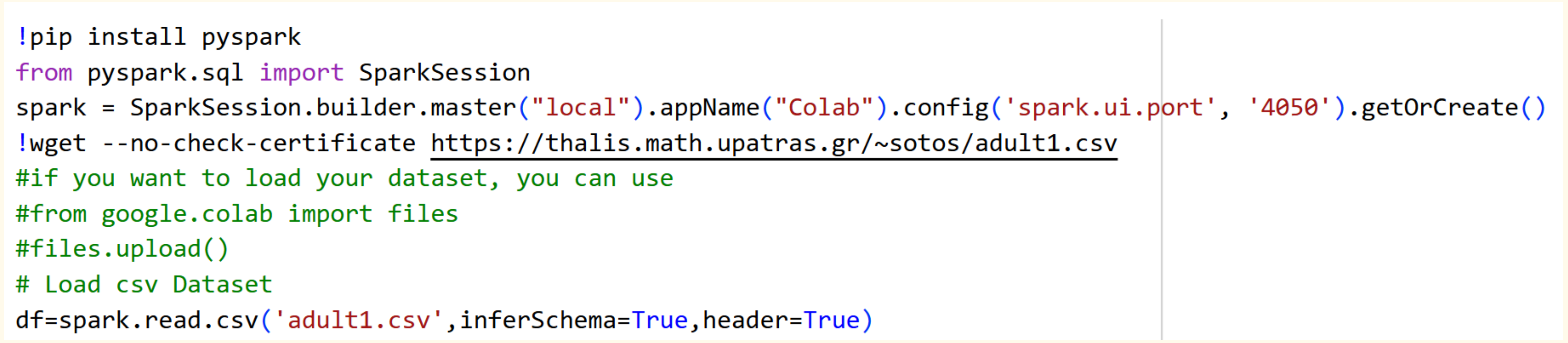
Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Data Processing using PySpark and Google Colab

- The following example shows how to use PySpark to process data sources to feed decision-making



```
!pip install pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local").appName("Colab").config('spark.ui.port', '4050').getOrCreate()
!wget --no-check-certificate https://thalis.math.upatras.gr/~sotos/adult1.csv
#if you want to load your dataset, you can use
#from google.colab import files
#files.upload()
# Load csv Dataset
df=spark.read.csv('adult1.csv',inferSchema=True,header=True)
```

Press Shift+Enter to execute the cell in colab cell

- Adult dataset is commonly used for binary classification tasks, where the goal is to predict whether a person earns more than $50,000 per year based on various demographic and socio-economic attributes.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

PySpark DataFrames

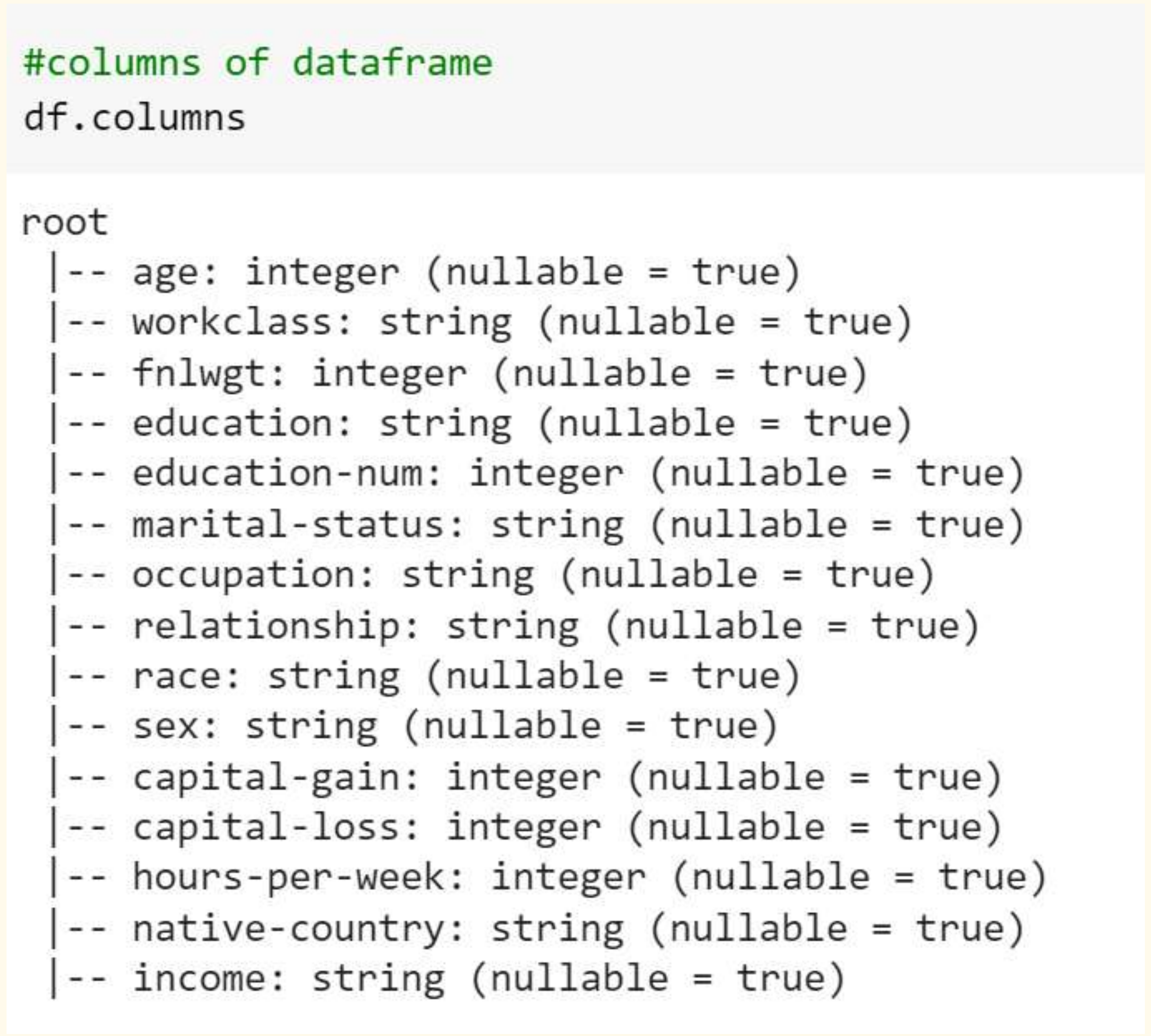
- Distributed collections of data organized into named columns, similar to a table in a relational database or a data frame in Pandas.
- Leverage the distributed computing capabilities of Apache Spark. They are split into partitions, and the operations are performed in parallel across a cluster of machines.
- Have a schema, which defines the structure of the data, including the names and types of columns. The schema allows for optimization and efficient execution of queries.
- Support a wide range of operations, including filtering, grouping, aggregation, joins, and more. These operations are similar to those in SQL or Pandas, making it easier for users to transition between different tools.

Co-funded by the European Union

smarco

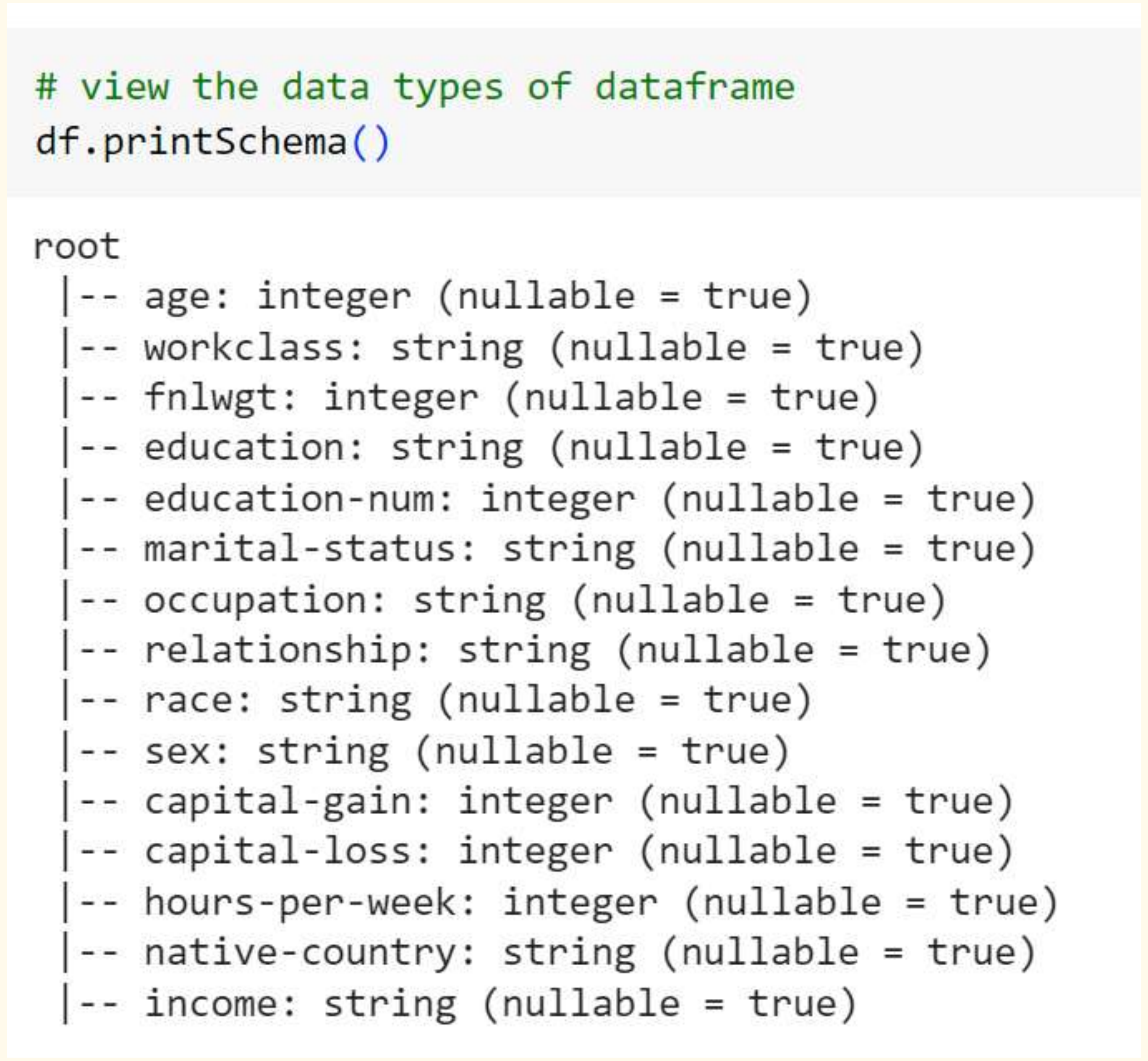
Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Columns and datatypes of the dataframe



```
#columns of dataframe
df.columns
```

```
root
 |-- age: integer (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: integer (nullable = true)
 |-- education: string (nullable = true)
 |-- education-num: integer (nullable = true)
 |-- marital-status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital-gain: integer (nullable = true)
 |-- capital-loss: integer (nullable = true)
 |-- hours-per-week: integer (nullable = true)
 |-- native-country: string (nullable = true)
 |-- income: string (nullable = true)
```



```
# view the data types of dataframe
df.printSchema()
```

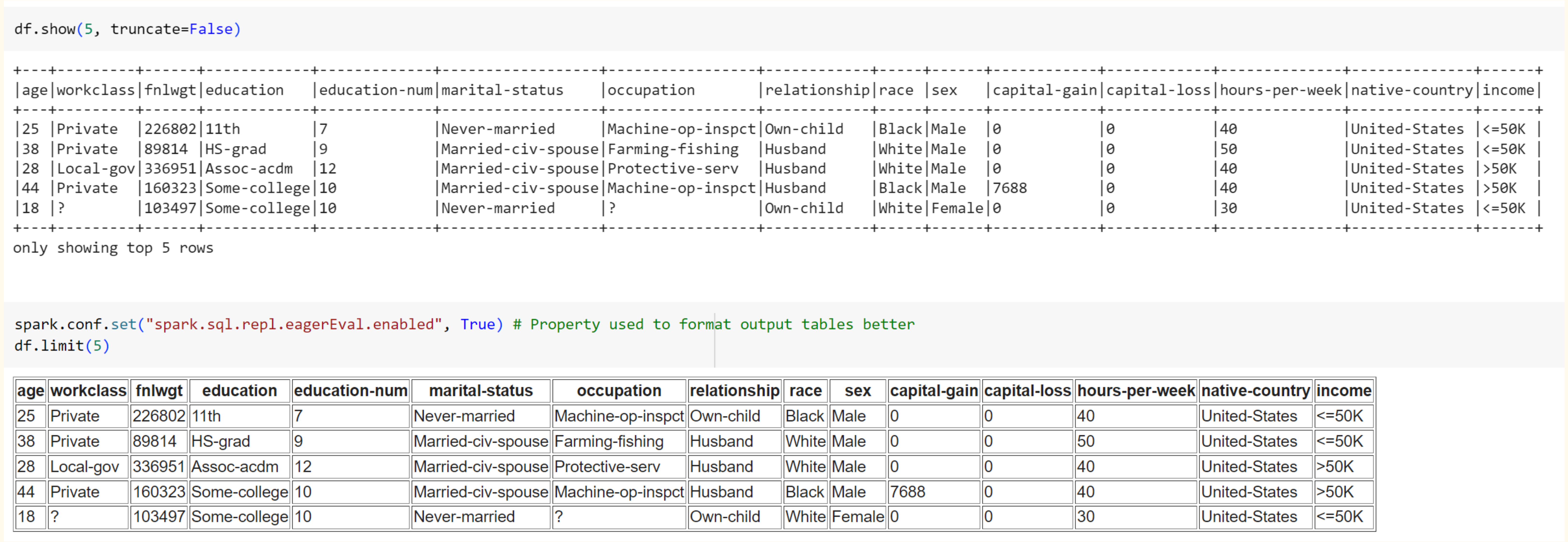
```
root
 |-- age: integer (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: integer (nullable = true)
 |-- education: string (nullable = true)
 |-- education-num: integer (nullable = true)
 |-- marital-status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital-gain: integer (nullable = true)
 |-- capital-loss: integer (nullable = true)
 |-- hours-per-week: integer (nullable = true)
 |-- native-country: string (nullable = true)
 |-- income: string (nullable = true)
```

Co-funded by
the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

View the dataframe



```
df.show(5, truncate=False)
```

```
+---+---------+------+------------+-------------+------------------+-----------------+------------+-----+------+------------+------------+--------------+--------------+------+
|age|workclass|fnlwgt|education   |education-num|marital-status    |occupation       |relationship|race |sex   |capital-gain|capital-loss|hours-per-week|native-country|income|
+---+---------+------+------------+-------------+------------------+-----------------+------------+-----+------+------------+------------+--------------+--------------+------+
|25 |Private  |226802|11th        |7            |Never-married     |Machine-op-inspct|Own-child   |Black|Male  |0           |0           |40            |United-States |<=50K |
|38 |Private  |89814 |HS-grad     |9            |Married-civ-spouse|Farming-fishing  |Husband     |White|Male  |0           |0           |50            |United-States |<=50K |
|28 |Local-gov|336951|Assoc-acdm  |12           |Married-civ-spouse|Protective-serv  |Husband     |White|Male  |0           |0           |40            |United-States |>50K  |
|44 |Private  |160323|Some-college|10           |Married-civ-spouse|Machine-op-inspct|Husband     |Black|Male  |7688        |0           |40            |United-States |>50K  |
|18 |?        |103497|Some-college|10           |Never-married     |?                |Own-child   |White|Female|0           |0           |30            |United-States |<=50K |
+---+---------+------+------------+-------------+------------------+-----------------+------------+-----+------+------------+------------+--------------+--------------+------+
only showing top 5 rows
```

```
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Property used to format output tables better
df.limit(5)
```

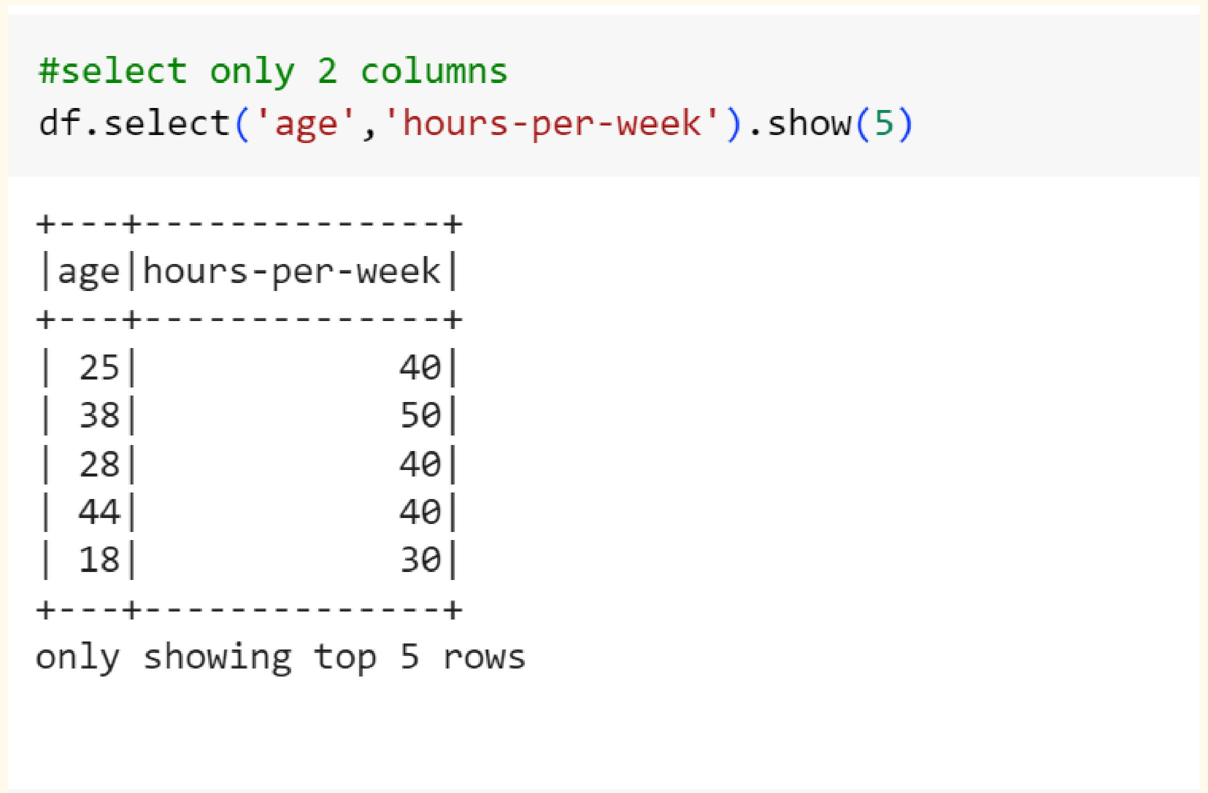
age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K

Co-funded by
the European Union

smarco

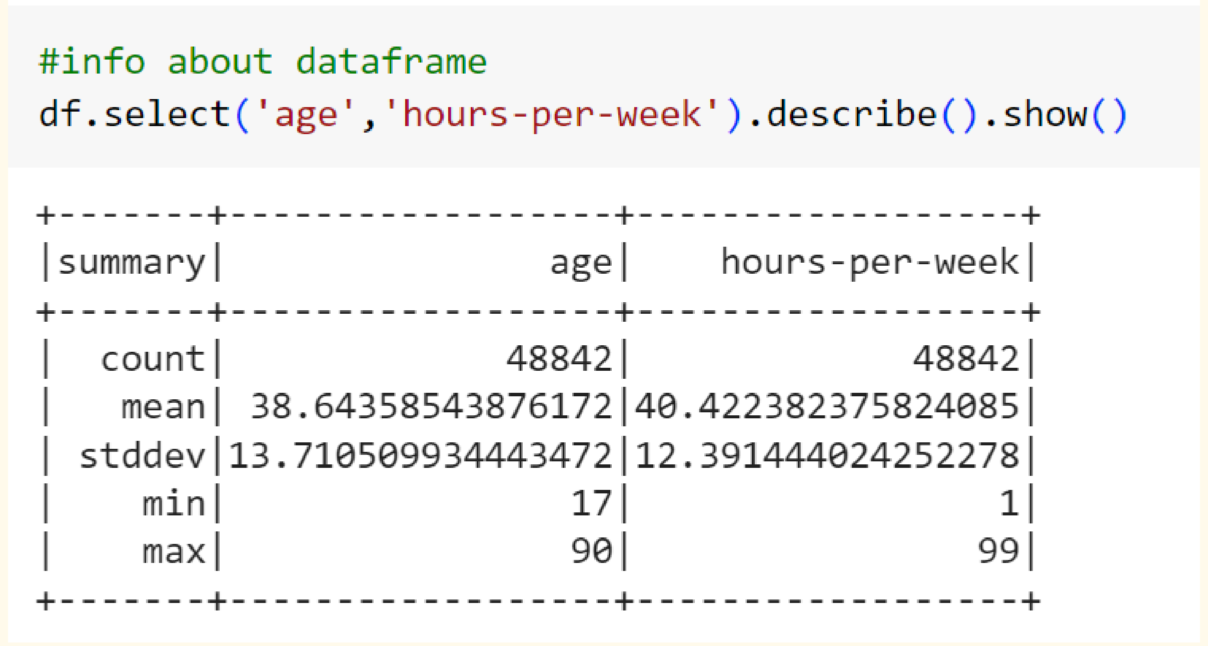
Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Select columns and group



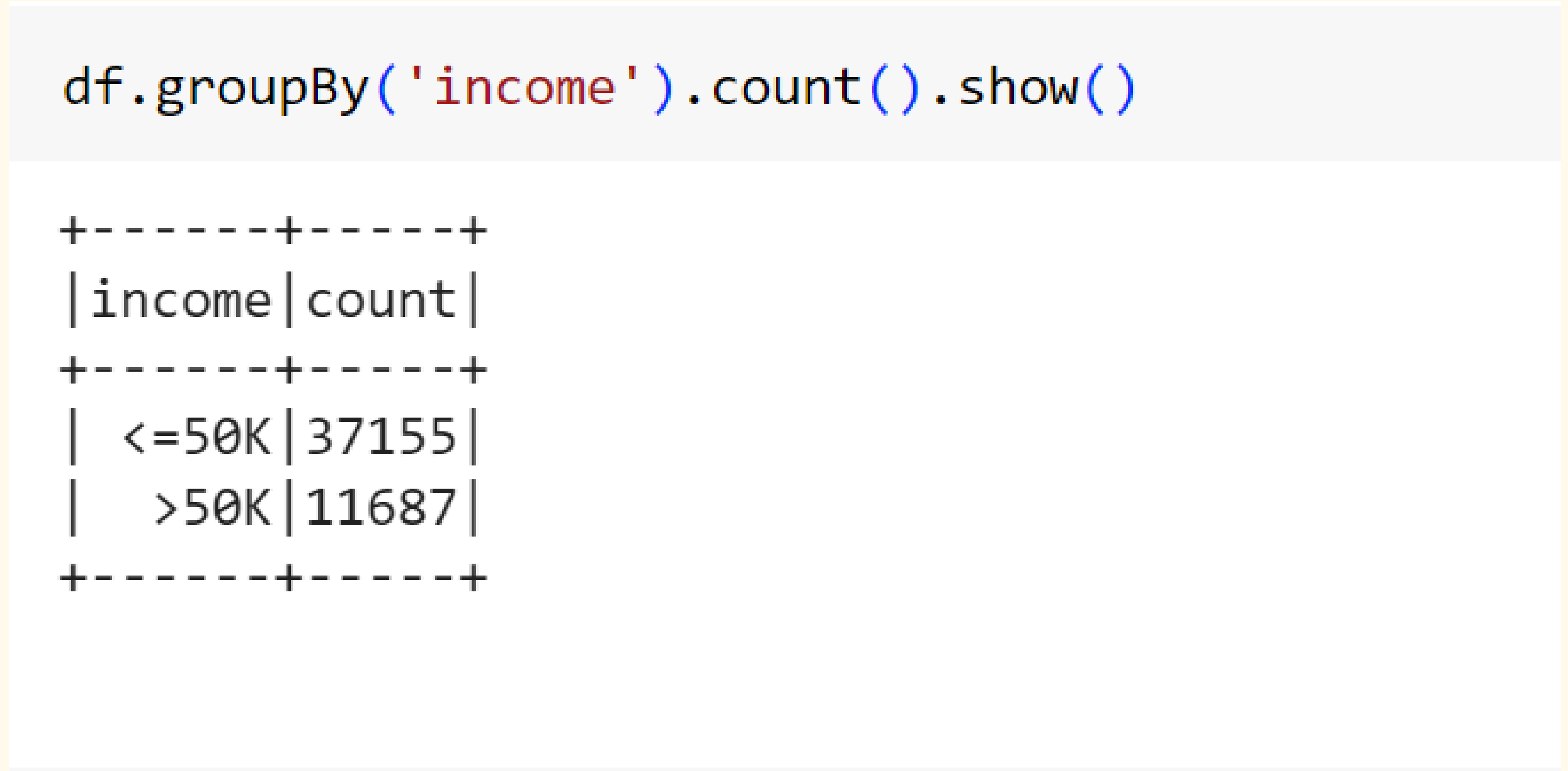
```
#select only 2 columns
df.select('age','hours-per-week').show(5)
```

```
+---+--------------+
|age|hours-per-week|
+---+--------------+
| 25|            40|
| 38|            50|
| 28|            40|
| 44|            40|
| 18|            30|
+---+--------------+
only showing top 5 rows
```



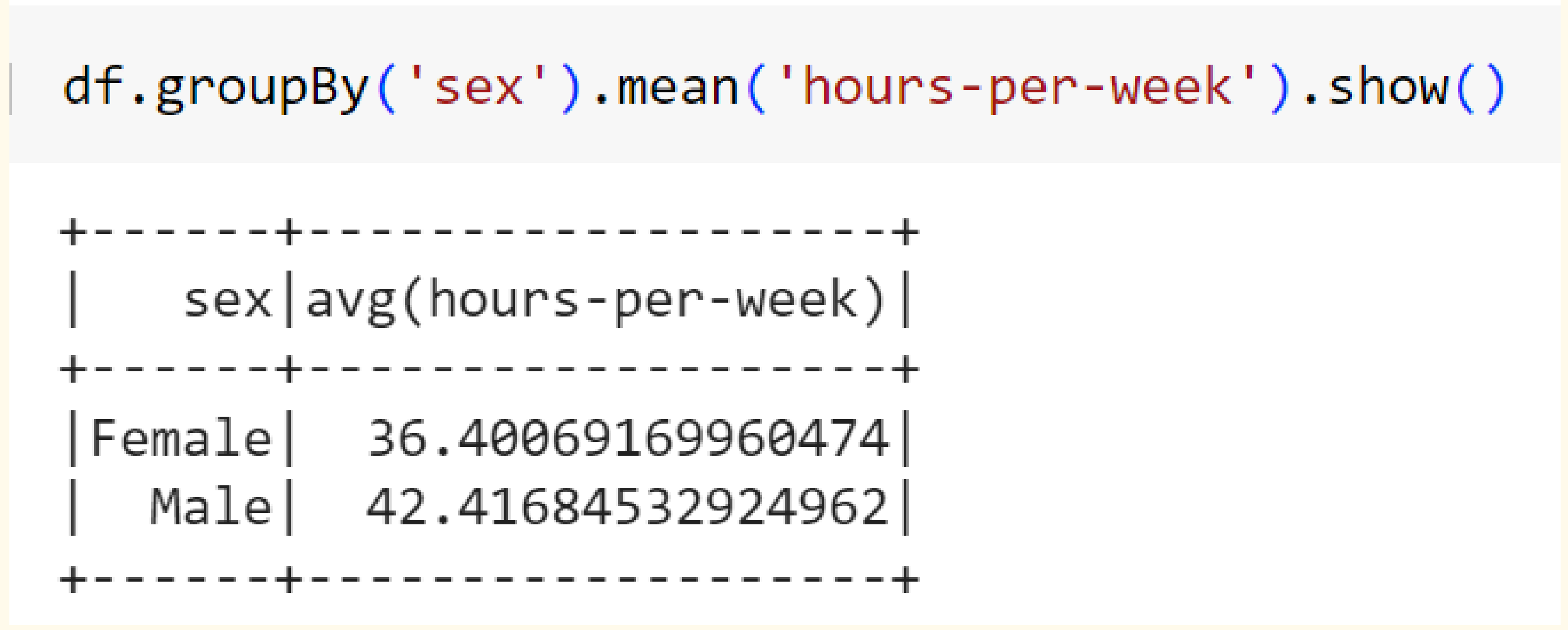
```
#info about dataframe
df.select('age','hours-per-week').describe().show()
```

```
+-------+-----------------+-----------------+
|summary|              age|   hours-per-week|
+-------+-----------------+-----------------+
|  count|            48842|            48842|
|   mean| 38.64358543876172|40.422382375824085|
| stddev|13.710509934443472|12.391444024252278|
|    min|               17|                1|
|    max|               90|               99|
+-------+-----------------+-----------------+
```



```
df.groupBy('income').count().show()
```

```
+------+-----+
|income|count|
+------+-----+
| <=50K|37155|
|  >50K|11687|
+------+-----+
```



```
df.groupBy('sex').mean('hours-per-week').show()
```

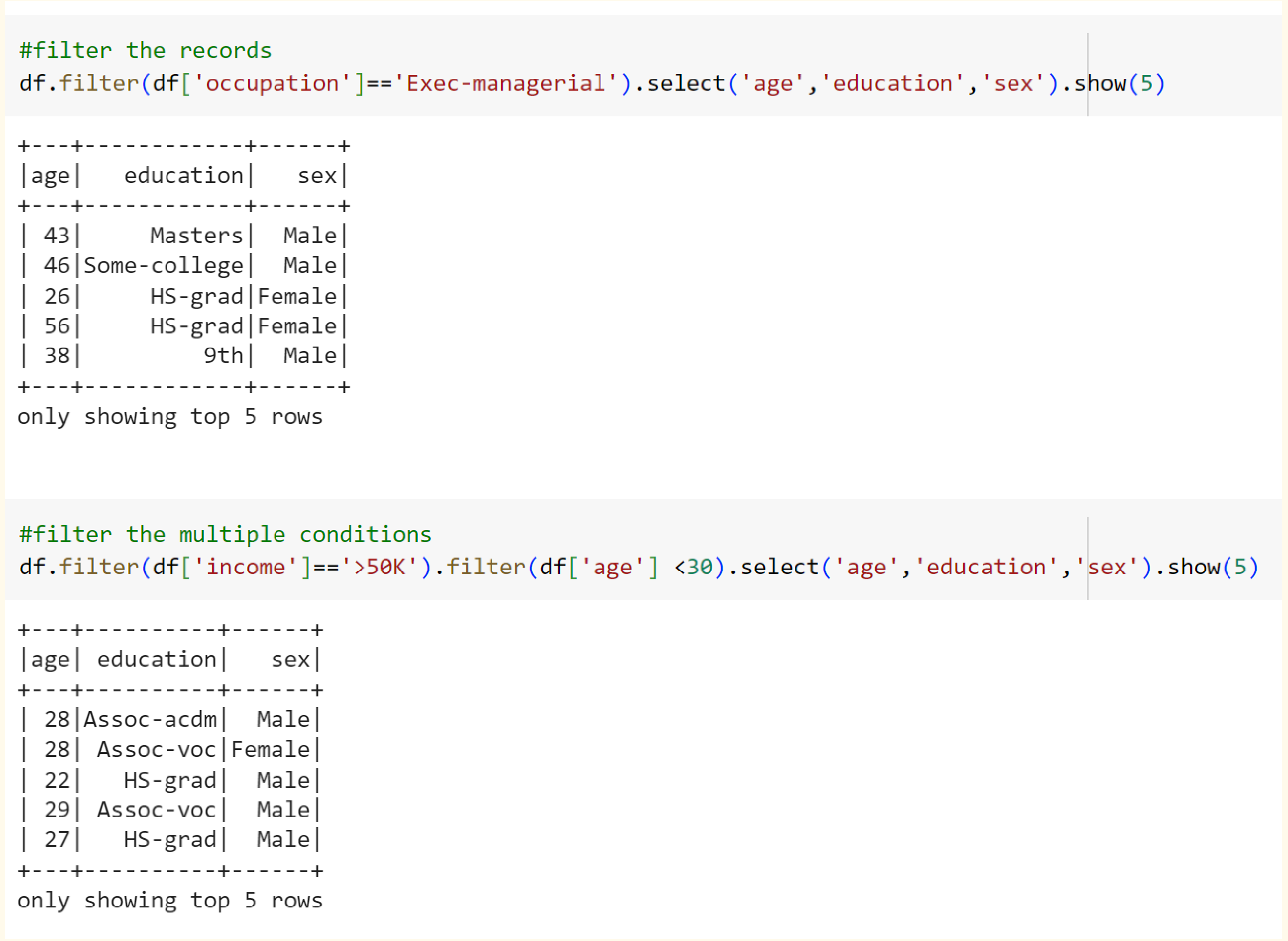
```
+------+-------------------+
|   sex|avg(hours-per-week)|
+------+-------------------+
|Female|  36.40069169960474|
|  Male|  42.41684532924962|
+------+-------------------+
```

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Filtering data and drop duplicates

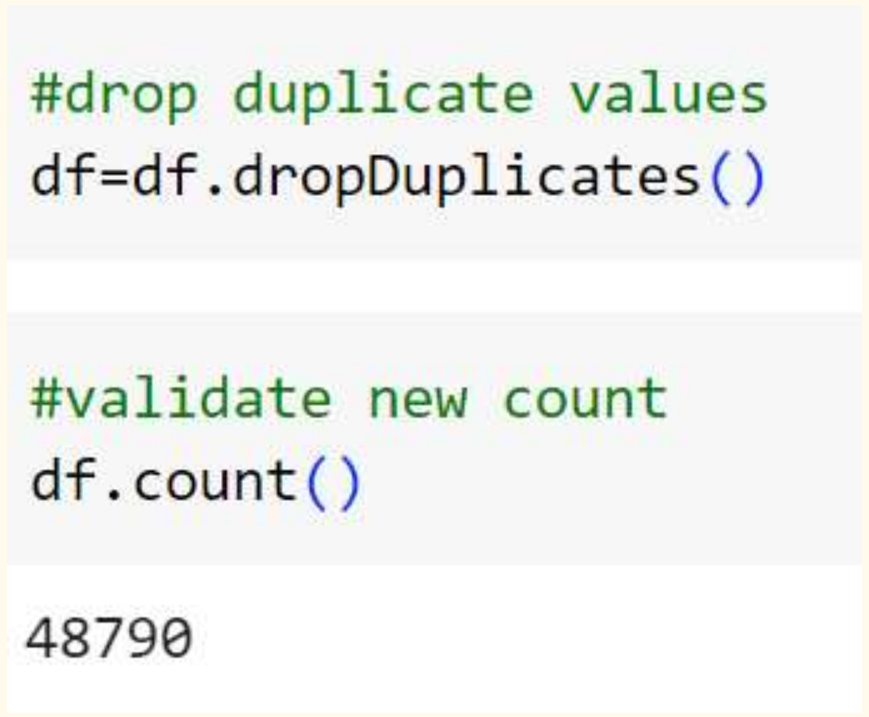


```
#filter the records
df.filter(df['occupation']=='Exec-managerial').select('age','education','sex').show(5)
```

```
+---+------------+------+
|age|   education|   sex|
+---+------------+------+
| 43|     Masters|  Male|
| 46|Some-college|  Male|
| 26|     HS-grad|Female|
| 56|     HS-grad|Female|
| 38|         9th|  Male|
+---+------------+------+
only showing top 5 rows
```

```
#filter the multiple conditions
df.filter(df['income']=='>50K').filter(df['age'] <30).select('age','education','sex').show(5)
```

```
+---+----------+------+
|age| education|   sex|
+---+----------+------+
| 28|Assoc-acdm|  Male|
| 28| Assoc-voc|Female|
| 22|   HS-grad|  Male|
| 29| Assoc-voc|  Male|
| 27|   HS-grad|  Male|
+---+----------+------+
only showing top 5 rows
```



```
#drop duplicate values
df=df.dropDuplicates()
```

```
#validate new count
df.count()
```

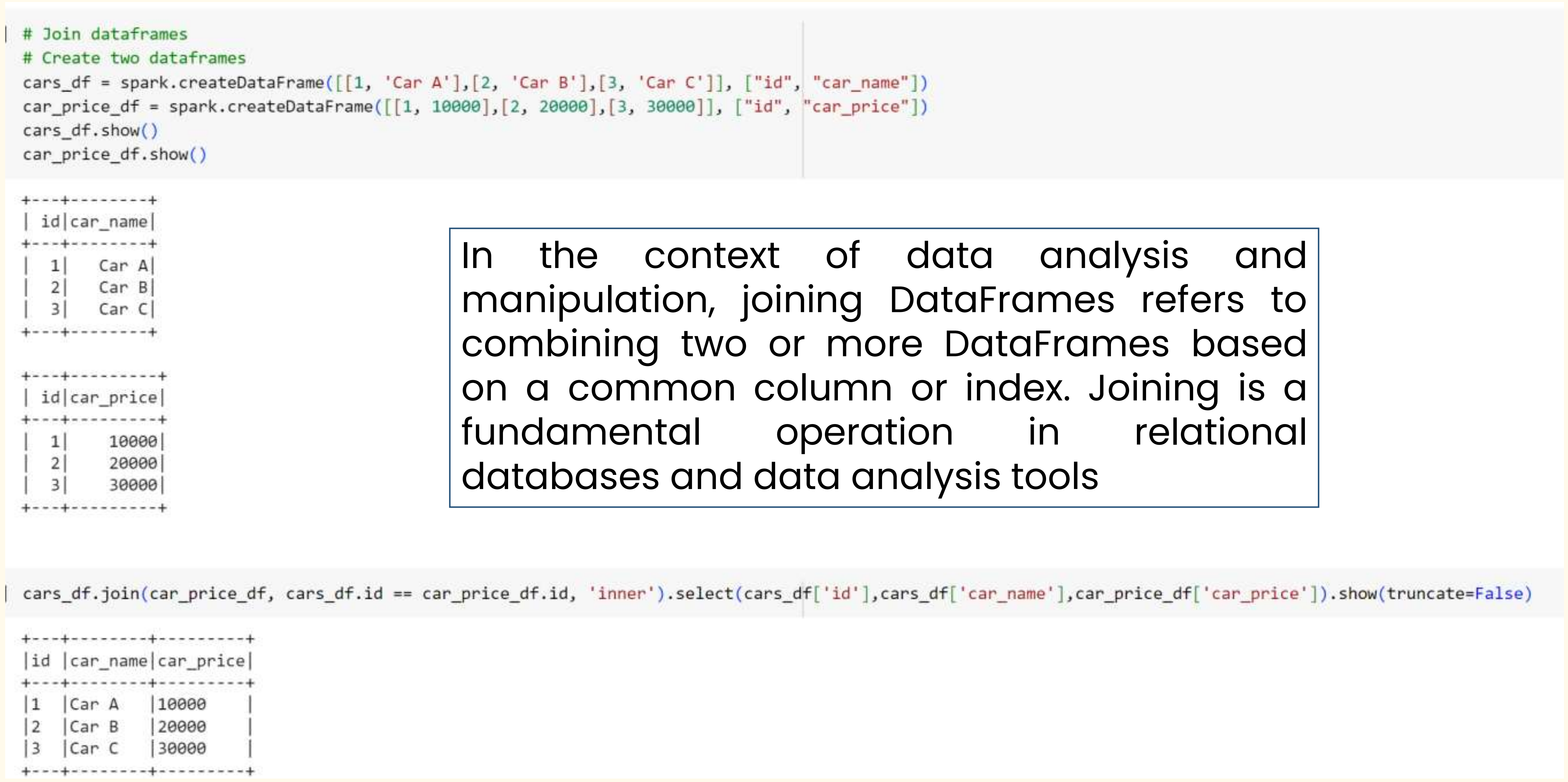
```
48790
```

Co-funded by
the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Join Dataframes



```
# Join dataframes
# Create two dataframes
cars_df = spark.createDataFrame([[1, 'Car A'],[2, 'Car B'],[3, 'Car C']], ["id", "car_name"])
car_price_df = spark.createDataFrame([[1, 10000],[2, 20000],[3, 30000]], ["id", "car_price"])
cars_df.show()
car_price_df.show()
```

```
+---+--------+
| id|car_name|
+---+--------+
|  1|   Car A|
|  2|   Car B|
|  3|   Car C|
+---+--------+

+---+---------+
| id|car_price|
+---+---------+
|  1|    10000|
|  2|    20000|
|  3|    30000|
+---+---------+
```

```
cars_df.join(car_price_df, cars_df.id == car_price_df.id, 'inner').select(cars_df['id'],cars_df['car_name'],car_price_df['car_price']).show(truncate=False)
```

```
+---+--------+---------+
|id |car_name|car_price|
+---+--------+---------+
|1  |Car A   |10000    |
|2  |Car B   |20000    |
|3  |Car C   |30000    |
+---+--------+---------+
```

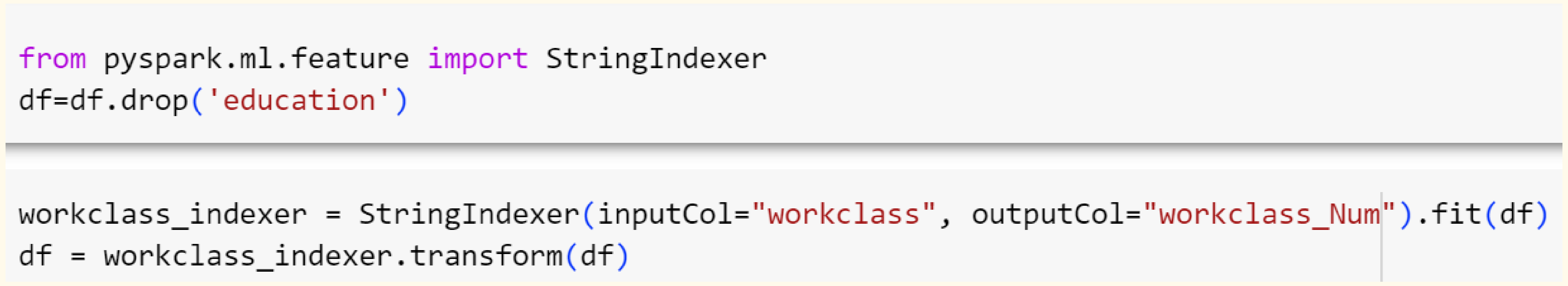
In the context of data analysis and manipulation, joining DataFrames refers to combining two or more DataFrames based on a common column or index. Joining is a fundamental operation in relational databases and data analysis tools

Co-funded by the European Union

smarco

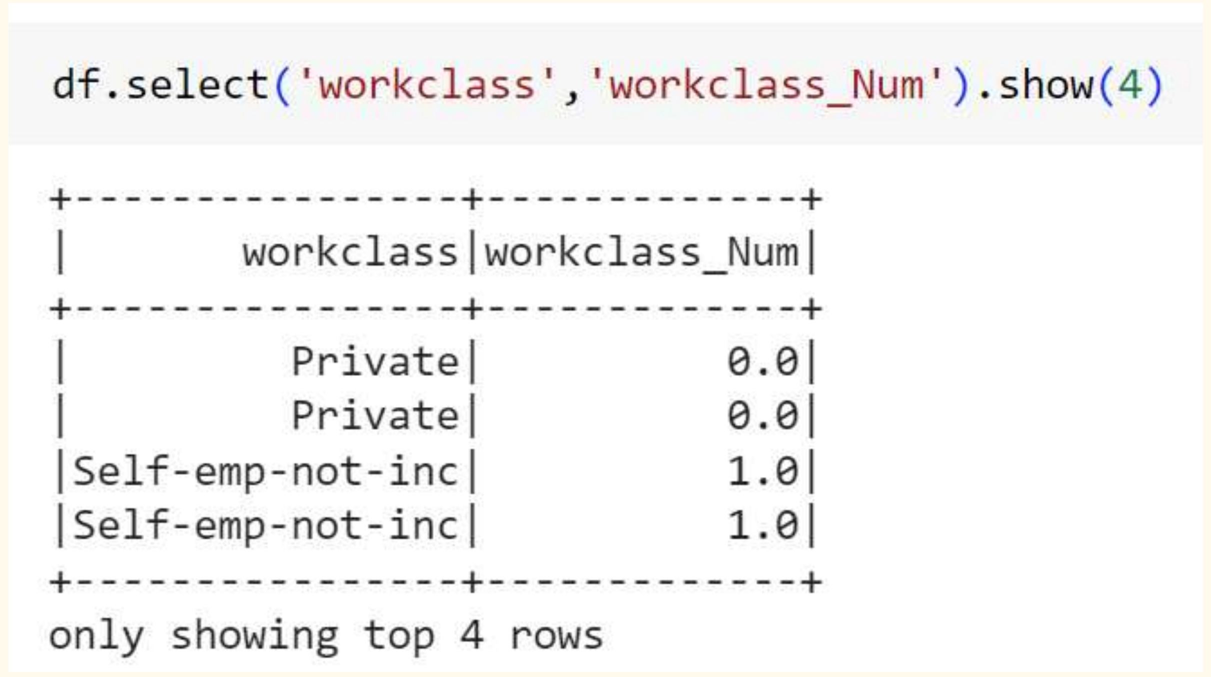
Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Converting categorical data to numerical form



```
from pyspark.ml.feature import StringIndexer
df=df.drop('education')
```

```
workclass_indexer = StringIndexer(inputCol="workclass", outputCol="workclass_Num").fit(df)
df = workclass_indexer.transform(df)
```



```
df.select('workclass','workclass_Num').show(4)
```

```
+----------------+-------------+
|       workclass|workclass_Num|
+----------------+-------------+
|         Private|          0.0|
|         Private|          0.0|
|Self-emp-not-inc|          1.0|
|Self-emp-not-inc|          1.0|
+----------------+-------------+
only showing top 4 rows
```

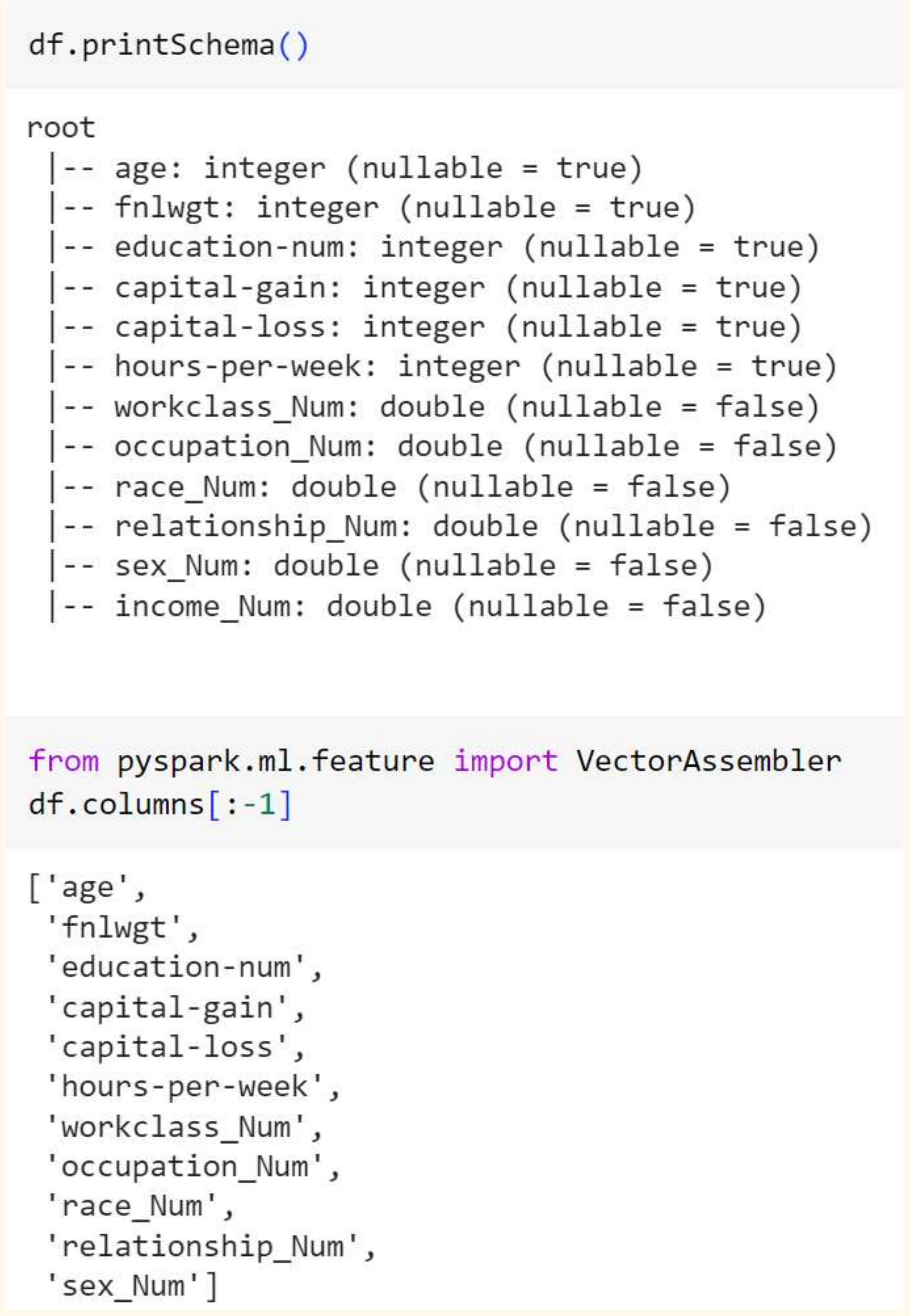
Converting categorical data to numerical form is a common and essential preprocessing step in machine learning.
Many machine learning algorithms are designed to work with numerical data. Converting categorical variables to numerical form allows you to use a wider range of algorithms without running into compatibility issues.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Apply a classification algorithm

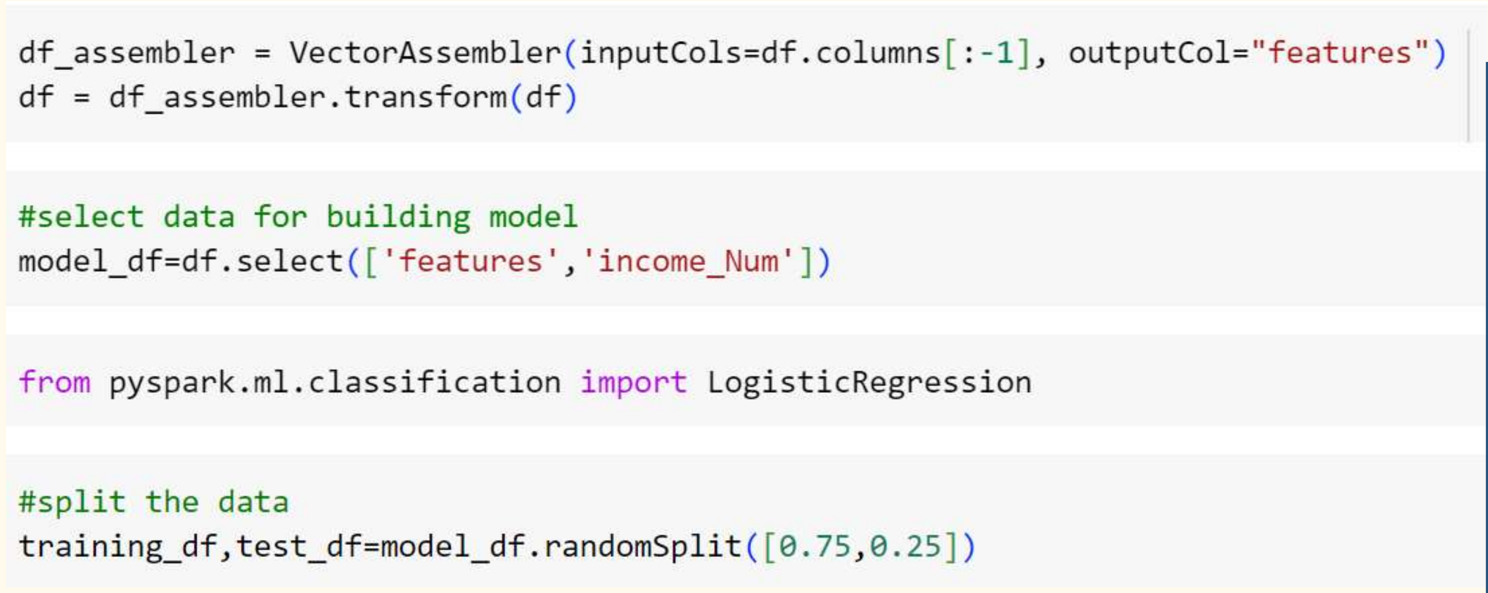


```
df.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- fnlwgt: integer (nullable = true)
 |-- education-num: integer (nullable = true)
 |-- capital-gain: integer (nullable = true)
 |-- capital-loss: integer (nullable = true)
 |-- hours-per-week: integer (nullable = true)
 |-- workclass_Num: double (nullable = false)
 |-- occupation_Num: double (nullable = false)
 |-- race_Num: double (nullable = false)
 |-- relationship_Num: double (nullable = false)
 |-- sex_Num: double (nullable = false)
 |-- income_Num: double (nullable = false)
```

```
from pyspark.ml.feature import VectorAssembler
df.columns[:-1]
```

```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'workclass_Num',
 'occupation_Num',
 'race_Num',
 'relationship_Num',
 'sex_Num']
```



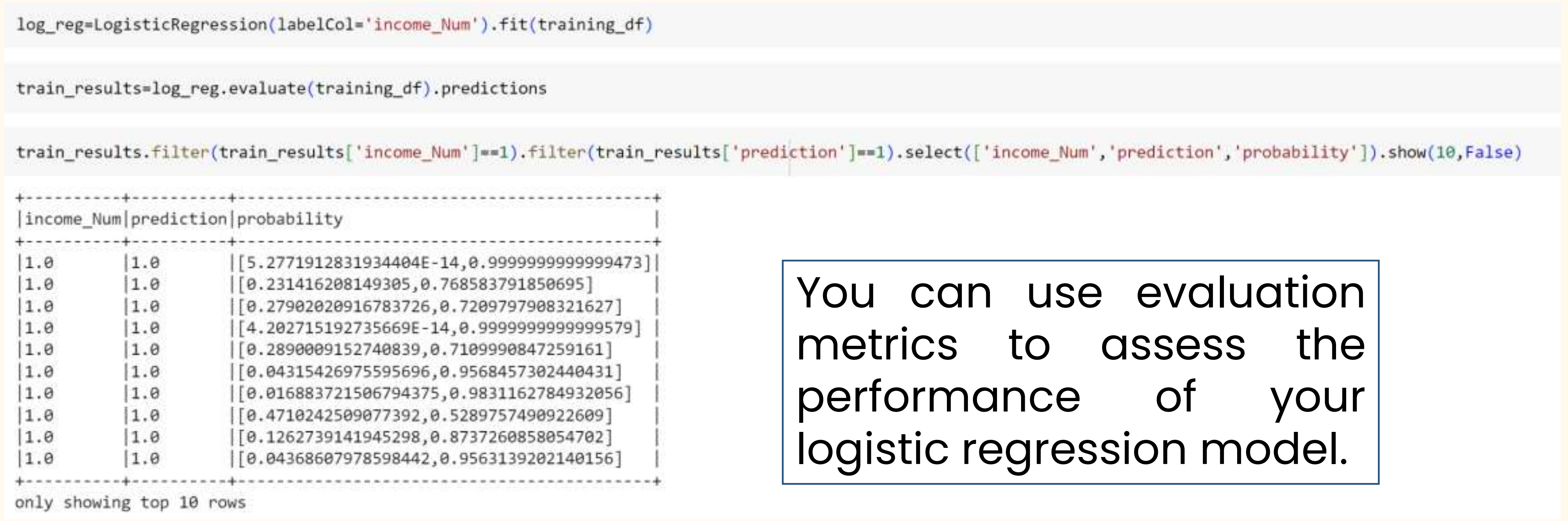
```
df_assembler = VectorAssembler(inputCols=df.columns[:-1], outputCol="features")
df = df_assembler.transform(df)
```

```
#select data for building model
model_df=df.select(['features','income_Num'])
```

```
from pyspark.ml.classification import LogisticRegression
```

```
#split the data
training_df,test_df=model_df.randomSplit([0.75,0.25])
```

You need to prepare your data by assembling features into a single vector column. If your data is not in this format, you can use **VectorAssembler** to create a new DataFrame with a vector column.



```
log_reg=LogisticRegression(labelCol='income_Num').fit(training_df)
```

```
train_results=log_reg.evaluate(training_df).predictions
```

```
train_results.filter(train_results['income_Num']==1).filter(train_results['prediction']==1).select(['income_Num','prediction','probability']).show(10,False)
```

```
+----------+----------+-------------------------------------------+
|income_Num|prediction|probability                                |
+----------+----------+-------------------------------------------+
|1.0       |1.0       |[5.2771912831934404E-14,0.9999999999999473]|
|1.0       |1.0       |[0.231416208149305,0.768583791850695]      |
|1.0       |1.0       |[0.27902020916783726,0.7209797908321627]   |
|1.0       |1.0       |[4.202715192735669E-14,0.9999999999999579] |
|1.0       |1.0       |[0.2890009152740839,0.7109990847259161]    |
|1.0       |1.0       |[0.043154269755956,0.9568457302440431]     |
|1.0       |1.0       |[0.016883721506794375,0.9831162784932056]  |
|1.0       |1.0       |[0.4710242509077392,0.5289757490922609]    |
|1.0       |1.0       |[0.1262739141945298,0.8737260858054702]    |
|1.0       |1.0       |[0.04368607978598442,0.9563139202140156]   |
+----------+----------+-------------------------------------------+
only showing top 10 rows
```

You can use evaluation metrics to assess the performance of your logistic regression model.

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

You can run the full example code

https://colab.research.google.com/drive/1RucEzI8NogTieUVCcVp_PPWKjopBOarE?usp=sharing

Co-funded by
the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

List of references

- Dasari, S., & Kaluri, R. (2023). Big Data Analytics, Processing Models, Taxonomy of Tools, V's, and Challenges: State-of-Art Review and Future Implications. Wireless Communications and Mobile Computing, 2023, 1-14. https://doi.org/10.1155/2023/3976302
- Ikegwu, A. C., Nweke, H. F., Anikwe, C. V., Alo, U. R., & Okonkwo, O. R. (2022). Big data analytics for data-driven industry: a review of data sources, tools, challenges, solutions, and research directions. Cluster Computing, 25(5), 3343-3387. https://doi.org/10.1007/s10586-022-03568-5
- Grover, P., & Kar, A. K. (2017). Big Data Analytics: A Review on Theoretical Contributions and Tools Used in Literature. Global Journal of Flexible Systems Management, 18(3), 203-229. https://doi.org/10.1007/s40171-017-0159-3

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Further reading

- Cesario, E. (2023). Big data analytics and smart cities: applications, challenges, and opportunities. Frontiers in Big Data, 6. https://doi.org/10.3389/fdata.2023.1149402
- Khan, M. A., Siddiqui, M. S., Rahmani, M. K. I., & Husain, S. (2022). Investigation of Big Data Analytics for Sustainable Smart City Development: An Emerging Country. IEEE Access, 10, 16028-16036. https://doi.org/10.1109/access.2021.3115987
- Cerquitelli, T., Migliorini, S., & Chiusano, S. (2021). Big Data Analytics for Smart Cities. Electronics, 10(12), 1439. https://doi.org/10.3390/electronics10121439
- Soomro, K., Bhutta, M. N. M., Khan, Z., & Tahir, M. A. (2019). Smart city big data analytics: An advanced review. WIREs Data Mining and Knowledge Discovery, 9(5). Portico. https://doi.org/10.1002/widm.1319

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Unit completed - What’s next?

- To consolidate your learning and reflect on the key concepts covered, please take a moment to complete this quiz.
- Your feedback and results will help you track your progress and support continuous improvement of the training experience.
- By completing this quiz, you will also become eligible to receive a certificate of successful training completion.
- Click this link to begin the quiz!

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

SMARCO

SMARt COmmunities Skills Development in Europe

www.smarco.eu

info@smarco.eu

We are social! Follow us on:

@smarcoproject

@smarcoproject

@smarcoproject

Co-funded by the European Union

smarco

Project 101186291 — SMARCO
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.