Distributed Data Systems

DIANE WOODBRIDGE, PH.D

Announcement

Individual Assignment

- HW 3 - Feb 23rd (Extended)
- HW 4 - March 3rd (Extended)

Group Assignment

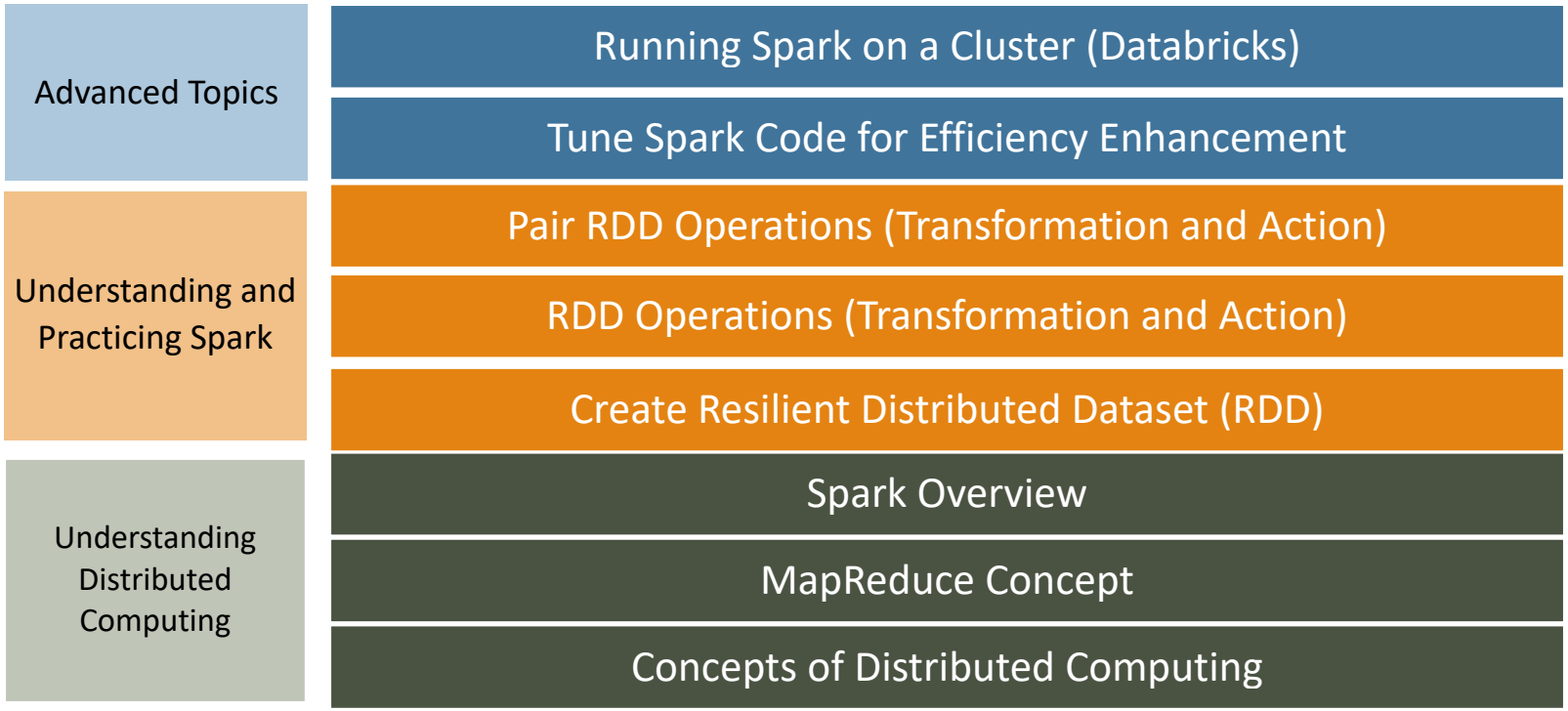
- Task 2 (Workflow for data collection, pre-processing and storage) - Feb 25th

Quiz 2

- Feb 28th 9 AM

Review

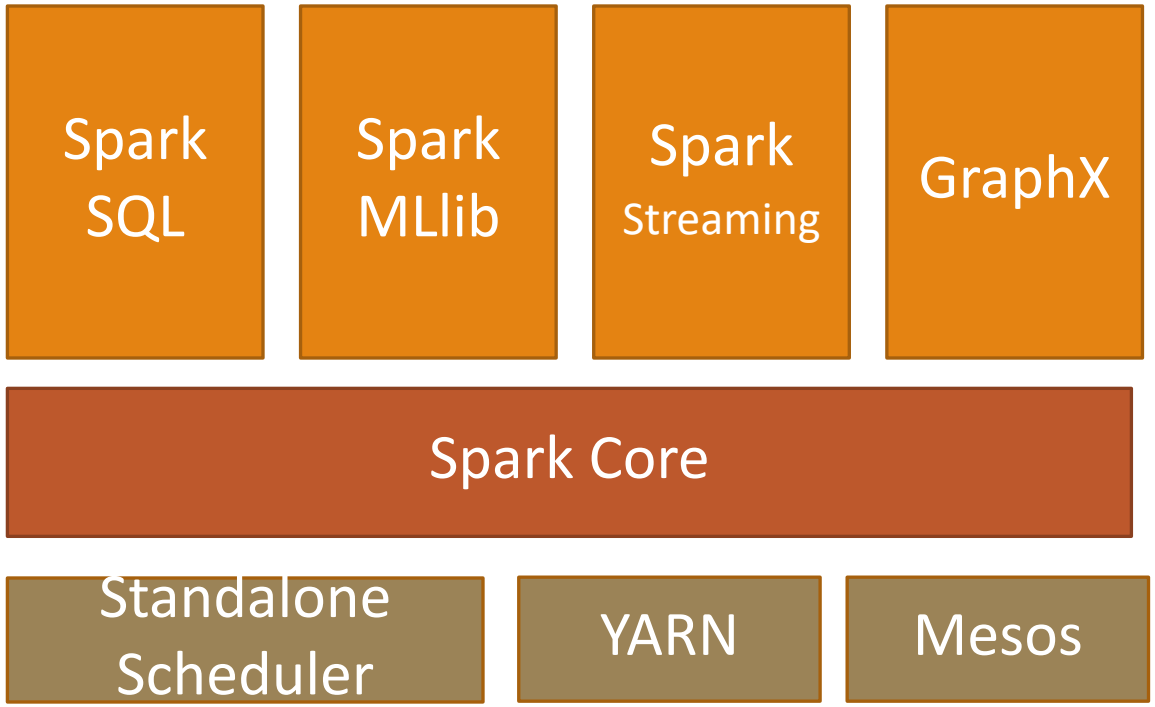
Apache Spark



Review

Spark Stack

- Cluster Managers
- Spark Core
- Spark SQL
- Spark Mllib
- Spark Streaming
- GraphX



Contents

Spark SQL

- Creating DataFrames
- DataFrame API Basics

Contents

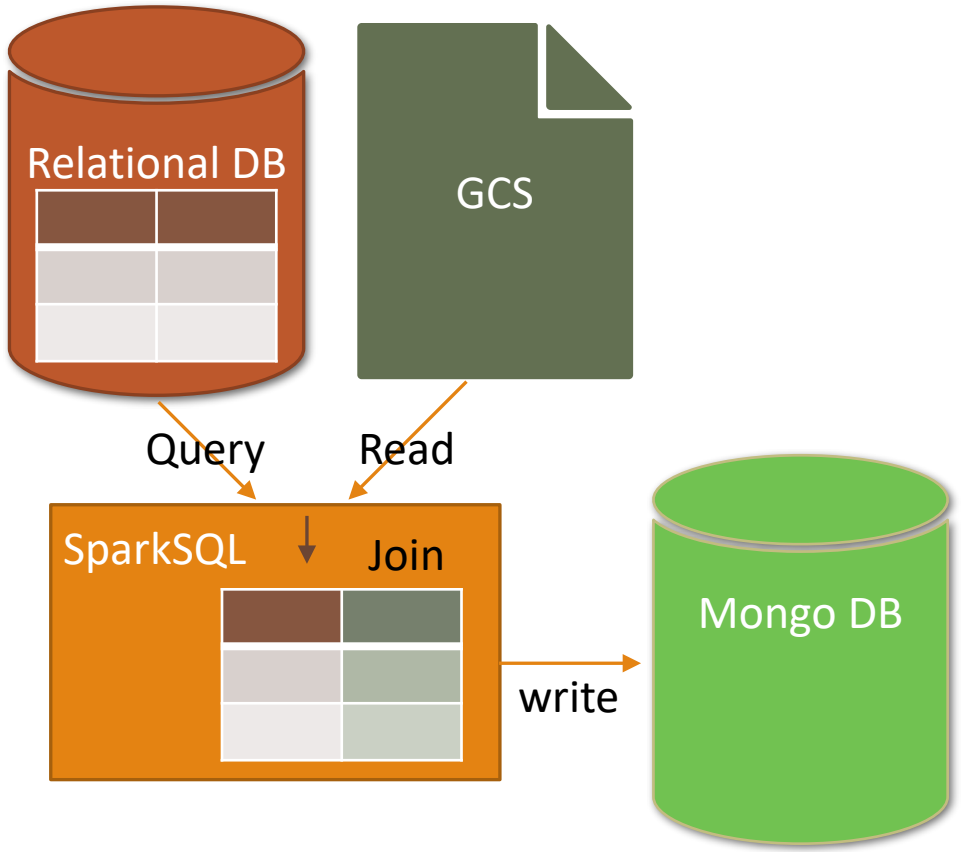
Spark SQL

- **Creating DataFrames**
- DataFrame API Basics

Spark SQL

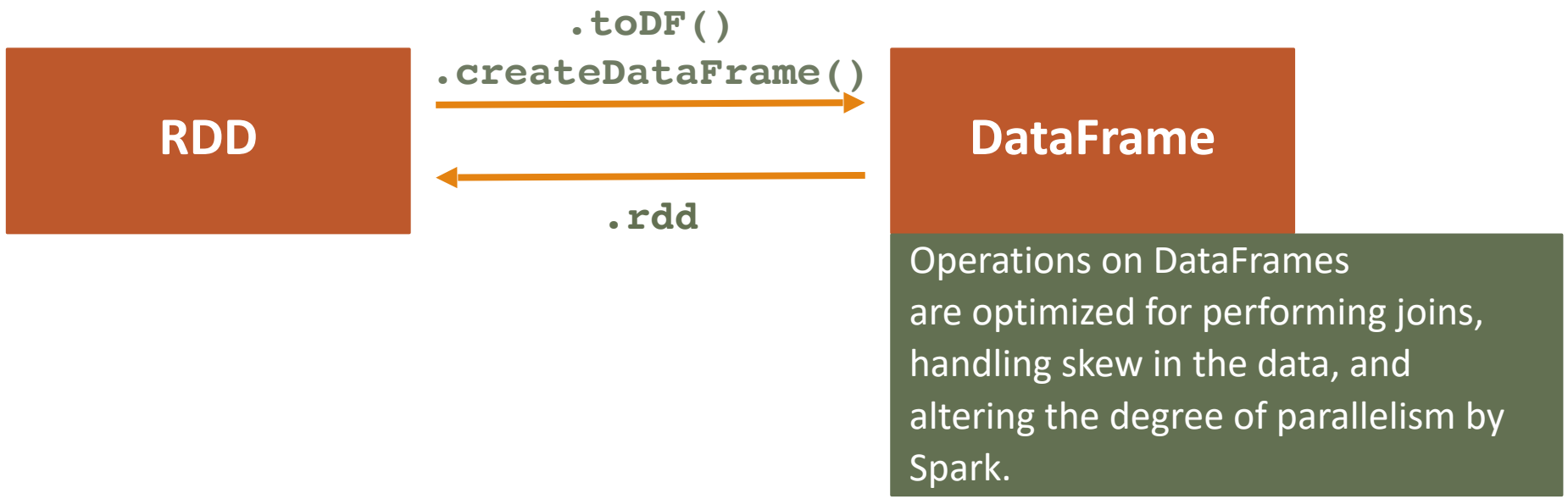
DataFrame

- Handle structured, distributed data with a table-like representation with named column declared with column types.
  - cf. RDD : low-level and direct way of manipulating data in Spark.
- You can join, query and save DataFrames.
- Spark SQL lets you register DataFrames from different sources (SQL, Parquet, JSON, etc.), Pandas dataframe as tables in the table catalog and query them.



https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/Dataset.html

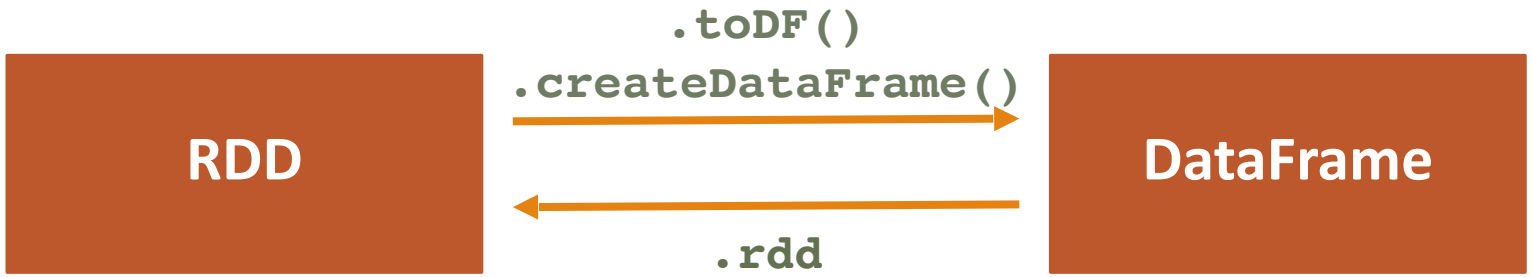
Spark SQL


.toDF()
.createDataFrame()
RDD
DataFrame
.rdd
Operations on DataFrames
are optimized for performing joins,
handling skew in the data, and
altering the degree of parallelism by
Spark.

Spark SQL

Creating DataFrames

- Convert existing RDDs.
  - In order to create DataFrames, you need to load data as text, parse the lines, and identify elements.
  - You can create DataFrames from RDD.
    1) Using RDDs containing row data as tuples using `.toDF()`
       - Limited as it doesn't allow to specify all the schema attributes.
    2) Specifying a schema using `.createDataFrame()`



SparkSQL is a database

True

False

SparkSQL is a database

True

False

SparkSQL is a database

True

False

Spark SQL

Creating DataFrames

- Convert existing RDDs.
  1) Using RDDs containing row data as tuples.
     - Need to add the following lines to initiate SparkSession.
       - Spark Session - The entry point to programming Spark with the DataFrame API.

```
from pyspark.sql import SparkSession
ss = SparkSession.builder.getOrCreate()
```

     - Parse the existing RDD and then call **`.toDF(column_names)`** on the resulting RDD.
  - Limitations
    - All the columns are string type.
    - All the columns are nullable.

➢ You can check the schema using **`.printSchema()`**

➢ You can see the data frame using **`.show(num_rows)`**

Example 1

Given business RDD, convert it to a data frame and see its schema.

If column names are not given, assign column names of 'zip', 'name', 'street', 'city' and 'state'.

Expected Output

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

Example 1

Given business RDD, convert it to a data frame and see its schema.

If column names are not given, assign column names of 'zip', 'name', 'street', 'city' and 'state'.

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

business = sc.textFile("../Data/filtered_registered_business_sf.csv").map(lambda x : x.split(','))
business.first()
```

Example 1

Given business RDD, convert it to a data frame and see its schema.

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()

business = sc.textFile("../Data/filtered_registered_business_sf.csv").map(lambda x : x.split(','))
business.first()
```

```
from pyspark.sql import SparkSession
ss = SparkSession.builder.getOrCreate()
```

```
business_df = business.toDF()
```

```
business_df.show()
```

Example 1

Given business RDD, convert it to a data frame and see its schema.

If column names are not given, assign column names of 'zip', 'name', 'street', 'city' and 'state'.

```
business_df.printSchema()
```

```
root
 |-- _1: string (nullable = true)
 |-- _2: string (nullable = true)
 |-- _3: string (nullable = true)
 |-- _4: string (nullable = true)
 |-- _5: string (nullable = true)
```

Example 1

Given business RDD, convert it to a data frame and see its schema.

If column names are not given, assign column names of 'zip', 'name', 'street', 'city' and 'state'.

```
names = ['zip', 'name', 'street', 'city', 'state']
```

```
business_df = business.toDF(names)
```

```
business_df.show(5)
```

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

Example 1

Given business RDD, convert it to a data frame and see its schema.

If column names are not given, assign column names of 'zip', 'name', 'street', 'city' and 'state'.

```
business_df.show(5)
```

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|94107|Stephens Institut...|    460 Townsend St|San Francisco|   CA|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

```
business_df.printSchema()
```

```
root
 |-- zip: string (nullable = true)
 |-- name: string (nullable = true)
 |-- street: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
```

All the columns are string type.
All the columns are nullable.

Example 1 - Extra

Given business RDD, create a data frame with name and zip code only.

Example 1 - Extra

Given business RDD, create a data frame with name and zip code only.

```
business_df = business.map(lambda x : (x[1], x[0])).toDF(['name','zip'])
```

```
business_df.show(5)
```

```
+--------------------+-----+
|                name|  zip|
+--------------------+-----+
|   Tournahu George L|94123|
|Stephens Institut...|94124|
|Stephens Institut...|94105|
|Stephens Institut...|94108|
|Stephens Institut...|94107|
+--------------------+-----+
only showing top 5 rows
```

```
business_df.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- zip: string (nullable = true)
```

Spark SQL

Creating DataFrames

- Convert existing RDDs.
  2) Specifying a schema using createDataFrame.
     - Use ss.`createDataFrame(data, schema=None, samplingRatio=None , verifySchema=True)`
     - Creates a DataFrame from an RDD.
     - Parameters
       - **data** – an RDD of Row/tuple/list/dict, list, or pandas.DataFrame.
       - **schema** – a StructType or list of column names. default None.
         - When schema is a list of column names, the type of each column will be inferred from data.
       - **samplingRatio** – the sample ratio of rows used for inferring the schema.
       - **verifySchema** – verify data types of every row against schema.

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.SparkSession.createDataFrame

Spark SQL

Creating DataFrames

- Convert existing RDDs.
  2) Specifying a schema using createDataFrame.
     - Use ss.createDataFrame(*data*, *schema=None*, *samplingRatio=None* , *verifySchema=True*)
  - Parameters
    - **schema** – a StructType or list of column names. default None.
      - **StructType** : Representing rows. Consisting of a list of StructField.
      - **StructField** : Representing columns. Including column **name(string), data type, nullable**(default : True), **metadata**(default: None). Need to add `from pyspark.sql.types import *`
        - DataType– NullType(), StringType(), BinaryType(), BooleanType(), DateType(),TimstampType(), DoubleType(), FloatType(), IntegerType(), ArrayType() etc.
      - Ex.

```
from pyspark.sql.types import *
schema = StructType([StructField("name", StringType(), True),
                     StructField("age", IntegerType(), True)])
```

http://spark.apache.org/docs/latest/api/python/_modules/pyspark/sql/types.html
http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#module-pyspark.sql.types

Example 2

1. In SF_business, create a schema for a dataframe for filtered_registered_business_sf.csv, where zip is a longtype, other fields are string types and the ”name” is not null.

2. Create a data frame for filtered_registered_business_sf.csv using the given schema.

Expected Output

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows

root
 |-- zip: long (nullable = true)
 |-- name: string (nullable = false)
 |-- street: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
```

Example 2

1. In SF_business, create a schema for a dataframe for filtered_registered_business_sf.csv, where zip is a longtype, other fields are string types and the ”name” is not null.

2. Create a data frame for filtered_registered_business_sf.csv using the given schema.

```
from pyspark.sql.types import *
schema = StructType([ StructField("zip", LongType(), True),
                      StructField("name", StringType(), False),
                      StructField("street", StringType(), True),
                      StructField("city", StringType(), True),
                      StructField("state", StringType(), True)
                    ])
```

Example 2

1. In SF_business, create a schema for a dataframe for filtered_registered_business_sf.csv, where zip is a longtype, other fields are string types and the ”name” is not null.

2. Create a data frame for filtered_registered_business_sf.csv using the given schema.

```
def IntegerSafe(value): # In case there are non-integer type to be converted.
    try:
        return int(value)
    except ValueError:
        return None

business = business.map(lambda x : (IntegerSafe(x[0]), x[1], x[2], x[3], x[4]))
business_df = ss.createDataFrame(business, schema)
business_df.show(5)
business_df.printSchema()
```

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows

root
 |-- zip: long (nullable = true)
 |-- name: string (nullable = false)
 |-- street: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
```

Contents

Spark SQL

- Creating DataFrames
- **DataFrame API Basics**

DataFrame API Basics (Example 3)

DataFrame

- Basic APIs

Operation	Details	Example
`select(column_names)`	Returns a new DataFrame with given cols.	`df.select('name', 'age')`
`drop(column_names)`	Returns a new DataFrame without the specified columns.	`df.drop('age')`
`filter(constraints), where(constraints)`	Filters rows using the given condition.	`df.filter(df.age > 3)`
`withColumnRenamed(existing_col_name, new_col_name)`	Returns a new DataFrame by renaming an existing column.	`df.withColumnRenamed('age', 'age2')`
`withColumn(columnName, columnExpression)`	Returns a new DataFrame by adding a column or replacing the existing column that has the same name.	`df.withColumn('age2', df.age + 2)`
`orderBy(cols), sort(cols)`	Returns a new DataFrame sorted by the specified column(s)	`df.orderBy(["age", "name"], ascending=[0, 1])`

- And many more : https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

DataFrame API Basics

DataFrame

- Basic APIs
  - `.select(<column_names> or list of dataframe[<column_name>])`
    - Work with Column objects.

Example 3

Choose five rows including name and zip from the business_df data frame.

Expected Output

```
+--------------------+-----+
|                name|  zip|
+--------------------+-----+
|   Tournahu George L|94123|
|Stephens Institut...|94124|
|Stephens Institut...|94105|
|Stephens Institut...|94108|
|Stephens Institut...|94107|
+--------------------+-----+
only showing top 5 rows
```

Example 3

Choose five rows including name and zip from the business_df data frame.

```
business_df.select("name", "zip").show(5)
```

```
+--------------------+-----+
|                name|  zip|
+--------------------+-----+
|   Tournahu George L|94123|
|Stephens Institut...|94124|
|Stephens Institut...|94105|
|Stephens Institut...|94108|
|Stephens Institut...|94107|
+--------------------+-----+
only showing top 5 rows
```

Example 3

Choose five rows including name and zip from the business_df data frame.

```
business_df.select(business_df["name"], business_df["zip"]).show(5)
```

```
+--------------------+-----+
|                name|  zip|
+--------------------+-----+
|   Tournahu George L|94123|
|Stephens Institut...|94124|
|Stephens Institut...|94105|
|Stephens Institut...|94108|
|Stephens Institut...|94107|
+--------------------+-----+
only showing top 5 rows
```

DataFrame API Basics

DataFrame

- Basic APIs
  - `.drop(<column_names>)`
    - Select all except the given columns.

Example 3

Select 5 rows with all the columns except for state and city

Expected Output

```
+-----+--------------------+--------------------+
|  zip|                name|              street|
+-----+--------------------+--------------------+
|94123|   Tournahu George L|   3301 Broderick St|
|94124|Stephens Institut...|    2225 Jerrold Ave|
|94105|Stephens Institut...|180 New Montgomer...|
|94108|Stephens Institut...|       540 Powell St|
|94107|Stephens Institut...|    460 Townsend St|
+-----+--------------------+--------------------+
only showing top 5 rows
```

Example 3

Select 5 rows with all the columns except for state and city

```
business_df.drop("state","city").show(5)
```

```
+-----+--------------------+--------------------+
|  zip|                name|              street|
+-----+--------------------+--------------------+
|94123|   Tournahu George L|   3301 Broderick St|
|94124|Stephens Institut...|    2225 Jerrold Ave|
|94105|Stephens Institut...|180 New Montgomer...|
|94108|Stephens Institut...|       540 Powell St|
|94107|Stephens Institut...|    460 Townsend St|
+-----+--------------------+--------------------+
only showing top 5 rows
```

In SparkSQL, .drop() is for dropping a database table.

True

False

In SparkSQL, .drop() is for dropping a database table.

True

False

In SparkSQL, .drop() is for dropping a database table.

True

False

DataFrame API Basics

DataFrame

- Basic APIs
  - `.filter("<constraints>"),.where("<constraints>")`
    - Apply constraints.

```
business_df.filter("zip == 94123").show(5)
```

```
+-----+--------------------+-----------------+-------------+-----+
|  zip|                name|           street|         city|state|
+-----+--------------------+-----------------+-------------+-----+
|94123|   Tournahu George L|3301 Broderick St|San Francisco|   CA|
|94123|         Amore Robert|    1958 Union St|San Francisco|   CA|
|94123|Aunt Anns Corp He...|    2722 Gough St|San Francisco|   CA|
|94123|Barbagelata & Co Inc| 2381 Chestnut St|San Francisco|   CA|
|94123|Boas Internationa...|  2098 Lombard St|San Francisco|   CA|
+-----+--------------------+-----------------+-------------+-----+
only showing top 5 rows
```

Example 3

Filter 10 rows where city is San Francisco but state is not CA.

Expected Output

zip	name	street	city	state
94134	Kugay Faruk	301 Executive Par...	San Francisco	
94114	Crystal Way Inc	2335 Market St	San Francisco	
94109	Protopopov	1190 Pine St	San Francisco	CO
94134	301 Rolph Street ...	65 Leland Ave	San Francisco	
94127	Nu Greek Wine Co Inc	225 Moncada Way	San Francisco	SC
94124	Calvary Hill Soci...	141 Industrial St	San Francisco	
94118	Leung Clifton	320 6th Ave	San Francisco	
94109	Chun Jimmy & Shuk Y	1529 Sacramento St	San Francisco	
94109	Chun Jimmy & Shuk Y	1354 Sacramento St	San Francisco	
94118	Leung Olivia	320 6th Ave	San Francisco	

only showing top 10 rows

Example 3

Filter 10 rows where city is San Francisco but state is not CA.

```
business_df.where("city == 'San Francisco' and state != 'CA'").show(10)
```

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94134|         Kugay Faruk|301 Executive Par...|San Francisco|     |
|94114|     Crystal Way Inc|      2335 Market St|San Francisco|     |
|94109|           Protopopov|        1190 Pine St|San Francisco|   CO|
|94134|301 Rolph Street ...|       65 Leland Ave|San Francisco|     |
|94127|Nu Greek Wine Co Inc|     225 Moncada Way|San Francisco|   SC|
|94124|Calvary Hill Soci...|   141 Industrial St|San Francisco|     |
|94118|       Leung Clifton|         320 6th Ave|San Francisco|     |
|94109| Chun Jimmy & Shuk Y|  1529 Sacramento St|San Francisco|     |
|94109| Chun Jimmy & Shuk Y|  1354 Sacramento St|San Francisco|     |
|94118|        Leung Olivia|         320 6th Ave|San Francisco|     |
+-----+--------------------+--------------------+-------------+-----+
only showing top 10 rows
```

DataFrame API Basics

DataFrame

- Basic APIs
  - `.withColumnRenamed(existing_col_name, new_col_name)`
    - Rename columns
  - `.withColumn(columnName, columnExpression)`
    - Add columns.

Example 3

Change a column name to zip code from zip.

Expected Output

```
+--------+--------------------+--------------------+-------------+-----+
|zip code|                name|              street|         city|state|
+--------+--------------------+--------------------+-------------+-----+
|   94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|   94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|   94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|   94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|   94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|
+--------+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

Example 3

Change a column name to zip code from zip.

```
business_df.withColumnRenamed('zip','zip code').show(5)
```

```
+--------+--------------------+--------------------+-------------+-----+
|zip code|                name|              street|         city|state|
+--------+--------------------+--------------------+-------------+-----+
|   94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|
|   94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|
|   94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|
|   94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|
|   94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|
+--------+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

Example 3

Add the “odd_zip” column and its value (True, False) to see whether zip is an odd number.

Expected Output

```
+-----+--------------------+--------------------+-------------+-----+-------+
|  zip|                name|              street|         city|state|odd_zip|
+-----+--------------------+--------------------+-------------+-----+-------+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|   true|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|  false|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|   true|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|  false|
|94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|   true|
+-----+--------------------+--------------------+-------------+-----+-------+
only showing top 5 rows
```

Example 3

Add the “odd_zip” column and its value (True, False) to see whether zip is an odd number.

```
business_df.withColumn('odd_zip', business_df['zip'] % 2 ).show(5)
```

```
+-----+--------------------+--------------------+-------------+-----+-------+
|  zip|                name|              street|         city|state|odd_zip|
+-----+--------------------+--------------------+-------------+-----+-------+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|      1|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|      0|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|      1|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|      0|
|94107|Stephens Institut...|    460 Townsend St|San Francisco|   CA|      1|
+-----+--------------------+--------------------+-------------+-----+-------+
only showing top 5 rows
```

Example 3

Add the “odd_zip” column and its value (True, False) to see whether zip is an odd number.

```
business_df.withColumn('odd_zip', business_df['zip'] % 2 == True ).show(5)
```

```
+-----+--------------------+--------------------+-------------+-----+-------+
|  zip|                name|              street|         city|state|odd_zip|
+-----+--------------------+--------------------+-------------+-----+-------+
|94123|   Tournahu George L|   3301 Broderick St|San Francisco|   CA|   true|
|94124|Stephens Institut...|    2225 Jerrold Ave|San Francisco|   CA|  false|
|94105|Stephens Institut...|180 New Montgomer...|San Francisco|   CA|   true|
|94108|Stephens Institut...|       540 Powell St|San Francisco|   CA|  false|
|94107|Stephens Institut...|     460 Townsend St|San Francisco|   CA|   true|
+-----+--------------------+--------------------+-------------+-----+-------+
only showing top 5 rows
```

DataFrame API Basics

DataFrame

- Basic APIs
  - `.orderBy(columns, ascending = True), .sort(columns, ascending = True)`
    - Sort data.

Example 3

Order business_df by name in ascending/descending order.

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94521|     Zzr Enterprises|5520 Pennsylvania...|      Concord|   CA|
|94121|Zzgor Entertainme...|     447 22nd Ave #3|San+francisco|   CA|
|94104|          Zyzzyva Inc|      57 Post St 604|San Francisco|   CA|
|94118|          Zyzzyva Inc|       44 Almaden Ct|San Francisco|   CA|
|53226|           Zywave Inc|10700 W Research ...|    Milwaukee|   WI|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

Example 3

Order business_df by name in ascending/descending order.

```
business_df.sort("name").show(5)
```

```
+-----+----------------+------+-------------------+----------------+
|  zip|            name|street|               city|           state|
+-----+----------------+------+-------------------+----------------+
|94103|             "1"|     2|3 Express Moving ...|31 Duboce Ave 31a|
|94105|      "1-2-3 Deli| Inc."|      123 Mission St|   San Francisco|
|94105|"1055 Pine Street|  Llc"|79 New Montgomery St|   San Francisco|
|94105|"1069 Pine Street|  Llc"|79 New Montgomery St|   San Francisco|
|94109|"1080 Bush Street|  Llc"|       1080 Bush St|   San Francisco|
+-----+----------------+------+-------------------+----------------+
only showing top 5 rows
```

Example 3

Order business_df by name in ascending/descending order.

```
business_df.orderBy("name", ascending = False).show(5)
```

```
+-----+--------------------+--------------------+-------------+-----+
|  zip|                name|              street|         city|state|
+-----+--------------------+--------------------+-------------+-----+
|94521|     Zzr Enterprises|5520 Pennsylvania...|      Concord|   CA|
|94121|Zzgor Entertainme...|     447 22nd Ave #3|San+francisco|   CA|
|94104|          Zyzzyva Inc|      57 Post St 604|San Francisco|   CA|
|94118|          Zyzzyva Inc|       44 Almaden Ct|San Francisco|   CA|
|53226|           Zywave Inc|10700 W Research ...|    Milwaukee|   WI|
+-----+--------------------+--------------------+-------------+-----+
only showing top 5 rows
```

Example 4

Create a dataframe using supervisor_sf.csv and where zip and id are both non-nullable and integertypes.

Find supervisor ids for zipcode, 94123.

Expected Output

```
+---+
| id|
+---+
|  2|
+---+
```

```
supervisor_df.printSchema()
```

```
root
 |-- zip: integer (nullable = false)
 |-- id: integer (nullable = false)
```

Example 4

Create a dataframe using supervisor_sf.csv and where zip and id are both non-nullable and integertypes.

Find supervisor ids for zipcode, 94123.

```
supervisor_schema = StructType([ StructField("zip", IntegerType(), False),
                    StructField("id", IntegerType(), False)
                    ])
```

```
supervisor_df = ss.createDataFrame(supervisor, supervisor_schema)
```

```
supervisor_df.filter("zip == 94123").select("id").show()
```

```
+---+
| id|
+---+
|  2|
+---+
```

```
supervisor_df.printSchema()
```

```
root
 |-- zip: integer (nullable = false)
 |-- id: integer (nullable = false)
```

DataFrame API Basics (Example 3)

DataFrame

- Basic APIs

Operation	Details	Example
`select(column_names)`	Returns a new DataFrame with given cols.	`df.select('name', 'age')`
`drop(column_names)`	Returns a new DataFrame without the specified columns.	`df.drop('age')`
`filter(constraints), where(constraints)`	Filters rows using the given condition.	`df.filter(df.age > 3)`
`withColumnRenamed(existing_col_name, new_col_name)`	Returns a new DataFrame by renaming an existing column.	`df.withColumnRenamed('age', 'age2')`
`withColumn(columnName, columnExpression)`	Returns a new DataFrame by adding a column or replacing the existing column that has the same name.	`df.withColumn('age2', df.age + 2)`
`orderBy(cols), sort(cols)`	Returns a new DataFrame sorted by the specified column(s)	`df.orderBy(["age", "name"], ascending=[0, 1])`

- And many more : https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Contents

Spark SQL

- Creating DataFrames
- DataFrame API Basics

Reference

Spark Online Documentation : http://spark.apache.org/docs/latest/

Karau, Holden, et al. *Learning spark: lightning-fast big data analysis*. O'Reilly Media, Inc., 2015.

Zecevic, Petar, et al. Spark in Action, Manning, 2016.