In [1]:
```
import findspark
findspark.init()
```

In [16]:
```
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

print(spark)
rdd=spark.sparkContext.parallelize([1,2,3,4,56])
print("RDD count :"+str(rdd.count()))

rdd = spark.sparkContext.emptyRDD
print(rdd)
rdd2 = spark.sparkContext.parallelize([])
print(rdd2)
```

```
<pyspark.sql.session.SparkSession object at 0x000001E9F74C29D0>
RDD count :5
<bound method SparkContext.emptyRDD of <SparkContext master=local[*] appName=Sp
arkByExamples.com>>
ParallelCollectionRDD[325] at readRDDFromFile at PythonRDD.scala:274
```

In [17]:
```
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
rdd=spark.sparkContext.parallelize([1,2,3,4,5])


rddCollect = rdd.collect()
print("Number of Partitions: "+str(rdd.getNumPartitions()))
print("Action: First element: "+str(rdd.first()))
print(rddCollect)

emptyRDD = spark.sparkContext.emptyRDD()
emptyRDD2 = rdd=spark.sparkContext.parallelize([])

print(""+str(emptyRDD2.isEmpty()))
```

```
Number of Partitions: 8
Action: First element: 1
[1, 2, 3, 4, 5]
True
```

In [2]:
```python
import pyspark
from pyspark.sql import SparkSession


spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

data = [("James","","Smith","36636","M",60000),
        ("Michael","Rose","","40288","M",70000),
        ("Robert","","Williams","42114","",400000),
        ("Maria","Anne","Jones","39192","F",500000),
        ("Jen","Mary","Brown","","F",0)]

columns = ["first_name","middle_name","last_name","dob","gender","salary"]
pysparkDF = spark.createDataFrame(data = data, schema = columns)
pysparkDF.printSchema()
pysparkDF.show(truncate=False)

pandasDF = pysparkDF.toPandas()
print(pandasDF)

# Nested structure elements
from pyspark.sql.types import StructType, StructField, StringType,IntegerType
dataStruct = [(("James","","Smith"),"36636","M","3000"), \
      (("Michael","Rose",""),"40288","M","4000"), \
      (("Robert","","Williams"),"42114","M","4000"), \
      (("Maria","Anne","Jones"),"39192","F","4000"), \
      (("Jen","Mary","Brown"),"","F","-1") \
]

schemaStruct = StructType([
        StructField('name', StructType([
             StructField('firstname', StringType(), True),
             StructField('middlename', StringType(), True),
             StructField('lastname', StringType(), True)
             ])),
          StructField('dob', StringType(), True),
         StructField('gender', StringType(), True),
         StructField('salary', StringType(), True)
         ])

df = spark.createDataFrame(data=dataStruct, schema = schemaStruct)
df.printSchema()
df.show(truncate=False)

pandasDF2 = df.toPandas()
print(pandasDF2)
```

```
root
 |-- first_name: string (nullable = true)
 |-- middle_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- dob: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+----------+-----------+---------+-----+------+------+
|first_name|middle_name|last_name|dob  |gender|salary|
+----------+-----------+---------+-----+------+------+
|James     |           |Smith    |36636|M     |60000 |
|Michael   |Rose       |         |40288|M     |70000 |
|Robert    |           |Williams |42114|      |400000|
|Maria     |Anne       |Jones    |39192|F     |500000|
|Jen       |Mary       |Brown    |     |F     |0     |
+----------+-----------+---------+-----+------+------+

  first_name middle_name last_name    dob gender  salary
0      James                 Smith  36636      M   60000
1    Michael        Rose            40288      M   70000
2     Robert              Williams  42114         400000
3      Maria        Anne     Jones  39192      F  500000
4        Jen        Mary     Brown             F       0
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- dob: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: string (nullable = true)

+--------------------+-----+------+------+
|name                |dob  |gender|salary|
+--------------------+-----+------+------+
|{James, , Smith}    |36636|M     |3000  |
|{Michael, Rose, }   |40288|M     |4000  |
|{Robert, , Williams}|42114|M     |4000  |
|{Maria, Anne, Jones}|39192|F     |4000  |
|{Jen, Mary, Brown}  |     |F     |-1    |
+--------------------+-----+------+------+

                   name    dob gender salary
0      (James, , Smith)  36636      M   3000
1     (Michael, Rose, )  40288      M   4000
2  (Robert, , Williams)  42114      M   4000
3  (Maria, Anne, Jones)  39192      F   4000
4     (Jen, Mary, Brown)             F     -1
```

In [3]:
```
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

data = ["Project Gutenberg’s",
        "Alice’s Adventures in Wonderland",
        "Project Gutenberg’s",
        "Adventures in Wonderland",
        "Project Gutenberg’s"]
rdd=spark.sparkContext.parallelize(data)

for element in rdd.collect():
    print(element)

#Flatmap
rdd2=rdd.flatMap(lambda x: x.split(" "))
for element in rdd2.collect():
    print(element)
```

```
Project Gutenberg’s
Alice’s Adventures in Wonderland
Project Gutenberg’s
Adventures in Wonderland
Project Gutenberg’s
Project
Gutenberg’s
Alice’s
Adventures
in
Wonderland
Project
Gutenberg’s
Adventures
in
Wonderland
Project
Gutenberg’s
```

In [4]:
```
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()


dept = [("Finance",10),
        ("Marketing",20),
        ("Sales",30),
        ("IT",40)
      ]
rdd = spark.sparkContext.parallelize(dept)


df = rdd.toDF()
df.printSchema()
df.show(truncate=False)


deptColumns = ["dept_name","dept_id"]
df2 = rdd.toDF(deptColumns)
df2.printSchema()
df2.show(truncate=False)


deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()
deptDF.show(truncate=False)




from pyspark.sql.types import StructType,StructField, StringType
deptSchema = StructType([
    StructField('dept_name', StringType(), True),
    StructField('dept_id', StringType(), True)
])


deptDF1 = spark.createDataFrame(data=dept, schema = deptSchema)
deptDF1.printSchema()
deptDF1.show(truncate=False)
```

```
root
 |-- _1: string (nullable = true)
 |-- _2: long (nullable = true)

+---------+---+
|_1       |_2 |
+---------+---+
|Finance  |10 |
|Marketing|20 |
|Sales    |30 |
|IT       |40 |
+---------+---+

root
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)
```

```
+---------+-------+
|dept_name|dept_id|
+---------+-------+
|Finance  |10     |
|Marketing|20     |
|Sales    |30     |
|IT       |40     |
+---------+-------+

root
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)

+---------+-------+
|dept_name|dept_id|
+---------+-------+
|Finance  |10     |
|Marketing|20     |
|Sales    |30     |
|IT       |40     |
+---------+-------+

root
 |-- dept_name: string (nullable = true)
 |-- dept_id: string (nullable = true)

+---------+-------+
|dept_name|dept_id|
+---------+-------+
|Finance  |10     |
|Marketing|20     |
|Sales    |30     |
|IT       |40     |
+---------+-------+
```

In [14]:
```
spark: SparkSession = SparkSession.builder \
    .master("local[1]") \
    .appName("SparkByExamples.com") \
    .getOrCreate()


filePath="C:\\Users\\TanujaNekkanti\\Downloads\\groceries.csv"
df = spark.read.options(header='True', inferSchema='True') \
          .csv(filePath)


df.printSchema()
df.show(truncate=False)


df.na.drop().show(truncate=False)


df.na.drop(how="any").show(truncate=False)


df.na.drop(subset=["Bananas","o1"]) \
   .show(truncate=False)
```

```
root
 |-- o1: string (nullable = true)
 |-- Seattle: string (nullable = true)
 |-- Bananas: string (nullable = true)
 |-- 01-01-2017: string (nullable = true)
 |-- 7: integer (nullable = true)

+---+---------+--------+----------+---+
|o1 |Seattle  |Bananas |01-01-2017|7  |
+---+---------+--------+----------+---+
|o2 |Kent     |Apples  |02-01-2017|20 |
|o3 |Bellevue |Flowers |02-01-2017|10 |
|o4 |Redmond  |Meat    |03-01-2017|40 |
|o5 |Seattle  |Potatoes|04-01-2017|9  |
|o6 |Bellevue |Bread   |04-01-2017|5  |
|o7 |Redmond  |Bread   |05-01-2017|5  |
|o8 |Issaquah |Onion   |05-01-2017|4  |
|o9 |Redmond  |Cheese  |05-01-2017|15 |
|o10|Issaquah |Onion   |06-01-2017|4  |
|o11|Renton   |Bread   |05-01-2017|5  |
|o12|Issaquah |Onion   |07-01-2017|4  |
|o13|Sammamish|Bread   |07-01-2017|5  |
|o14|Issaquah |Tomato  |07-01-2017|6  |
+---+---------+--------+----------+---+

+---+---------+--------+----------+---+
|o1 |Seattle  |Bananas |01-01-2017|7  |
+---+---------+--------+----------+---+
|o2 |Kent     |Apples  |02-01-2017|20 |
|o3 |Bellevue |Flowers |02-01-2017|10 |
|o4 |Redmond  |Meat    |03-01-2017|40 |
|o5 |Seattle  |Potatoes|04-01-2017|9  |
|o6 |Bellevue |Bread   |04-01-2017|5  |
|o7 |Redmond  |Bread   |05-01-2017|5  |
|o8 |Issaquah |Onion   |05-01-2017|4  |
|o9 |Redmond  |Cheese  |05-01-2017|15 |
|o10|Issaquah |Onion   |06-01-2017|4  |
|o11|Renton   |Bread   |05-01-2017|5  |
|o12|Issaquah |Onion   |07-01-2017|4  |
|o13|Sammamish|Bread   |07-01-2017|5  |
|o14|Issaquah |Tomato  |07-01-2017|6  |
+---+---------+--------+----------+---+

+---+---------+--------+----------+---+
|o1 |Seattle  |Bananas |01-01-2017|7  |
+---+---------+--------+----------+---+
|o2 |Kent     |Apples  |02-01-2017|20 |
|o3 |Bellevue |Flowers |02-01-2017|10 |
|o4 |Redmond  |Meat    |03-01-2017|40 |
|o5 |Seattle  |Potatoes|04-01-2017|9  |
|o6 |Bellevue |Bread   |04-01-2017|5  |
|o7 |Redmond  |Bread   |05-01-2017|5  |
|o8 |Issaquah |Onion   |05-01-2017|4  |
|o9 |Redmond  |Cheese  |05-01-2017|15 |
|o10|Issaquah |Onion   |06-01-2017|4  |
|o11|Renton   |Bread   |05-01-2017|5  |
|o12|Issaquah |Onion   |07-01-2017|4  |
|o13|Sammamish|Bread   |07-01-2017|5  |
|o14|Issaquah |Tomato  |07-01-2017|6  |
+---+---------+--------+----------+---+

+---+---------+--------+----------+---+
|o1 |Seattle  |Bananas |01-01-2017|7  |
+---+---------+--------+----------+---+
|o2 |Kent     |Apples  |02-01-2017|20 |
|o3 |Bellevue |Flowers |02-01-2017|10 |
|o4 |Redmond  |Meat    |03-01-2017|40 |
|o5 |Seattle  |Potatoes|04-01-2017|9  |
|o6 |Bellevue |Bread   |04-01-2017|5  |
|o7 |Redmond  |Bread   |05-01-2017|5  |
|o8 |Issaquah |Onion   |05-01-2017|4  |
|o9 |Redmond  |Cheese  |05-01-2017|15 |
|o10|Issaquah |Onion   |06-01-2017|4  |
|o11|Renton   |Bread   |05-01-2017|5  |
|o12|Issaquah |Onion   |07-01-2017|4  |
|o13|Sammamish|Bread   |07-01-2017|5  |
|o14|Issaquah |Tomato  |07-01-2017|6  |
+---+---------+--------+----------+---+
```

In [15]:
```
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()


data = [("James","Smith","USA","CA"),
    ("Michael","Rose","USA","NY"),
    ("Robert","Williams","USA","CA"),
    ("Maria","Jones","USA","FL")
  ]

columns = ["firstname","lastname","country","state"]
df = spark.createDataFrame(data = data, schema = columns)
df.show(truncate=False)

df.select("firstname","lastname").show()

#Using Dataframe object name
df.select(df.firstname,df.lastname).show()
df.select(df["firstname"],df["lastname"]).show()

# Using col function
from pyspark.sql.functions import col
df.select(col("firstname").alias("fname"),col("lastname")).show()

# Show all columns
df.select("*").show()
df.select([col for col in df.columns]).show()
df.select(*columns).show()

df.select(df.columns[:3]).show(3)
df.select(df.columns[2:4]).show(3)

df.select(df.colRegex("`^.*name*`")).show()

data = [
        (("James",None,"Smith"),"OH","M"),
        (("Anna","Rose",""),"NY","F"),
        (("Julia","","Williams"),"OH","F"),
        (("Maria","Anne","Jones"),"NY","M"),
        (("Jen","Mary","Brown"),"NY","M"),
        (("Mike","Mary","Williams"),"OH","M")
        ]

from pyspark.sql.types import StructType,StructField, StringType
schema = StructType([
    StructField('name', StructType([
         StructField('firstname', StringType(), True),
         StructField('middlename', StringType(), True),
         StructField('lastname', StringType(), True)
         ])),
     StructField('state', StringType(), True),
     StructField('gender', StringType(), True)
     ])

df2 = spark.createDataFrame(data = data, schema = schema)
df2.printSchema()
df2.show(truncate=False) # shows all columns
```

```
df2.select("name").show(truncate=False)
df2.select("name.firstname","name.lastname").show(truncate=False)
df2.select("name.*").show(truncate=False)
```

```
+---------+--------+-------+-----+
|firstname|lastname|country|state|
+---------+--------+-------+-----+
|James    |Smith   |USA    |CA   |
|Michael  |Rose    |USA    |NY   |
|Robert   |Williams|USA    |CA   |
|Maria    |Jones   |USA    |FL   |
+---------+--------+-------+-----+

+---------+--------+
|firstname|lastname|
+---------+--------+
|    James|   Smith|
|  Michael|    Rose|
|   Robert|Williams|
|    Maria|   Jones|
+---------+--------+

+---------+--------+
```

In [18]:
```
spark = SparkSession.builder.master("local[1]") \
                    .appName('SparkByExamples.com') \
                    .getOrCreate()

data = [("James","Smith","USA","CA"),("Michael","Rose","USA","NY"), \
    ("Robert","Williams","USA","CA"),("Maria","Jones","USA","FL") \
  ]
columns=["firstname","lastname","country","state"]
df=spark.createDataFrame(data=data,schema=columns)
df.show()
print(df.collect())

states1=df.rdd.map(lambda x: x[3]).collect()
print(states1)
#['CA', 'NY', 'CA', 'FL']
from collections import OrderedDict
res = list(OrderedDict.fromkeys(states1))
print(res)
#['CA', 'NY', 'FL']

#Example 2
states2=df.rdd.map(lambda x: x.state).collect()
print(states2)
#['CA', 'NY', 'CA', 'FL']

states3=df.select(df.state).collect()
print(states3)
#[Row(state='CA'), Row(state='NY'), Row(state='CA'), Row(state='FL')]

states4=df.select(df.state).rdd.flatMap(lambda x: x).collect()
print(states4)
#['CA', 'NY', 'CA', 'FL']

states5=df.select(df.state).toPandas()['state']
states6=list(states5)
print(states6)
#['CA', 'NY', 'CA', 'FL']

pandDF=df.select(df.state,df.firstname).toPandas()
print(list(pandDF['state']))
print(list(pandDF['firstname']))
```

```
+---------+--------+-------+-----+
|firstname|lastname|country|state|
+---------+--------+-------+-----+
|    James|   Smith|    USA|   CA|
|  Michael|    Rose|    USA|   NY|
|   Robert|Williams|    USA|   CA|
|    Maria|   Jones|    USA|   FL|
+---------+--------+-------+-----+

[Row(firstname='James', lastname='Smith', country='USA', state='CA'), Row(first
name='Michael', lastname='Rose', country='USA', state='NY'), Row(firstname='Rob
ert', lastname='Williams', country='USA', state='CA'), Row(firstname='Maria', l
astname='Jones', country='USA', state='FL')]
['CA', 'NY', 'CA', 'FL']
```

```
['CA', 'NY', 'FL']
['CA', 'NY', 'CA', 'FL']
[Row(state='CA'), Row(state='NY'), Row(state='CA'), Row(state='FL')]
['CA', 'NY', 'CA', 'FL']
['CA', 'NY', 'CA', 'FL']
['CA', 'NY', 'CA', 'FL']
['James', 'Michael', 'Robert', 'Maria']
```

In [19]:
```
spark = SparkSession.builder.master("local[1]") \
                    .appName('SparkByExamples.com') \
                    .getOrCreate()

columns = ["name","languagesAtSchool","currentState"]
data = [("James,,Smith",["Java","Scala","C++"],"CA"), \
    ("Michael,Rose,",["Spark","Java","C++"],"NJ"), \
    ("Robert,,Williams",["CSharp","VB"],"NV")]

df = spark.createDataFrame(data=data,schema=columns)
df.printSchema()
df.show(truncate=False)

from pyspark.sql.functions import col, concat_ws
df2 = df.withColumn("languagesAtSchool",
   concat_ws(",",col("languagesAtSchool")))
df2.printSchema()
df2.show(truncate=False)


df.createOrReplaceTempView("ARRAY_STRING")
spark.sql("select name, concat_ws(',',languagesAtSchool) as languagesAtSchool," +
    " currentState from ARRAY_STRING") \
    .show(truncate=False)
```

```
root
 |-- name: string (nullable = true)
 |-- languagesAtSchool: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- currentState: string (nullable = true)

+----------------+------------------+------------+
|name            |languagesAtSchool |currentState|
+----------------+------------------+------------+
|James,,Smith    |[Java, Scala, C++]|CA          |
|Michael,Rose,   |[Spark, Java, C++]|NJ          |
|Robert,,Williams|[CSharp, VB]      |NV          |
+----------------+------------------+------------+

root
 |-- name: string (nullable = true)
 |-- languagesAtSchool: string (nullable = false)
 |-- currentState: string (nullable = true)

+----------------+-----------------+------------+
|name            |languagesAtSchool|currentState|
+----------------+-----------------+------------+
|James,,Smith    |Java,Scala,C++   |CA          |
|Michael,Rose,   |Spark,Java,C++   |NJ          |
|Robert,,Williams|CSharp,VB        |NV          |
+----------------+-----------------+------------+

+----------------+-----------------+------------+
|name            |languagesAtSchool|currentState|
+----------------+-----------------+------------+
|James,,Smith    |Java,Scala,C++   |CA          |
|Michael,Rose,   |Spark,Java,C++   |NJ          |
|Robert,,Williams|CSharp,VB        |NV          |
+----------------+-----------------+------------+
```

In [20]:
```
spark = SparkSession.builder \
               .appName('SparkByExamples.com') \
               .getOrCreate()
data=[["1"],["2"]]
df=spark.createDataFrame(data,["id"])

from pyspark.sql.functions import *

#current_date() & current_timestamp()
df.withColumn("current_date",current_date()) \
  .withColumn("current_timestamp",current_timestamp()) \
  .show(truncate=False)

#SQL
spark.sql("select current_date(), current_timestamp()") \
     .show(truncate=False)

# Date & Timestamp into custom format
df.withColumn("date_format",date_format(current_date(),"MM-dd-yyyy")) \
  .withColumn("to_timestamp",to_timestamp(current_timestamp(),"MM-dd-yyyy HH mm s
  .show(truncate=False)

#SQL
spark.sql("select date_format(current_date(),'MM-dd-yyyy') as date_format ," + \
          "to_timestamp(current_timestamp(),'MM-dd-yyyy HH mm ss SSS') as to_time
     .show(truncate=False)
```

```
+---+------------+-----------------------+
|id |current_date|current_timestamp      |
+---+------------+-----------------------+
|1  |2022-02-07  |2022-02-07 17:02:55.431|
|2  |2022-02-07  |2022-02-07 17:02:55.431|
+---+------------+-----------------------+

+--------------+-----------------------+
|current_date()|current_timestamp()    |
+--------------+-----------------------+
|2022-02-07    |2022-02-07 17:03:00.942|
+--------------+-----------------------+

+---+-----------+-----------------------+
|id |date_format|to_timestamp           |
+---+-----------+-----------------------+
|1  |02-07-2022 |2022-02-07 17:03:00.996|
|2  |02-07-2022 |2022-02-07 17:03:00.996|
+---+-----------+-----------------------+

+-----------+-----------------------+
|date_format|to_timestamp           |
+-----------+-----------------------+
|02-07-2022 |2022-02-07 17:03:06.506|
+-----------+-----------------------+
In [ ]:
```