Apache Spark SQL
Structured Data Processing

Dr Fahimeh Jafari

f.jafari@uel.ac.uk

https://youtu.be/53FP8BaVzks

CN7031 – Lecture 6

University of East London

Outline

- Spark vs Hadoop (Recap)
- Spark Ecosystem
- What is Spark SQL
- What is DataFrame
- DataFrame Operations
- Spark SQL Queries
- Visualization with Matplotlib

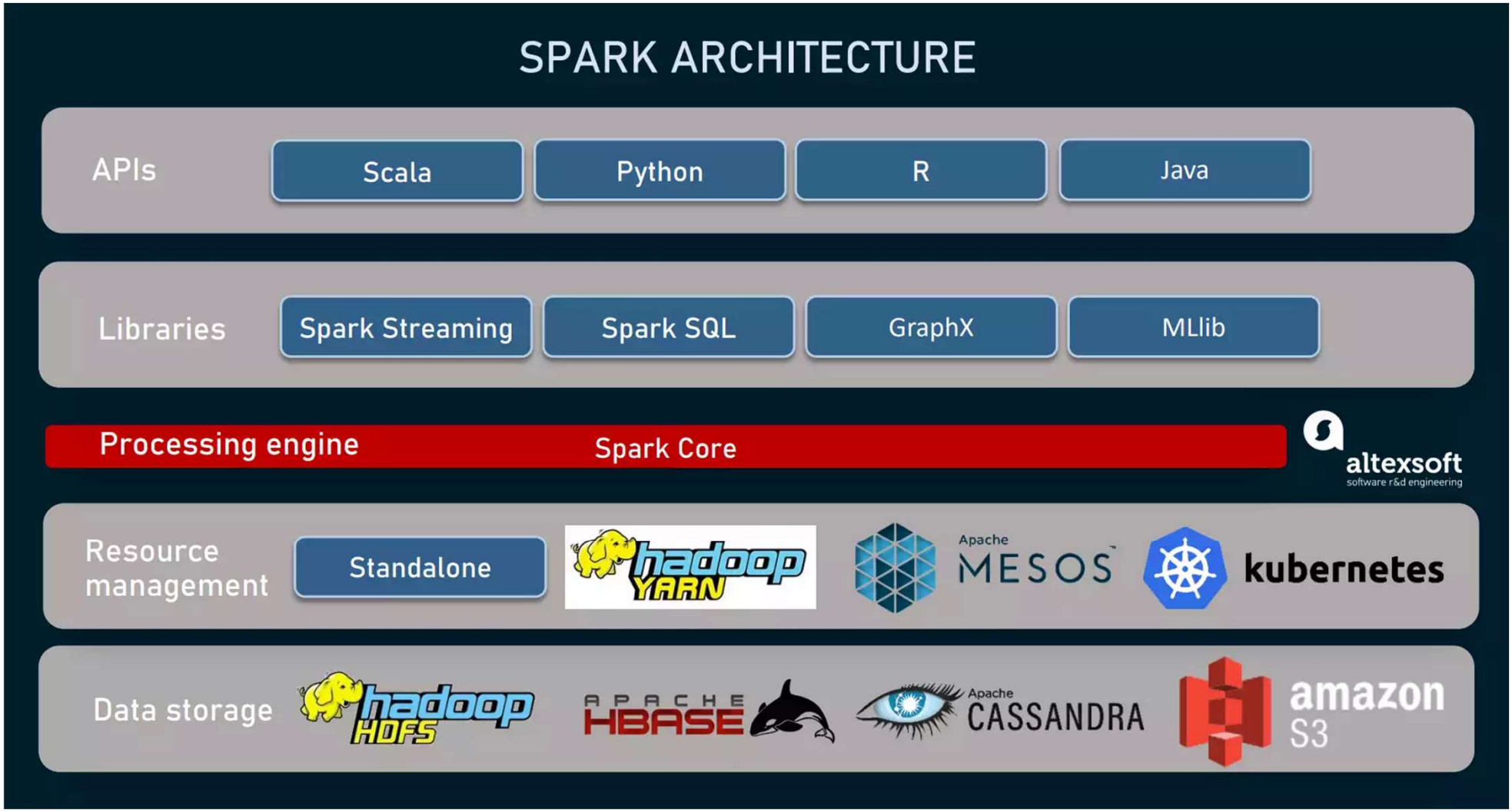
Learning Outcomes

- Be able to explain Spark SQL and its features
- Understand DataFrame and compare it with RDD
- Understand DataFrame operations
- Be able to write Spark SQL queries
- Use Matplotlib for visualization

Spark vs Hadoop (Recap)

Factors	Spark	Hadoop MapReduce
Speed	100x times than MapReduce	Faster than traditional system
Written In	Scala	Java
Data Processing	Batch / real-time / iterative / interactive /graph	Batch processing
Ease of Use	Compact & easier than Hadoop	Complex & lengthy
Caching	Caches the data in-memory & enhances the system performance	Doesn't support caching of data

Spark Ecosystem



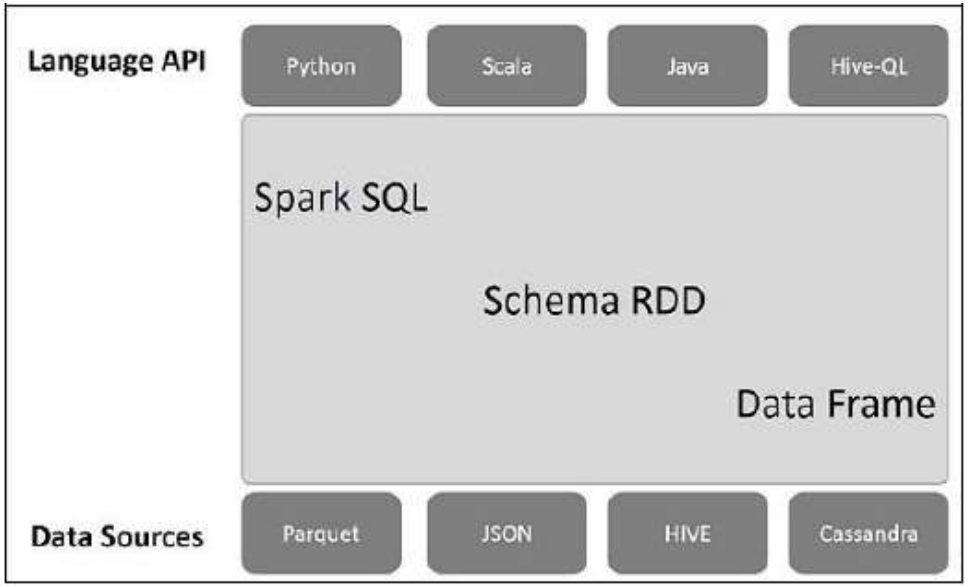
Spark Ecosystem

- **Engine — Spark Core**: It is the basic core component of Spark ecosystem on top of which the entire ecosystem is built. It performs the tasks of scheduling/monitoring and basic IO functionality
- **Management** — Spark cluster can be managed by Hadoop YARN, Mesos or Spark cluster manager.
- **Library** — Spark ecosystem comprises of Spark SQL (for running SQL like queries on RDD or data frame from external sources), Spark Mlib (for ML), Spark GraphX (for constructing graphs for better visualisation of data), Spark streaming (for batch processing and streaming of data in the same application)
- **APIs/Programming** can be done in Python, Java, Scala and R
- **Storage** — Data can be stored in HDFS, S3, local storage and it supports both SQL and NoSQL databases.

What is Spark SQL?

- Spark introduces a programming module for structured and semi-structured data processing called Spark SQL.
- Spark SQL is a component on top of Spark Core
- It introduces a new data abstraction called Schema RDD or DataFrame and can act as distributed SQL query engine.

- **Language API**: Spark is compatible with different languages- API (python, scala, java, HiveQL).
- **Schema RDD**: Spark Core is designed with RDD. Spark SQL works on Schema RDD which is called Data Frame.
- **Data Sources**: Data source for spark-core is a text file, Avro file, etc.
  The Data Sources for Spark SQL are Parquet file, JSON document, HIVE tables.

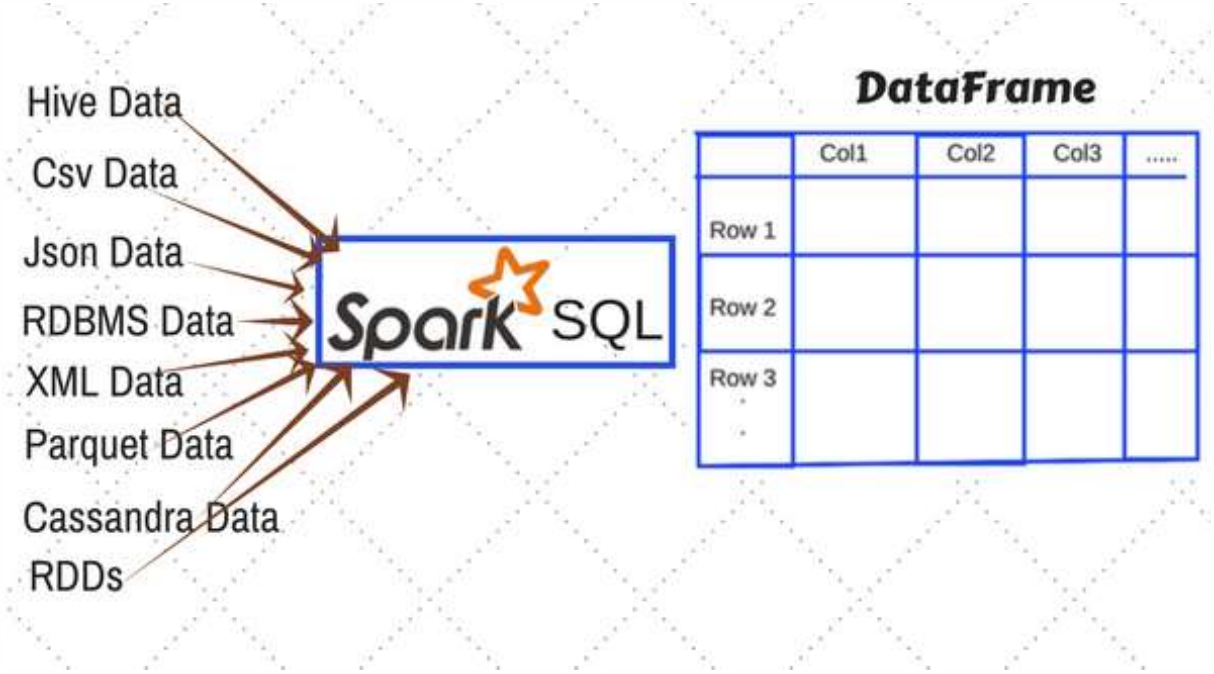


Features of Spark SQL

1. **Integrated:** Mix SQL queries with Spark programs. Spark SQL lets you query structured data as a distributed dataset (RDD) in Spark, with integrated APIs in Python, Scala and Java. This tight integration makes it easy to run SQL queries alongside complex analytic algorithms.
2. **Unified Data Access:** Load and query data from a variety of sources. SchemaRDDs provide a single interface for efficiently working with structured data, including Apache Hive tables, parquet files and JSON files.
3. **Hive Compatibility:** Run unmodified Hive queries on existing warehouses. Spark SQL reuses the Hive frontend and MetaStore, giving you full compatibility with existing Hive data, queries, and UDFs. Simply install it alongside Hive.
4. **Standard Connectivity:** Connect through JDBC or ODBC. Spark SQL includes a server mode with industry standard JDBC and ODBC connectivity.
5. **Scalability:** Use the same engine for both interactive and long queries. Spark SQL takes advantage of the RDD model to support mid-query fault tolerance, letting it scale to large jobs too. Do not worry about using a different engine for historical data.

What is DataFrame?

- A DataFrame is a distributed collection of data, which is organized into named columns. Conceptually, it is equivalent to relational tables with good optimization techniques.
- A DataFrame can be constructed from different sources such as Hive tables, Structured Data files, external databases, or existing RDDs.



Features of DataFrame

Here is a set of few characteristic features of DataFrame:

- Ability to process the data in the size of Kilobytes to Petabytes on a single node cluster to large cluster.
- Supports different data formats (Avro, csv, elastic search, and Cassandra) and storage systems (HDFS, HIVE tables, mysql, etc).
- State of art optimization and code generation through the Spark SQL Catalyst optimizer (tree transformation framework).
- Can be easily integrated with all Big Data tools and frameworks via Spark-Core.
- Provides API for Python, Java, Scala, and R Programming.

RDD vs DataFrame

- **RDD**: The RDD APIs have been on Spark in 1.0 release. It is a distributed collection of data elements spread across many machines in the cluster. RDDs are a set of Java or Scala objects representing data.
- **DataFrames**: Spark introduced DataFrames in Spark 1.3 release. A DataFrame is a distributed collection of data organized into named columns, like a table in a relational database, using off-heap storage.
- **DataSet**: Spark introduced Dataset in Spark 1.6 release. It is an extension of DataFrame API that provides the benefits of the Catalyst query optimizer and off heap storage mechanism.

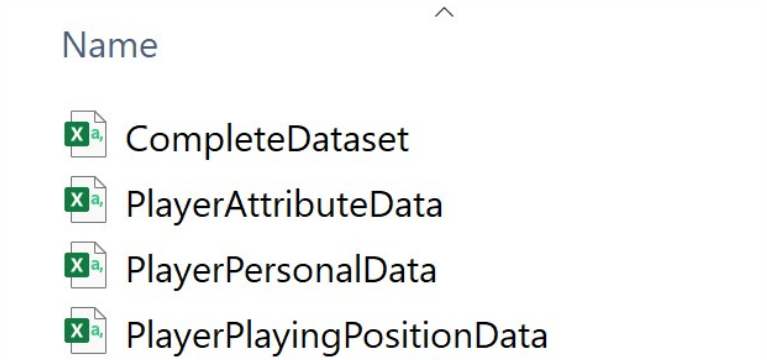
RDD disadvantages

1. **Outdated:** DataFrame and Dataset are distributed collections of data with the benefits of Spark SQL's optimized execution engine.

2. **Hard to Use:** RDD needs Python/Scala/Java coding, but DataFrame and Dataset need SQL-like queries, and anyone who knows SQL will understand it in one go.
3. **Slow Speed:** the main reason to not use RDD is its performance, which can be a major issue for some applications.

Example: FIFA 18

- We need to convert RDD/Hive to DataFrame, or find a relational dataset like csv, SQL, etc.
- Download data (FIFA 18 Player Dataset) from https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset.
- This dataset contains every player featuring in the famous Soccer Video Game, FIFA 18.
- Extracting the downloaded zip file, you have four `csv` files.



Example: FIFA 18

`PlayerAttributeData.csv` — This file contains Player performance attributes (Overall, Potential, Aggression, Agility etc.) indexed by player id.

`PlayerPersonalData.csv` — This file contains basic Player personal attributes (Nationality, Club, Photo, Age, Wage, Value etc.)

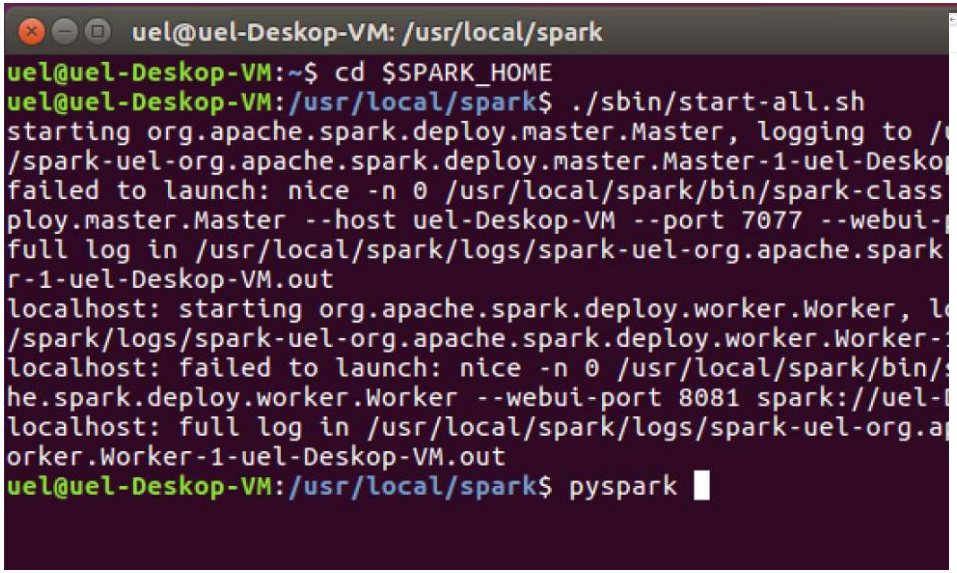
`PlayerPlayingPositionData.csv` — This file contains Player preferred position and ratings at all positions.

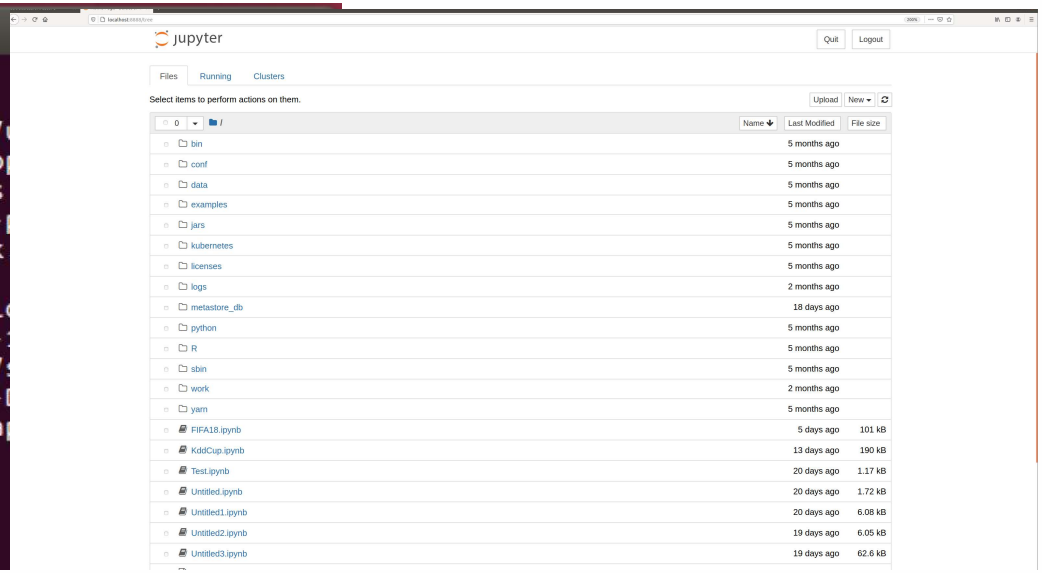
`CompleteDataset.csv` — This file is the complete dataset contains all information in above three dataset.

Launch PySpark in Ubuntu

You can lunch PySpark in Ubuntu and execute the commands in Jupyter notebook.

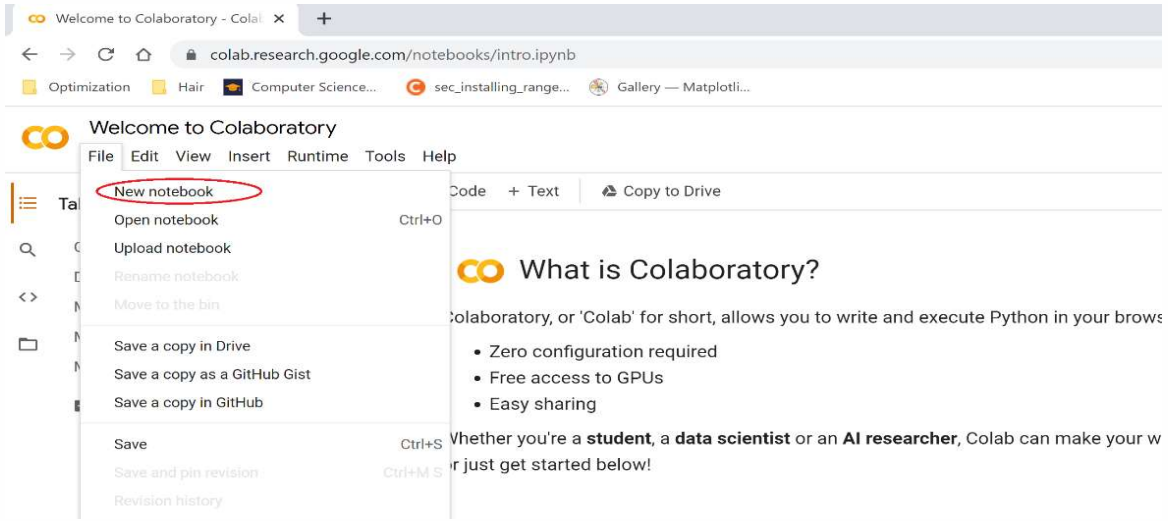
- Lunch VMWare
- Open and start up Ubuntu in VMWare
- Right the following commands in the terminal to lunch PySpark and Jupyter
- Open a new notebook in Jupyter and type your commands.





Launch Spark in Google Colab

- Open the Google Colab through https://colab.research.google.com/
- Open a new notebook to type the commands.
- Type and run the following commands in the editor to install and lunch Spark.



Old Version

```
# [1] download and install Spark in Google Colab
!sudo apt update
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-3.0.1/spark-3.0.1-bin-hadoop3.2.tgz
!tar xf spark-3.0.1-bin-hadoop3.2.tgz
!pip install -q findspark
```

```
# [2] Config JAVA and SPARK home
import os
os.environ["JAVA_HOME"]="/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"]="/content/spark-3.0.1-bin-hadoop3.2"
import findspark
findspark.init()
```

New Version

```
!pip3 install pyspark
```

University of East London

Starting Point: SparkSession

The entry point into all functionality in Spark is the SparkSession class. To create a basic `SparkSession`, just use `SparkSession.builder`:

```
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

`SparkSession` in Spark 2.0 provides builtin support for Hive features including the ability to write queries using HiveQL, access to Hive UDFs, and the ability to read data from Hive tables. To use these features, you do not need to have an existing Hive setup.

Creating DataFrames (DFs)

With a SparkSession, applications can create DataFrames from an existing RDD, from a Hive table, or from Spark data sources.

- Read Data from a JSON file:
  Create a DataFrame based on the content of a JSON file

```
# spark is an existing SparkSession
df = spark.read.json("examples/src/main/resources/people.json")
```

OR

```
df = spark.read.load("examples/src/main/resources/people.json", format="json")
```

Creating DataFrames (DFs) - continued

- Read Data from a CSV file: FIFA Example

```
fifa_df = spark.read.csv("/home/bigdata/Desktop/CompleteDataset.csv", inferSchema=True, header=True)
```

OR

```
fifa_df2 = spark.read.load("/home/bigdata/Desktop/CompleteDataset.csv", format="csv", inferSchema=True, header=True
```

Creating DataFrames (DFs) - continued

- Read Data from Hive

1) Spark SQL supports reading and writing data stored in Hive.
2) Hive has a large number of dependencies, these dependencies are not included in the default Spark distribution.
3) If Hive dependencies can be found on the classpath, Spark will load them automatically.

Note: Hive dependencies must be present on all of the worker nodes, as they will need access to the Hive serialization and deserialization libraries (SerDes) in order to access data stored in Hive.

https://spark.apache.org/docs/latest/sql-data-sources-hive-tables.html

Creating DataFrames (DFs) - continued

```
from os.path import join, abspath

from pyspark.sql import SparkSession
from pyspark.sql import Row

# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')

spark = SparkSession \
    .builder \
    .appName("Python Spark SQL Hive integration example") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()

# spark is an existing SparkSession
spark.sql("CREATE TABLE IF NOT EXISTS src (key INT, value STRING) USING hive")
spark.sql("LOAD DATA LOCAL INPATH 'examples/src/main/resources/kv1.txt' INTO TABLE src")

# Queries are expressed in HiveQL
spark.sql("SELECT * FROM src").show()
```

https://spark.apache.org/docs/latest/sql-data-sources-hive-tables.html

Untyped Dataset Operations (DF Operations)

DataFrames provide a domain-specific language for structured data manipulation in Scala, Java, Python and R.

DF operations are also referred as “untyped transformations” in contrast to “typed transformations” come with strongly typed Scala/Java Datasets.

We are going to explain some basic examples of structured data processing using Datasets.

NOTE: In Python, it’s possible to access a DataFrame’s columns either by attribute (`df.age`) or by indexing (`df['age']`).

For a complete list of the types of operations that can be performed on a Dataset refer to the API Documentation.

DF Operations: Show ()

Show () Displays the Dataset in a tabular form.

```
fifa_df = spark.read.csv("/home/bigdata/Desktop/CompleteDataset.csv", inferSchema=True, header=True)
```

```
fifa_df.show()
```

```
+---+----+----+----+----+----+----+----+----+--------------------+----+----+----+----+----+----+----+----+----+
|_c0|             Name|Age|               Photo|Nationality|                Flag|Overall|Potential|                C
lub|           Club Logo| Value| Wage|Special|Acceleration|Aggression|Agility|Balance|Ball control|Composure|Crossi
ng|Curve|Dribbling|Finishing|Free kick accuracy|GK diving|GK handling|GK kicking|GK positioning|GK reflexes|Heading
accuracy|Interceptions|Jumping|Long passing|Long shots|Marking|Penalties|Positioning|Reactions|Short passing|Shot p
ower|Sliding tackle|Sprint speed|Stamina|Standing tackle|Strength|Vision|Volleys| CAM|  CB| CDM|  CF|  CM|    ID| L
AM|  LB| LCB| LCM| LDM|  LF|  LM|  LS|  LW| LWB|Preferred Positions| RAM|  RB| RCB| RCM| RDM|  RF|  RM|  RS|  RW| R
WB|  ST|
+---+-----------------+---+--------------------+-----------+--------------------+-------+---------+---------------
---+--------------------+------+-----+-------+-----------+----------+-------+-------+-----------+---------+------
--+-----+---------+---------+------------------+---------+-----------+----------+--------------+-----------+------
---------+-------------+-------+-----------+----------+-------+---------+-----------+----------+-------------+-----
-----+--------------+-----------+-------+---------------+--------+------+-------+----+----+----+----+----+------+-
---+----+----+----+----+----+----+----+----+----+--------------------+----+----+----+----+----+----+----+----+----+
---+----+
|  0|Cristiano Ronaldo| 32|https://cdn.sofif...|   Portugal|https://cdn.sofif...|     94|       94|     Real Madric
CF|https://cdn.sofif...|€95.5M|€565K|   2228|          89|        63|     89|     63|          93|       95|      8
5|   81|       91|       94|                76|        7|         11|         15|            14|         11|
88|           29|     95|          77|         92|     22|       85|         95|       96|           83|         94|
```

DF Operations: printSchema ()

To have a look at the structure of the Dataframe, we'll use the `printSchema()` method.

```
fifa_df.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Photo: string (nullable = true)
 |-- Nationality: string (nullable = true)
 |-- Flag: string (nullable = true)
 |-- Overall: integer (nullable = true)
 |-- Potential: integer (nullable = true)
 |-- Club: string (nullable = true)
 |-- Club Logo: string (nullable = true)
 |-- Value: string (nullable = true)
 |-- Wage: string (nullable = true)
 |-- Special: integer (nullable = true)
 |-- Acceleration: string (nullable = true)
 |-- Aggression: string (nullable = true)
 |-- Agility: string (nullable = true)
 |-- Balance: string (nullable = true)
 |-- Ball control: string (nullable = true)
 |-- Composure: string (nullable = true)
 |-- Crossing: string (nullable = true)
 |-- Curve: string (nullable = true)
 |-- Dribbling: string (nullable = true)
```

DF Operations: Columns information

```
fifa_df.columns
```

```
['_c0',
 'Name',
 'Age',
 'Photo',
 'Nationality',
 'Flag',
 'Overall',
 'Potential',
 'Club',
```

```
fifa_df.count()
```

```
17981
```

```
len(fifa_df.columns)
```

```
75
```

DF Operations: Select

```
fifa_df.select('Name','Nationality','club').show()
```

```
+-----------------+-----------+-------------------+
|             Name|Nationality|               club|
+-----------------+-----------+-------------------+
|Cristiano Ronaldo|   Portugal|     Real Madrid CF|
|          L. Messi|  Argentina|       FC Barcelona|
|            Neymar|     Brazil|Paris Saint-Germain|
|         L. Suárez|    Uruguay|       FC Barcelona|
|          M. Neuer|    Germany|   FC Bayern Munich|
|    R. Lewandowski|     Poland|   FC Bayern Munich|
|            De Gea|      Spain| Manchester United|
|         E. Hazard|    Belgium|            Chelsea|
|          T. Kroos|    Germany|     Real Madrid CF|
|        G. Higuaín|  Argentina|           Juventus|
|      Sergio Ramos|      Spain|     Real Madrid CF|
|      K. De Bruyne|    Belgium|   Manchester City|
|       T. Courtois|    Belgium|            Chelsea|
|        A. Sánchez|      Chile|            Arsenal|
|         L. Modrić|    Croatia|     Real Madrid CF|
|           G. Bale|      Wales|     Real Madrid CF|
|         S. Agüero|  Argentina|   Manchester City|
|      G. Chiellini|      Italy|           Juventus|
|         G. Buffon|      Italy|           Juventus|
|         P. Dybala|  Argentina|           Juventus|
+-----------------+-----------+-------------------+
only showing top 20 rows
```

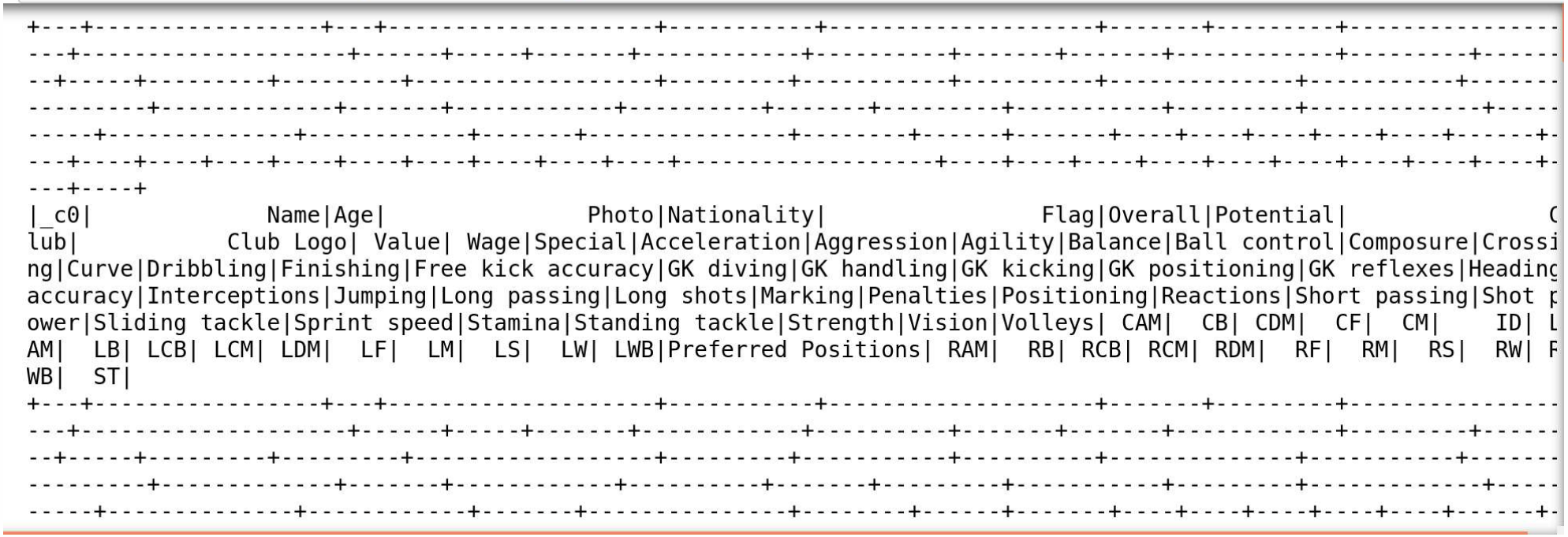
```
fifa_df.select('Name','Long shots').distinct().show()
```

```
+-----------------+----------+
|             Name|Long shots|
+-----------------+----------+
|Cristiano Ronaldo|        92|
|       J. Cuadrado|        80|
|       M. Brozović|        79|
|           A. Rami|        58|
|        D. Abraham|        65|
|      Borja Bastón|        73|
|        J. Montero|        68|
|       T. Barnetta|        74|
|           Wallace|        26|
|        A. Barreca|        42|
|     Y. Benalouane|        39|
|           Juankar|        64|
|         D. Appiah|        38|
|    Rafael Martins|        69|
|           Granell|        77|
|      A. Cornelius|        68|
|          J. Henry|        75|
|         M. Ozdoev|        69|
|             Fábio|        58|
|       T. Dingomé|        60|
+-----------------+----------+
only showing top 20 rows
```

DF Operations: Filter

This operation applies a filter on the existing DataFrame. For example, select people older than 21.

```
fifa_df.filter(fifa_df['age'] > 21).show()
```



University of East London

DF Operations: GroupBy

```
fifa_df.groupBy("age").count().show()
```

```
+---+-----+
|age|count|
+---+-----+
| 29| 1121|
| 30|  804|
| 34|  272|
| 28| 1051|
| 22| 1324|
| 35|  191|
| 16|   13|
| 47|    1|
| 43|    2|
| 31|  671|
| 18|  672|
| 27| 1152|
| 17|  258|
| 26| 1202|
| 19| 1069|
| 23| 1394|
| 41|    3|
| 38|   36|
| 40|    8|
| 25| 1522|
+---+-----+
only showing top 20 rows
```

Running SQL Queries (1)

We need to first register the DataFrame as a SQL temporary view and then define the queries on the view.

```
# Register the DataFrame as a SQL temporary view
fifa_df.createOrReplaceTempView("FifaView")
```

```
sqlDF = spark.sql("SELECT * FROM FifaView")
sqlDF.show()
```

```
15|   14|       30|       13|                11|       91|         90|        95|             91|         89|
25|           30|     78|          59|        16|     10|       47|         12|       85|           55|        25|
11|          61|     44|            10|      83|    70|     11|null|null|null|null|null|167495|null|null|null|nul
l|null|null|null|null|null|null|                 GK |null|null|null|null|null|null|null|null|null|null|null|
|  5|   R. Lewandowski| 28|https://cdn.sofif...|     Poland|https://cdn.sofif...|     91|       91|   FC Bayern Mur
ich|https://cdn.sofif...|  €92M|€355K|   2143|          79|        80|     78|     80|          89|       87|
62|   77|       85|       91|                84|       15|          6|        12|              8|         10|
85|           39|     84|          65|        83|     25|       81|         91|       91|           83|        88|
19|          83|     79|            42|      84|    78|     87|84.0|57.0|62.0|87.0|78.0|188545|84.0|58.0|57.0|78.
0|62.0|87.0|82.0|88.0|84.0|61.0|                 ST |84.0|58.0|57.0|78.0|62.0|87.0|82.0|88.0|84.0|61.0|88.0|
|  6|           De Gea| 26|https://cdn.sofif...|     Spain|https://cdn.sofif...|     90|       92|  Manchester Uni
ted|https://cdn.sofif...|€64.5M|€215K|   1458|          57|        38|     60|     43|          42|       64|
17|   21|       18|       13|                19|       90|         85|        87|             86|         90|
21|           30|     67|          51|        12|     13|       40|         12|       88|           50|        31|
13|          58|     40|            21|      64|    68|     13|null|null|null|null|null|193080|null|null|null|nul
l|null|null|null|null|null|null|                 GK |null|null|null|null|null|null|null|null|null|null|null|
|  7|        E. Hazard| 26|https://cdn.sofif...|    Belgium|https://cdn.sofif...|     90|       91|            Chel
sea|https://cdn.sofif...|€90.5M|€295K|   2096|          93|        54|     93|     91|          92|       87|
80|   82|       93|       83|                79|       11|         12|          6|              8|          8|
```

Spark SQL and DataFrames:
https://spark.apache.org/docs/2.3.0/sql-programming-guide.html

University of East London

Running SQL Queries (2)

Count the number of players by age.

```
sqlDF = spark.sql("SELECT age, count(*) as count from FifaView GROUP BY age")
sqlDF.show()
```

```
+---+-----+
|age|count|
+---+-----+
| 31|  671|
| 34|  272|
| 28| 1051|
| 26| 1202|
| 27| 1152|
| 44|    2|
| 22| 1324|
| 47|    1|
| 16|   13|
| 20| 1245|
| 40|    8|
| 19| 1069|
| 41|    3|
| 43|    2|
| 37|   69|
| 17|  258|
| 35|  191|
| 39|   20|
| 23| 1394|
| 38|   36|
+---+-----+
only showing top 20 rows
```

Spark SQL and DataFrames:
https://spark.apache.org/docs/2.3.0/sql-programming-guide.html

University of East London

Running SQL Queries (3)

Count the number of players in each club.

```
sqlDF = spark.sql("SELECT club, count(*) FROM FifaView GROUP BY club")
sqlDF.show()
```

```
+--------------------+--------+
|                club|count(1)|
+--------------------+--------+
|             Palermo|      28|
|         Yeovil Town|      21|
|  1. FC Union Berlin|      27|
|  Santiago Wanderers|      20|
|               Carpi|      30|
|Evkur Yeni Malaty...|      30|
|          Sagan Tosu|      25|
|            FC Basel|      25|
|  Argentinos Juniors|      28|
|       Karlsruher SC|      27|
|  Lorca Deportiva CF|      29|
|     SC Paderborn 07|      28|
|     Cheltenham Town|      28|
|San Lorenzo de Al...|      28|
|         SC Freiburg|      32|
|  SpVgg Unterhaching|      28|
```

Spark SQL and DataFrames:
https://spark.apache.org/docs/2.3.0/sql-programming-guide.html

Running SQL Queries (4)

Count the number of players in each club and displays those have more than 33 members.

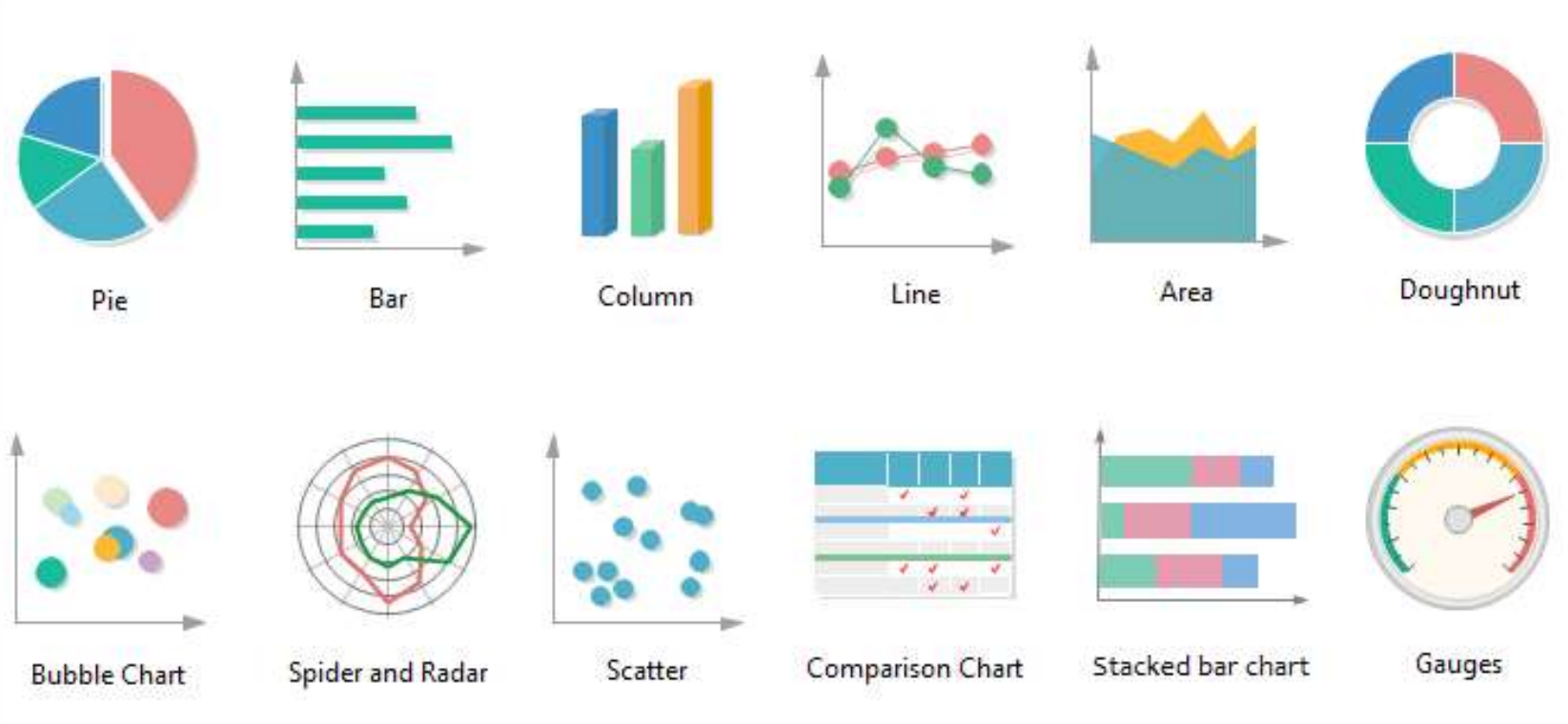
```
sqlDF = spark.sql("SELECT club, count(*) FROM FifaView GROUP BY club HAVING Count(*) > 33")
sqlDF.show()
```

```
+------------------+--------+
|              club|count(1)|
+------------------+--------+
| Manchester United|      34|
|     UD Las Palmas|      34|
|              null|     248|
|Olympique Lyonnais|      34|
|     VfL Wolfsburg|      34|
|          OGC Nice|      34|
|     Villarreal CF|      35|
|         FC Nantes|      34|
| Borussia Dortmund|      34|
+------------------+--------+
```

Spark SQL and DataFrames:
https://spark.apache.org/docs/2.3.0/sql-programming-guide.html

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.



Visualization with Python

Here are steps to visualize the output of Spark SQL using Matplotlib:

1. Import required libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2. Convert SQL dataframe to Pandas dataframe

```
pandas_df = sqlDF.toPandas()
```

3. Select parameters need to be visualised.
4. Decide which chart is appropriate for presenting your outpt.
5. Use pandas.DataFrame.plot()to define your chart.

Visualization with Python

You can also create these plots using the method `DataFrame.plot ()`

Syntax

```
DataFrame.plot (x, y, kind)
```

x : label or position, default None
y : label, position or list of label, positions, default None
Allows plotting of one column versus another
kind : str

- 'line' : line plot (default)
- 'bar' : vertical bar plot
- 'barh' : horizontal bar plot
- 'hist' : histogram
- 'box' : boxplot
- 'kde' : Kernel Density Estimation plot
- 'density' : same as 'kde'
- 'area' : area plot
- 'pie' : pie plot
- 'scatter' : scatter plot
- 'hexbin' : hexbin plot

Each plot kind has a corresponding method on the `DataFrame.plot` accessor:

```
df.plot(kind='line') ≈ df.plot.line()
```

https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.plot.html

University of East London

Bar Chart

A bar chart is a chart that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.

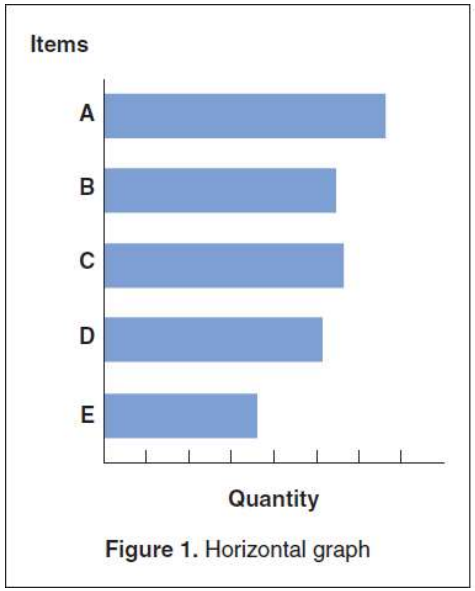


Figure 1. Horizontal graph

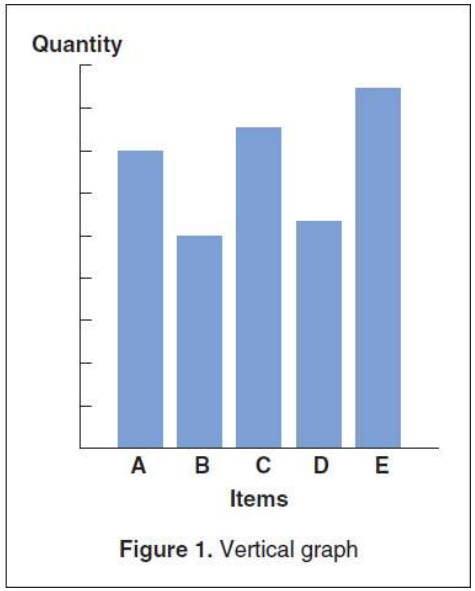


Figure 1. Vertical graph

Syntax

```
DataFrame.plot (x:str, y:str, kind='bar', logy:boolean)
```

OR

```
DataFrame.plot.bar (x:str, y:str, logy:boolean)
```

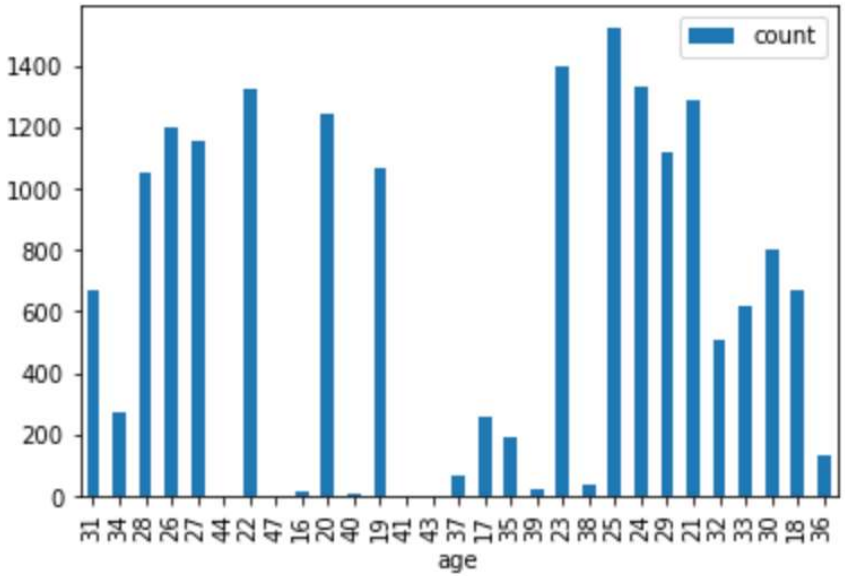
Bar Chart Example

```
sqlDF = spark.sql("SELECT age, count(*) as count from FifaView GROUP BY age")
sqlDF.show()
```

```
pandas_df = sqlDF.toPandas()
```

```
pandas_df.plot(x ='age', y='count', kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa24f2c7580>
```



University of East London

Bar Chart – Sorting Values

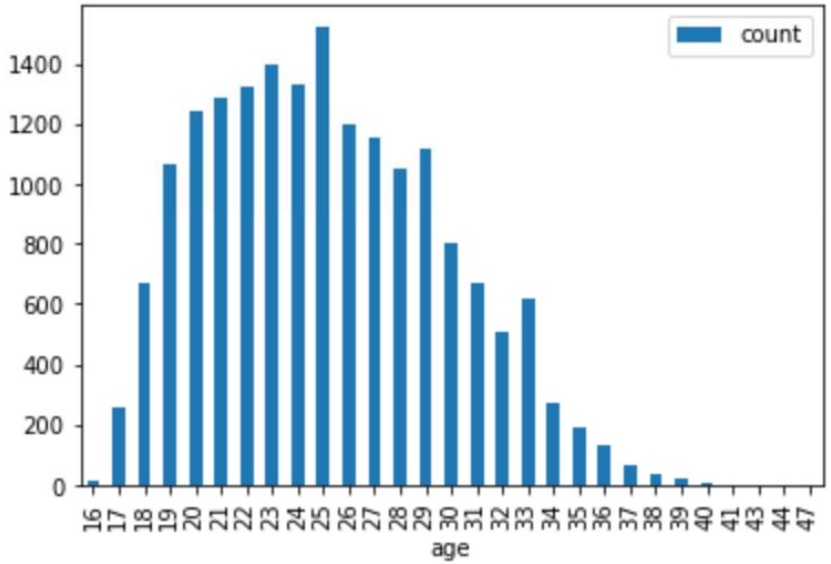
Syntax

```
DataFrame.sort_values (by:str, ascending:boolean)
```

Example

```
pandas_df.sort_values(by='age',ascending=True).plot(x ='age', y='count', kind = 'bar')
```

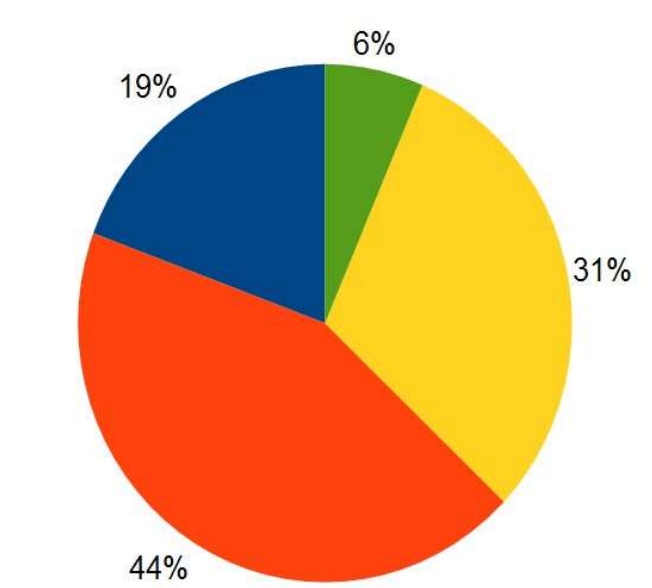
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7303ebb6a0>
```



https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html

Pie Chart

A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice, is proportional to the quantity it represents.



Syntax

```
DataFrame.plot (x, y, kind='pie')
```

OR

```
DataFrame.plot.pie (x, y)
```

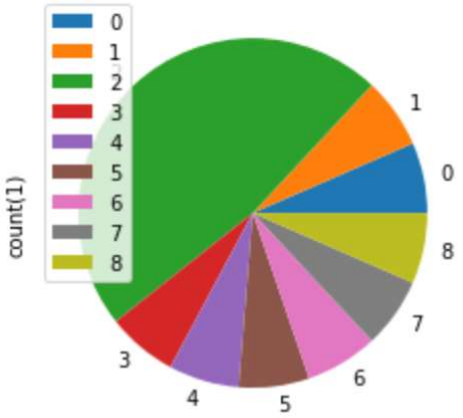
Pie Chart Example

```
sqlDF = spark.sql("SELECT club, count(*) FROM FifaView GROUP BY club HAVING Count(*) > 33")
sqlDF.show()
```

```
pandas_df = sqlDF.toPandas()
```

```
pandas_df.plot(x ='club', y='count(1)', kind = 'pie')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ace8480d0>
```



Summary

- Introduced DataFrame
- Compared RDD vs DataFrame
- DataFrame Operations
- Learned SparkSQL
- Practiced SparkSQL Queries
- Learned Visualization with Matplotlib in Python

University of East London