# Advanced Databases
# PySpark

This document includes instructions for installing and configurating Spark's Python binding, known as PySpark. It's a very popular alternative to the use of Scala or Java, and it's widely used in the field of Data Science. Using Python, we should be able to complete all the Spark sessions. Some of the exercises may have small differences compared to its Java counterparts, both due to different function signatures and the characteristics of the language.

## 1 Configuration and installation

### 1.1 Prerequisites

In order to have `PySpark` working in our machine, we need the following software installed:

- `Python` interpreter version 3.3 or higher installed [1]
- `PIP` package manager.

Usually PIP is already installed together with Python.If your Python distribution doesn't have it installed, you can install it from a shell using:

```
python -m ensurepip --default-pip
```

Both requirements can be installed on Windows, Linux or MacOs, using precompiled packages or package-managers on Linux and MacOs(as apt-get or brew). For windows, using the new WLS (Windows Linux Subsystem) is also an option in order to use the apt-get package manager.

### 1.2 Installing Spark and other requirements

We'll start first with a recommendation: the usage of virtual environments for doing the exercises on the sessions. Virtual environments are self-contained python installations that allow you to have different python

[1] Python 2.x is deprecated, so although it's possible to install pyspark in that version, we suggest updating to Python 3.x and using PySpark there.

interpreters with different packages installed in each of them. Moreover, it also has ways to share the packages required for running a certain script.

The module used to create and manage virtual environments is called `venv`. It's usually already installed on the latest python versions. The first step is to create the virtual environment using the following command on the root directory of your project:

```
python -m venv .env
```

This creates a `.env` directory in the directory (it will be conveniently hidden due to a period at the beginning of the name, but you can use any other name too, not necessarily using the dot in the beginning). The directory contains both an interpreter and the installed modules. The next step is to activate the environment using the following commands:

On Windows:

```
.env\Scrips\activate.bat
```

On Linux or Mac Os X:

```
source .env/bin/activate
```

Once the virtual environment is activated, the shell prompt will change to reflect the virtual environment name and all the references to `Python` will point to the virtual environment installation. So, any package installation you may perform now, it'll only affect the virtual environment interpreter. In a common use case, we'll have a different virtual environment for each session, and we would have only the modules required specifically for that session. But we can also have a single virtual environment for all the nine Spark sessions. It's just a matter of preference.

In any case, the common requirement we'll have for all the Spark sessions is the `pyspark` module, so we can start by installing it by using the following command:

```
python -m pip install pyspark
```

Note that additional packages can be installed using a configuration file. For instance, one can create a `requirements.txt` file listing all the required packages together with their version numbers, and install them simply using the command:

```
python -m pip install -r requirements.txt
```

## 2 Testing

Once PySpark is installed, you can test if it has been properly installed. Open a python shell execution `Python` on your system shell and type the following script:

```
import pyspark

spark = pyspark.sql.SparkSession.builder\
.appName('configurationtest').getOrCreate()

spark.range(5).collect()
```

If everything is installed properly, the execution of the script should display the following output in the shell:

```
[Row(id=0), Row(id=1), Row(id=2), Row(id=3), Row(id=4)]
```

If it's not working properly, make sure you have enabled your Python virtual environment where you installed PySpark. You can check the path of the current interpret being used using the following commands:

On Windows:

```
where python
```

On Linux or Mac Os X:

```
which python
```

If you have enabled your local virtual environment, the path will be pointing to its bin directory.