

## 1. 初始化 SparkSession

### 初始化

SparkSession 用于创建数据帧，将数据帧注册为表，  
执行 SQL 查询，缓存表及读取 Parquet 文件。

```
> from pyspark.sql import SparkSession

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

## 3. 查阅数据信息

### 查阅 spark Dataframe 的信息

`df.dtypes` # 返回 df 的列名与数据类型

`df.show()` # 显示 df 的内容

`df.head(n)` # 返回前 n 行数据

`df.first()` # 返回第 1 行数据

`df.take(n)` # 返回前 n 行数据

`df.schema` # 返回 df 的 Schema

`df.describe().show()` # 汇总统计数据

`df.columns` # 返回 df 的列名

`df.count()` # 返回 df 的行数

`df.distinct().count()` # 返回 df 中不重复的行数

`df.printSchema()` # 返回 df 的 Schema

`df.explain()` # 返回逻辑与实体方案

## 4. 重复值

`dropDuplicates` 函数

`df = df.dropDuplicates()`

## 2. 创建数据帧

### 2.1 从 RDD 创建

```
from pyspark.sql.types import *

推断 Schema

sc = spark.sparkContext

lines = sc.textFile("people.txt")

parts = lines.map(lambda l: l.split(","))

people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

peopledf = spark.createDataFrame(people)
```

### 指定 Schema

```
people = parts.map(lambda p: Row(name=p[0], age=int(p[1].strip())))

schemaString = "name age"

fields = [StructField(field_name, StringType(), True) \
           for field_name in schemaString.split()]

schema = StructType(fields)

spark.createDataFrame(people, schema).show()
```

```
name age
Mine  28
Filip 29
Jonathan 30
```

### 2.2 从 Spark 数据源创建

#### JSON

```
df = spark.read.json("customer.json")

df.show()

address  age  firstName  lastName  phoneNumber
[NewYork,10021,N...  25  John  Smith  [[212 555-1234,ho...]]
[NewYork,10021,N...  21  Jane  Doe    [[322 888-1234,ho...]]
```

`df2 = spark.read.load("people.json", format="json")`

#### Parquet 文件

`df3 = spark.read.load("users.parquet")`

#### 文本文件

`df4 = spark.read.text("people.txt")`

## 5. 查询

```
from pyspark.sql import functions as F
```

### 5.1 Select

# 显示 firstName 列的所有条目

```
df.select("firstName").show()

df.select("firstName", "lastName").show()
```

# 显示 firstName、age 的所有条目和类型

```
df.select("firstName", "age", explode("phoneNumber") \
    .alias("contactInfo")) \
    .select("contactInfo.type", "firstName", "age") \
    .show()
```

# 显示 firstName 和 age 列的所有记录，并对 age 记录添加 1

```
df.select(df["firstName"], df["age"]+1).show()
```

# 显示所有小于 24 岁的记录

```
df.select(df['age'] > 24).show()
```

### 5.2 When

# 显示 firstName，且大于 30 岁显示 1，小于 30 岁显示 0

```
df.select("firstName", \
    F.when(df.age > 30, 1).otherwise(0)).show()
```

# 显示符合指定条件的 firstName 列的记录

```
df[df.firstName.isin("Jane", "Boris")].collect()
```

### 5.3 Like

# 显示 lastName 列中包含 Smith 的 firstName 列的记录

```
df.select("firstName", \
    df.lastName.like("Smith")).show()
```



Spark 是基于内存计算的大数据并行计算框架。它包含 MapReduce 计算模型，而且高效地支持更多计算模式，包括交互式查询和流处理。Spark 适用于各种各样原先需要多种不同的分布式平台的场景，包括批处理、迭代算法、交互式查询、流处理。Spark 生态系统已经发展成为一个包含多个子项目的集合，其中包含 SparkSQL、Spark Streaming、GraphX/GraphFrame、MLlib、SparkR 等子项目。Spark SQL 是 Apache Spark 处理结构化数据的模块。

## 5.4 Startswith - Endswith

```
# 显示 lastName 列中以 Sm 开头的 firstName 列的记录
df.select("firstName", df.lastName.startswith("Sm")).show()

# 显示以 th 结尾的 lastName
df.select(df.lastName.endsWith("th")).show()
```

## 5.5 Substring

```
# 返回 firstName 的子字符串
df.select(df.firstName.substr(1,3).alias("name")).collect()
```

## 5.6 Between

```
# 显示介于 22 岁至 24 岁之间的 age 列的记录
df.select(df.age.between(22, 24)).show()
```

## 10. 替换缺失值 / replace 操作

```
df.na.fill(50).show() # 用一个值替换空值
df.na.drop().show() # 去除 df 中为空值的行
df.na.replace(10, 20).show() # 用一个值替换另一个值
```

## 11. 重分区 / repartition 重分区

```
# 将 df 拆分为 10 个分区
df.repartition(10).rdd.getNumPartitions()

# 将 df 合并为 1 个分区
df.coalesce(1).rdd.getNumPartitions()
```

## 14. 终止 SparkSession

终止 spark session  
`spark.stop()`

## 6. 添加、修改、删除列

### 6.1 添加列

```
df = df.withColumn('city', df.address.city) \
        .withColumn('postalCode', df.address.postalCode) \
        .withColumn('state', df.address.state) \
        .withColumn('streetAddress', df.address.streetAddress) \
        .withColumn('telePhoneNumber', explode(df.phoneNumber.number)) \
        .withColumn('telephoneType', explode(df.phoneNumber.type))
```

### 6.2 修改列

```
df = df.withColumnRenamed('telePhoneNumber', 'phoneNumber')
```

### 6.3 删除列

```
df = df.drop("address", "phoneNumber")
df = df.drop(df.address).drop(df.phoneNumber)
```

## 12. 运行 SQL 查询

将数据帧注册为视图

```
peopledf.createGlobalTempView("people")
df.createTempView("customer")
df.createOrReplaceTempView("customer")
```

查询视图

```
df5 = spark.sql("SELECT * From customer").show()
peopledf2 = spark.sql("SELECT * From global_temp.people").show()
```

## 13. 输出

数据结构

```
rdd1 = df.rdd # 将 df 转换为 RDD
df.toJSON().first() # 将 df 转换为 RDD 字符串
df.toPandas() # 将 df 的内容转为 Pandas 的数据帧
```

保存至文件

```
df.select("firstName", "city").write.save("nameAndCity.parquet")
df.select("firstName", "age").write.save("namesAndAges.json", format="json")
```

## 7. 分组 / groupBy 操作

```
# 按 age 列分组, 统计每组人数
df.groupBy("age").count().show()
```

## 8. 筛选 / filter 筛选

```
# 按 age 列筛选, 保留年龄大于 24 岁的
df.filter(df["age"] > 24).show()
```

## 9. 排序 / sort 与 orderBy 操作

```
peopledf.sort(peopledf.age.desc()).collect()
df.sort("age", ascending=False).collect()
df.orderBy(["age", "city"], ascending=[0, 1]).collect()
```



Spark SQL 速查表

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

扫码回复“数据科学”

参考 | DataCamp Cheatsheet

下载最新全套速查表

Spark  
SQL

Spark  
Streaming

MLlib  
(machine  
learning)

GraphX  
(graph)

Apache Spark

Show Me AI

## 数据科学工具库速查表



**NumPy** 是 Python 数据科学计算的核心库，提供了高性能多维数组对象及处理数组的工具。使用以下语句导入 NumPy 库：

```
import numpy as np
```



**SciPy** 是基于 NumPy 创建的 Python 科学计算核心库，提供了众多数学算法与函数。



**Pandas** 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
import pandas as pd
```



**Matplotlib** 是 Python 的二维绘图库，用于生成符合出版质量或跨平台交互环境的各类图形。

```
import matplotlib.pyplot as plt
```



**Seaborn** 是基于 matplotlib 开发的高阶 Python 数据可视图库，用于绘制优雅、美观的统计图形。使用下列别名导入该库：

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```



**Bokeh** 是 Python 的交互式可视图库，用于生成在浏览器里显示的大规模数据集高性能可视图。Bokeh 的中间层通用 **bokeh.plotting** 界面主要为两个组件：数据与图示例。

```
from bokeh.plotting import figure
```

```
from bokeh.io import output_file, show
```



**PySpark** 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

## AI 垂直领域工具库速查表



**Scikit-learn** 是开源的 Python 库，通过统一的界面实现机器学习、预处理、交叉验证及可视化算法。



**Keras** 是强大、易用的深度学习库，基于 Theano 和 TensorFlow 提供了高阶神经网络 API，用于开发和评估深度学习模型。



“TensorFlow™ is an open source software library for numerical computation using data flow graphs.” **TensorFlow** 是 Google 公司开发的机器学习架构，兼顾灵活性和扩展性，既适合用于工业生产也适合用于科学研究。



**PyTorch** 是 Facebook 团队 2017 年初发布的深度学习框架，有利于研究人员、爱好者、小规模项目等快速搞出原型。**PyTorch** 也是 Python 程序员最容易上手的深度学习框架。



**Hugging Face** 以开源的 NLP 预训练模型库 **Transformers** 而广为人知，目前 GitHub Star 已超过 54000+。**Transformers** 提供 100+ 种语言的 32 种预训练语言模型，简单，强大，高性能，是新手入门的不二选择。



**OpenCV** 是一个跨平台计算机视觉库，由 C 函数 /C++ 类构成，提供了 Python、MATLAB 等语言的接口。**OpenCV** 实现了图像处理和计算机视觉领域的很多通用算法。

## 编程语言速查表



**SQL** 是管理关系数据库的结构化查询语言，包括数据的增删查改等。作为数据分析的必备技能、岗位 JD 的重要关键词，SQL 是技术及相关岗位同学一定要掌握的语言。



**Python** 编程语言简洁快速、入门简单且功能强大，拥有丰富的第三方库，已经成为大数据和人工智能领域的主流编程语言。

More...

## AI 知识技能速查表



**Jupyter Notebook** 交互式计算环境，支持运行 40+ 种编程语言，可以用来编写漂亮的交互式文档。这个教程把常用的基础功能讲解得很清楚，对新手非常友好。



**正则表达式** 非常强大，能匹配很多规则的文本，常用于文本提取和爬虫处理。这也是一门令人难以捉摸的语言，字母、数字和符号堆在一起，像极了“火星文”。

More...



ShowMeAI 速查表 (©2021)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

数据科学工具库速查表

扫码回复“数据科学”

获取最新全套速查表

AI 垂直领域工具库速查表

扫码回复“工具库”

获取最新全套速查表

编程语言速查表

扫码回复“编程语言”

获取最新全套速查表

AI 知识技能速查表

扫码回复“知识技能”

获取最新全套速查表