databricks Databricks Practice Noteboook

(https://databricks.com)

```
columns = ["language","users_count"]
data = [("Java", "20000"), ("Python", "100000"), ("Scala", "3000")]
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
rdd = spark.sparkContext.parallelize(data)
```

```
dfFromRDD1 = rdd.toDF()
dfFromRDD1.printSchema()
```

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

```
columns = ["language","users_count"]
dfFromRDD1 = rdd.toDF(columns)
dfFromRDD1.printSchema()
```

```
root
 |-- language: string (nullable = true)
 |-- users_count: string (nullable = true)
```

```
dfFromRDD2 = spark.createDataFrame(rdd).toDF(*columns)
```

```
dfFromData2 = spark.createDataFrame(data).toDF(*columns)
dfFromData2.show(truncate= False)
```

```
+--------+-----------+
|language|users_count|
+--------+-----------+
|Java    |20000      |
|Python  |100000     |
|Scala   |3000       |
+--------+-----------+
```

```
from pyspark.sql.types import StructType,StructField, StringType, IntegerType
data2 = [("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)
  ]

schema = StructType([ \
    StructField("firstname",StringType(),True), \
    StructField("middlename",StringType(),True), \
    StructField("lastname",StringType(),True), \
    StructField("id", StringType(), True), \
    StructField("gender", StringType(), True), \
    StructField("salary", IntegerType(), True) \
  ])

df = spark.createDataFrame(data=data2,schema=schema)
df.printSchema()
df.show(truncate=False)
```

```
root
 |-- firstname: string (nullable = true)
 |-- middlename: string (nullable = true)
 |-- lastname: string (nullable = true)
 |-- id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)

+---------+----------+--------+-----+------+------+
|firstname|middlename|lastname|id   |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    |
+---------+----------+--------+-----+------+------+
```

```
emptyRDD = spark.sparkContext.emptyRDD()
print(emptyRDD)
```

```
EmptyRDD[49] at emptyRDD at NativeMethodAccessorImpl.java:0
```

```
#Create Schema
from pyspark.sql.types import StructType,StructField, StringType
schema = StructType([
  StructField('firstname', StringType(), True),
  StructField('middlename', StringType(), True),
  StructField('lastname', StringType(), True)
  ])
```

```
df = spark.createDataFrame(emptyRDD,schema)
df.printSchema()
```

```
root
 |-- firstname: string (nullable = true)
 |-- middlename: string (nullable = true)
 |-- lastname: string (nullable = true)
```

```
#Convert empty RDD to Dataframe
df1 = emptyRDD.toDF(schema)
df1.printSchema()
```

```
root
 |-- firstname: string (nullable = true)
 |-- middlename: string (nullable = true)
 |-- lastname: string (nullable = true)
```

```
df2 = spark.createDataFrame([], schema)
df2.printSchema()
```

```
root
 |-- firstname: string (nullable = true)
 |-- middlename: string (nullable = true)
 |-- lastname: string (nullable = true)
```

```
#Create empty DatFrame with no schema (no columns)
df3 = spark.createDataFrame([], StructType([]))
df3.printSchema()

#print below empty schema
#root
```

```
root
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
columns = ["Seqno","Quote"]
data = [("1", "Be the change that you wish to see in the world"),
    ("2", "Everyone thinks of changing the world, but no one thinks of changing
himself."),
    ("3", "The purpose of our lives is to be happy."),
    ("4", "Be cool.")]
df = spark.createDataFrame(data,columns)
df.show()
```

```
+-----+--------------------+
|Seqno|               Quote|
+-----+--------------------+
|    1|Be the change tha...|
|    2|Everyone thinks o...|
|    3|The purpose of ou...|
|    4|            Be cool.|
+-----+--------------------+
```

```
#Display full column contents
df.show(truncate=False)
```

```
+-----+-----------------------------------------------------------------------
------+
|Seqno|Quote
|
+-----+-----------------------------------------------------------------------
------+
```

```
|1    |Be the change that you wish to see in the world
|
|2    |Everyone thinks of changing the world, but no one thinks of changing hi
mself.|
|3    |The purpose of our lives is to be happy.
|
|4    |Be cool.
|
+-----+--------------------------------------------------------------------
------+
```

```
df.show(2,truncate=False)
```

```
+-----+--------------------------------------------------------------------
------+
|Seqno|Quote
|
+-----+--------------------------------------------------------------------
------+
|1    |Be the change that you wish to see in the world
|
|2    |Everyone thinks of changing the world, but no one thinks of changing hi
mself.|
+-----+--------------------------------------------------------------------
------+
only showing top 2 rows
```

```
df.show(2,truncate=25)
```

```
+-----+-------------------------+
|Seqno|                    Quote|
+-----+-------------------------+
|    1|Be the change that you...|
|    2|Everyone thinks of cha...|
+-----+-------------------------+
only showing top 2 rows
```

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType,StructField, StringType, IntegerType

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

data = [("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)
  ]

schema = StructType([ \
    StructField("firstname",StringType(),True), \
    StructField("middlename",StringType(),True), \
    StructField("lastname",StringType(),True), \
    StructField("id", StringType(), True), \
    StructField("gender", StringType(), True), \
    StructField("salary", IntegerType(), True) \
  ])

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

root
 |-- firstname: string (nullable = true)
 |-- middlename: string (nullable = true)
 |-- lastname: string (nullable = true)
 |-- id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)

+---------+----------+--------+-----+------+------+
|firstname|middlename|lastname|id   |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    |
+---------+----------+--------+-----+------+------+
```

```
structureData = [
    (("James","","Smith"),"36636","M",3100),
    (("Michael","Rose",""),"40288","M",4300),
    (("Robert","","Williams"),"42114","M",1400),
    (("Maria","Anne","Jones"),"39192","F",5500),
    (("Jen","Mary","Brown"),"","F",-1)
  ]
structureSchema = StructType([
        StructField('name', StructType([
             StructField('firstname', StringType(), True),
             StructField('middlename', StringType(), True),
             StructField('lastname', StringType(), True)
             ])),
         StructField('id', StringType(), True),
         StructField('gender', StringType(), True),
         StructField('salary', IntegerType(), True)
         ])

df2 = spark.createDataFrame(data=structureData,schema=structureSchema)
df2.printSchema()
df2.show(truncate=False)
```

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)

+--------------------+-----+------+------+
|name                |id   |gender|salary|
+--------------------+-----+------+------+
|{James, , Smith}    |36636|M     |3100  |
|{Michael, Rose, }   |40288|M     |4300  |
|{Robert, , Williams}|42114|M     |1400  |
|{Maria, Anne, Jones}|39192|F     |5500  |
|{Jen, Mary, Brown}  |     |F     |-1    |
+--------------------+-----+------+------+
```

```
from pyspark.sql.functions import col,struct,when
updatedDF = df2.withColumn("OtherInfo",
    struct(col("id").alias("identifier"),
    col("gender").alias("gender"),
    col("salary").alias("salary"),
    when(col("salary").cast(IntegerType()) < 2000,"Low")
      .when(col("salary").cast(IntegerType()) < 4000,"Medium")
      .otherwise("High").alias("Salary_Grade")
  )).drop("id","gender","salary")

updatedDF.printSchema()
updatedDF.show(truncate=False)
```

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- OtherInfo: struct (nullable = false)
 |    |-- identifier: string (nullable = true)
 |    |-- gender: string (nullable = true)
 |    |-- salary: integer (nullable = true)
 |    |-- Salary_Grade: string (nullable = false)

+--------------------+-----------------------+
|name                |OtherInfo              |
+--------------------+-----------------------+
|{James, , Smith}    |{36636, M, 3100, Medium}|
|{Michael, Rose, }   |{40288, M, 4300, High}  |
|{Robert, , Williams}|{42114, M, 1400, Low}   |
|{Maria, Anne, Jones}|{39192, F, 5500, High}  |
|{Jen, Mary, Brown}  |{, F, -1, Low}          |
+--------------------+-----------------------+
```

```
import pyspark
from pyspark.sql import SparkSession

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)
```

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

```
df.select("firstname","lastname").show()
df.select(df.firstname,df.lastname).show()
df.select(df["firstname"],df["lastname"]).show()

#By using col() function
from pyspark.sql.functions import col
df.select(col("firstname"),col("lastname")).show()

#Select columns by regular expression
df.select(df.colRegex("`^.*name*`")).show()
```

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

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

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

+---------+--------+
|firstname|lastname|
```

```
# Select All columns from List
df.select(*columns).show()

# Select All columns
df.select([col for col in df.columns]).show()
df.select("*").show()
```

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

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

+---------+--------+-------+-----+
|firstname|lastname|country|state|
+---------+--------+-------+-----+
```

```
#Selects first 3 columns and top 3 rows
df.select(df.columns[0:2]).show(3)

#Selects columns 2 to 4  and top 3 rows
df.select(df.columns[2:4]).show(3)
```

```
+---------+--------+
|firstname|lastname|
+---------+--------+
|    James|   Smith|
|  Michael|    Rose|
|   Robert|Williams|
+---------+--------+
only showing top 3 rows

+-------+-----+
|country|state|
+-------+-----+
|    USA|   CA|
|    USA|   NY|
|    USA|   CA|
+-------+-----+
only showing top 3 rows
```

```
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
```

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- state: string (nullable = true)
 |-- gender: string (nullable = true)

+----------------------+-----+------+
|name                  |state|gender|
+----------------------+-----+------+
|{James, null, 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     |
+----------------------+-----+------+
```

```
df2.select("name").show(truncate=False)
```

```
+----------------------+
|name                  |
+----------------------+
|{James, null, Smith}  |
|{Anna, Rose, }        |
|{Julia, , Williams}   |
|{Maria, Anne, Jones}  |
|{Jen, Mary, Brown}    |
|{Mike, Mary, Williams}|
+----------------------+
```

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

```
+---------+--------+
|firstname|lastname|
+---------+--------+
|James    |Smith   |
|Anna     |        |
|Julia    |Williams|
|Maria    |Jones   |
|Jen      |Brown   |
|Mike     |Williams|
+---------+--------+
```

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

```
+---------+----------+--------+
|firstname|middlename|lastname|
+---------+----------+--------+
|James    |null      |Smith   |
|Anna     |Rose      |        |
|Julia    |          |Williams|
|Maria    |Anne      |Jones   |
|Jen      |Mary      |Brown   |
|Mike     |Mary      |Williams|
+---------+----------+--------+
```

```
import pyspark
from pyspark.sql import SparkSession

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

dept = [("Finance",10), \
    ("Marketing",20), \
    ("Sales",30), \
    ("IT",40) \
  ]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.show(truncate=False)
```

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

```
dataCollect = deptDF.collect()
print(dataCollect)
```

```
[Row(dept_name='Finance', dept_id=10), Row(dept_name='Marketing', dept_id=20),
Row(dept_name='Sales', dept_id=30), Row(dept_name='IT', dept_id=40)]
```

```
for row in dataCollect:
    print(row['dept_name'] + "," +str(row['dept_id']))
```

```
Finance,10
Marketing,20
Sales,30
IT,40
```

```
#Returns value of First Row, First Column which is "Finance"
deptDF.collect()[0][0]
```

```
Out[42]: 'Finance'
```

```
dataCollect = deptDF.select("dept_name").collect()
```

```
data = [('James','','Smith','1991-04-01','M',3000),
  ('Michael','Rose','','2000-05-19','M',4000),
  ('Robert','','Williams','1978-09-05','M',4000),
  ('Maria','Anne','Jones','1967-12-01','F',4000),
  ('Jen','Mary','Brown','1980-02-17','F',-1)
]

columns = ["firstname","middlename","lastname","dob","gender","salary"]
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
df = spark.createDataFrame(data=data, schema = columns)
```

```
df.withColumn("salary",col("salary").cast("Integer")).show()
```

```
+---------+----------+--------+----------+------+------+
|firstname|middlename|lastname|       dob|gender|salary|
+---------+----------+--------+----------+------+------+
|    James|          |   Smith|1991-04-01|     M|  3000|
|  Michael|      Rose|        |2000-05-19|     M|  4000|
|   Robert|          |Williams|1978-09-05|     M|  4000|
|    Maria|      Anne|   Jones|1967-12-01|     F|  4000|
|      Jen|      Mary|   Brown|1980-02-17|     F|    -1|
+---------+----------+--------+----------+------+------+
```

```
df.withColumn("salary",col("salary")*100).show()
```

```
+---------+----------+--------+----------+------+------+
|firstname|middlename|lastname|       dob|gender|salary|
+---------+----------+--------+----------+------+------+
|    James|          |   Smith|1991-04-01|     M|300000|
|  Michael|      Rose|        |2000-05-19|     M|400000|
|   Robert|          |Williams|1978-09-05|     M|400000|
|    Maria|      Anne|   Jones|1967-12-01|     F|400000|
|      Jen|      Mary|   Brown|1980-02-17|     F|  -100|
+---------+----------+--------+----------+------+------+
```

```
df.withColumn("CopiedColumn",col("salary")* -1).show()
```

```
+---------+----------+--------+----------+------+------+------------+
|firstname|middlename|lastname|       dob|gender|salary|CopiedColumn|
+---------+----------+--------+----------+------+------+------------+
|    James|          |   Smith|1991-04-01|     M|  3000|       -3000|
|  Michael|      Rose|        |2000-05-19|     M|  4000|       -4000|
|   Robert|          |Williams|1978-09-05|     M|  4000|       -4000|
|    Maria|      Anne|   Jones|1967-12-01|     F|  4000|       -4000|
|      Jen|      Mary|   Brown|1980-02-17|     F|    -1|           1|
+---------+----------+--------+----------+------+------+------------+
```

```
from pyspark.sql.functions import lit
df.withColumn("Country", lit("USA")).show()
df.withColumn("Country", lit("USA")) \
  .withColumn("anotherColumn",lit("anotherValue")) \
  .show()
```

```
+---------+----------+--------+----------+------+------+-------+
|firstname|middlename|lastname|       dob|gender|salary|Country|
+---------+----------+--------+----------+------+------+-------+
|    James|          |   Smith|1991-04-01|     M|  3000|    USA|
|  Michael|      Rose|        |2000-05-19|     M|  4000|    USA|
|   Robert|          |Williams|1978-09-05|     M|  4000|    USA|
|    Maria|      Anne|   Jones|1967-12-01|     F|  4000|    USA|
|      Jen|      Mary|   Brown|1980-02-17|     F|    -1|    USA|
+---------+----------+--------+----------+------+------+-------+
```

```
+---------+----------+--------+----------+------+------+-------+-------------+
|firstname|middlename|lastname|       dob|gender|salary|Country|anotherColumn|
+---------+----------+--------+----------+------+------+-------+-------------+
|    James|          |   Smith|1991-04-01|     M|  3000|    USA| anotherValue|
|  Michael|      Rose|        |2000-05-19|     M|  4000|    USA| anotherValue|
|   Robert|          |Williams|1978-09-05|     M|  4000|    USA| anotherValue|
|    Maria|      Anne|   Jones|1967-12-01|     F|  4000|    USA| anotherValue|
|      Jen|      Mary|   Brown|1980-02-17|     F|    -1|    USA| anotherValue|
+---------+----------+--------+----------+------+------+-------+-------------+
```

```
df.withColumnRenamed("gender","sex") \
  .show(truncate=False)
```

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

```
|firstname|middlename|lastname|dob       |sex|salary|
+---------+----------+--------+----------+---+------+
|James    |          |Smith   |1991-04-01|M  |3000  |
|Michael  |Rose      |        |2000-05-19|M  |4000  |
|Robert   |          |Williams|1978-09-05|M  |4000  |
|Maria    |Anne      |Jones   |1967-12-01|F  |4000  |
|Jen      |Mary      |Brown   |1980-02-17|F  |-1    |
+---------+----------+--------+----------+---+------+
```

```
df.drop("salary") \
  .show()
```

```
+---------+----------+--------+----------+------+
|firstname|middlename|lastname|       dob|gender|
+---------+----------+--------+----------+------+
|    James|          |   Smith|1991-04-01|     M|
|  Michael|      Rose|        |2000-05-19|     M|
|   Robert|          |Williams|1978-09-05|     M|
|    Maria|      Anne|   Jones|1967-12-01|     F|
|      Jen|      Mary|   Brown|1980-02-17|     F|
+---------+----------+--------+----------+------+
```

```
df2 = df.withColumnRenamed("firstname","FName")\
        .withColumnRenamed("lastname","LName")
df2.show(truncate=False)
```

```
+-------+----------+--------+----------+------+------+
|FName  |middlename|LName   |dob       |gender|salary|
+-------+----------+--------+----------+------+------+
|James  |          |Smith   |1991-04-01|M     |3000  |
|Michael|Rose      |        |2000-05-19|M     |4000  |
|Robert |          |Williams|1978-09-05|M     |4000  |
|Maria  |Anne      |Jones   |1967-12-01|F     |4000  |
|Jen    |Mary      |Brown   |1980-02-17|F     |-1    |
+-------+----------+--------+----------+------+------+
```

```
from pyspark.sql.types import StructType,StructField
from pyspark.sql.types import StringType, IntegerType, ArrayType
data = [
    (("James","","Smith"),["Java","Scala","C++"],"OH","M"),
    (("Anna","Rose",""),["Spark","Java","C++"],"NY","F"),
    (("Julia","","Williams"),["CSharp","VB"],"OH","F"),
    (("Maria","Anne","Jones"),["CSharp","VB"],"NY","M"),
    (("Jen","Mary","Brown"),["CSharp","VB"],"NY","M"),
    (("Mike","Mary","Williams"),["Python","VB"],"OH","M")
 ]

schema = StructType([
     StructField('name', StructType([
        StructField('firstname', StringType(), True),
        StructField('middlename', StringType(), True),
         StructField('lastname', StringType(), True)
     ])),
     StructField('languages', ArrayType(StringType()), True),
     StructField('state', StringType(), True),
     StructField('gender', StringType(), True)
 ])

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

```
from pyspark.sql.types import StructType,StructField
from pyspark.sql.types import StringType, IntegerType, ArrayType
data = [
    (("James","","Smith"),["Java","Scala","C++"],"OH","M"),
    (("Anna","Rose",""),["Spark","Java","C++"],"NY","F"),
    (("Julia","","Williams"),["CSharp","VB"],"OH","F"),
    (("Maria","Anne","Jones"),["CSharp","VB"],"NY","M"),
    (("Jen","Mary","Brown"),["CSharp","VB"],"NY","M"),
    (("Mike","Mary","Williams"),["Python","VB"],"OH","M")
 ]

schema = StructType([
     StructField('name', StructType([
        StructField('firstname', StringType(), True),
        StructField('middlename', StringType(), True),
         StructField('lastname', StringType(), True)
     ])),
     StructField('languages', ArrayType(StringType()), True),
     StructField('state', StringType(), True),
     StructField('gender', StringType(), True)
 ])

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

```
root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- languages: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- state: string (nullable = true)
 |-- gender: string (nullable = true)

+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Anna, Rose, }        |[Spark, Java, C++]|NY   |F     |
|{Julia, , Williams}   |[CSharp, VB]      |OH   |F     |
|{Maria, Anne, Jones}  |[CSharp, VB]      |NY   |M     |
|{Jen, Mary, Brown}    |[CSharp, VB]      |NY   |M     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
```

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

```
df.filter(df.state == "OH").show(truncate = False)
```

```
+---------------------+-----------------+-----+------+
|name                 |languages        |state|gender|
+---------------------+-----------------+-----+------+
|{James, , Smith}     |[Java, Scala, C++]|OH   |M     |
|{Julia, , Williams}  |[CSharp, VB]     |OH   |F     |
|{Mike, Mary, Williams}|[Python, VB]     |OH   |M     |
+---------------------+-----------------+-----+------+
```

```
df.filter(df.state != "OH") \
    .show(truncate=False)
df.filter(~(df.state == "OH")) \
    .show(truncate=False)
```

```
+-------------------+-----------------+-----+------+
|name               |languages        |state|gender|
+-------------------+-----------------+-----+------+
|{Anna, Rose, }     |[Spark, Java, C++]|NY   |F     |
|{Maria, Anne, Jones}|[CSharp, VB]     |NY   |M     |
|{Jen, Mary, Brown} |[CSharp, VB]     |NY   |M     |
+-------------------+-----------------+-----+------+
```

```
+-------------------+-----------------+-----+------+
|name               |languages        |state|gender|
+-------------------+-----------------+-----+------+
|{Anna, Rose, }     |[Spark, Java, C++]|NY   |F     |
|{Maria, Anne, Jones}|[CSharp, VB]     |NY   |M     |
|{Jen, Mary, Brown} |[CSharp, VB]     |NY   |M     |
+-------------------+-----------------+-----+------+
```

```
from pyspark.sql.functions import col
df.filter(col("state") == "OH")\
        .show(truncate = False)
```

```
+---------------------+-----------------+-----+------+
|name                 |languages        |state|gender|
+---------------------+-----------------+-----+------+
|{James, , Smith}     |[Java, Scala, C++]|OH   |M     |
|{Julia, , Williams}  |[CSharp, VB]     |OH   |F     |
|{Mike, Mary, Williams}|[Python, VB]     |OH   |M     |
+---------------------+-----------------+-----+------+
```

```
#Using SQL Expression
df.filter("gender == 'M'").show()
#For not equal
df.filter("gender != 'M'").show()
df.filter("gender <> 'M'").show()
```

```
+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+-----------------+-----+------+

+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|     {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Julia, , Williams}|      [CSharp, VB]|   OH|     F|
+--------------------+-----------------+-----+------+

+--------------------+-----------------+-----+------+
|                name|        languages|state|gender|
+--------------------+-----------------+-----+------+
|     {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Julia, , Williams}|      [CSharp, VB]|   OH|     F|
```

```
df.filter( (df.state  == "OH") & (df.gender  == "M") ) \
    .show(truncate=False)
```

```
+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
+----------------------+------------------+-----+------+
```

```
df.filter((col("state") == "OH") & (col("gender") == "M"))\
  .show(truncate = False)
```

```
+----------------------+------------------+-----+------+
|name                  |languages         |state|gender|
+----------------------+------------------+-----+------+
```

```
|{James, , Smith}      |[Java, Scala, C++]|OH   |M     |
|{Mike, Mary, Williams}|[Python, VB]      |OH   |M     |
+----------------------+------------------+-----+------+
```

```
li=["OH","CA","DE"]
df.filter(col("state").isin(li)).show()
```

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
| {Julia, , Williams}|      [CSharp, VB]|   OH|     F|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

```
df.filter(~df.state.isin(li)).show()
df.filter(df.state.isin(li)==False).show()
```

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|      {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+------------------+-----+------+
```

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|      {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+------------------+-----+------+
```

```
df.filter(df.state.startswith("N")).show()
```

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|      {Anna, Rose, }|[Spark, Java, C++]|   NY|     F|
|{Maria, Anne, Jones}|      [CSharp, VB]|   NY|     M|
|  {Jen, Mary, Brown}|      [CSharp, VB]|   NY|     M|
+--------------------+------------------+-----+------+
```

```
df.filter(df.state.endswith("H")).show()
```

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
| {Julia, , Williams}|      [CSharp, VB]|   OH|     F|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

```
df.filter(df.state.contains("H")).show()
```

```
+--------------------+------------------+-----+------+
|                name|         languages|state|gender|
+--------------------+------------------+-----+------+
|    {James, , Smith}|[Java, Scala, C++]|   OH|     M|
| {Julia, , Williams}|      [CSharp, VB]|   OH|     F|
|{Mike, Mary, Will...|      [Python, VB]|   OH|     M|
+--------------------+------------------+-----+------+
```

```
data2 = [(2,"Michael Rose"),(3,"Robert Williams"),
     (4,"Rames Rose"),(5,"Rames rose")
  ]
df2 = spark.createDataFrame(data = data2, schema = ["id","name"])

# like - SQL LIKE pattern
df2.filter(df2.name.like("%rose%")).show()

# rlike - SQL RLIKE pattern (LIKE with Regex)
#This check case insensitive
df2.filter(df2.name.rlike("(?i)^*rose$")).show()
```

```
+---+----------+
| id|      name|
+---+----------+
|  5|Rames rose|
+---+----------+
```

```
+---+------------+
| id|        name|
+---+------------+
```

```
|  2|Michael Rose|
|  4|  Rames Rose|
|  5|  Rames rose|
+---+------------+
```

```
from pyspark.sql.functions import array_contains
df.filter(array_contains(df.languages,"Java")) \
    .show(truncate=False)
```

```
+----------------+------------------+-----+------+
|name            |languages         |state|gender|
+----------------+------------------+-----+------+
|{James, , Smith}|[Java, Scala, C++]|OH   |M     |
|{Anna, Rose, }  |[Spark, Java, C++]|NY   |F     |
+----------------+------------------+-----+------+
```

```
df.filter(df.name.lastname == "Williams") \
    .show(truncate=False)
```

```
+----------------------+------------+-----+------+
|name                  |languages   |state|gender|
+----------------------+------------+-----+------+
|{Julia, , Williams}   |[CSharp, VB]|OH   |F     |
|{Mike, Mary, Williams}|[Python, VB]|OH   |M     |
+----------------------+------------+-----+------+
```

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

data = [("James", "Sales", 3000), \
    ("Michael", "Sales", 4600), \
    ("Robert", "Sales", 4100), \
    ("Maria", "Finance", 3000), \
    ("James", "Sales", 3000), \
    ("Scott", "Finance", 3300), \
    ("Jen", "Finance", 3900), \
    ("Jeff", "Marketing", 3000), \
    ("Kumar", "Marketing", 2000), \
    ("Saif", "Sales", 4100) \
  ]
columns= ["employee_name", "department", "salary"]
df = spark.createDataFrame(data = data, schema = columns)
df.printSchema()
df.show(truncate=False)
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- salary: long (nullable = true)

+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|James        |Sales     |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3900  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+
```

```
distinctDF = df.distinct()
print("Distinct count: "+str(distinctDF.count()))
distinctDF.show(truncate=False)
```

```
Distinct count: 9
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3900  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+
```

```
df2 = df.dropDuplicates()
print("Distinct count: "+str(df2.count()))
df2.show(truncate=False)
```

```
Distinct count: 9
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3900  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+
```

```
dropDisDF = df.dropDuplicates(["department","salary"])
print("Distinct count of department & salary : "+str(dropDisDF.count()))
dropDisDF.show(truncate=False)
```

```
Distinct count of department & salary : 8
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
```

```
|Maria        |Finance   |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3900  |
|Kumar        |Marketing |2000  |
|Jeff         |Marketing |3000  |
|James        |Sales     |3000  |
|Robert       |Sales     |4100  |
|Michael      |Sales     |4600  |
+-------------+----------+------+
```

```
simpleData = [("James","Sales","NY",90000,34,10000), \
    ("Michael","Sales","NY",86000,56,20000), \
    ("Robert","Sales","CA",81000,30,23000), \
    ("Maria","Finance","CA",90000,24,23000), \
    ("Raman","Finance","CA",99000,40,24000), \
    ("Scott","Finance","NY",83000,36,19000), \
    ("Jen","Finance","NY",79000,53,15000), \
    ("Jeff","Marketing","CA",80000,25,18000), \
    ("Kumar","Marketing","NY",91000,50,21000) \
  ]
columns= ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data = simpleData, schema = columns)
df.printSchema()
df.show(truncate=False)
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Raman        |Finance   |CA   |99000 |40 |24000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
```

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

```
df.sort("department","state").show(truncate=False)
df.sort(col("department"),col("state")).show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Maria        |Finance   |CA   |90000 |24 |23000|
|Raman        |Finance   |CA   |99000 |40 |24000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|James        |Sales     |NY   |90000 |34 |10000|
+-------------+----------+-----+------+---+-----+

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Raman        |Finance   |CA   |99000 |40 |24000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jen          |Finance   |NY   |79000 |53 |15000|
```

```
df.orderBy("department","state").show(truncate=False)
df.orderBy(col("department"),col("state")).show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Raman        |Finance   |CA   |99000 |40 |24000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|James        |Sales     |NY   |90000 |34 |10000|
+-------------+----------+-----+------+---+-----+

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
```

```
+-------------+----------+-----+------+---+-----+
|Raman        |Finance   |CA   |99000 |40 |24000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Scott        |Finance   |NY   |83000 |36 |19000|
```

```
df.sort(df.department.asc(),df.state.asc()).show(truncate=False)
df.sort(col("department").asc(),col("state").asc()).show(truncate=False)
df.orderBy(col("department").asc(),col("state").asc()).show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Maria        |Finance   |CA   |90000 |24 |23000|
|Raman        |Finance   |CA   |99000 |40 |24000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|James        |Sales     |NY   |90000 |34 |10000|
+-------------+----------+-----+------+---+-----+

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Raman        |Finance   |CA   |99000 |40 |24000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Scott        |Finance   |NY   |83000 |36 |19000|
```

```
df.sort(df.department.asc(),df.state.desc()).show(truncate=False)
df.sort(col("department").asc(),col("state").desc()).show(truncate=False)
df.orderBy(col("department").asc(),col("state").desc()).show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Raman        |Finance   |CA   |99000 |40 |24000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Michael      |Sales     |NY   |86000 |56 |20000|
```

```
|James        |Sales     |NY   |90000 |34 |10000|
|Robert       |Sales     |CA   |81000 |30 |23000|
+-------------+----------+-----+------+---+-----+

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Raman        |Finance   |CA   |99000 |40 |24000|
```

```
df.createOrReplaceTempView("EMP")
spark.sql("select employee_name,department,state,salary,age,bonus from EMP
ORDER BY department asc").show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|Raman        |Finance   |CA   |99000 |40 |24000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|James        |Sales     |NY   |90000 |34 |10000|
+-------------+----------+-----+------+---+-----+
```

```
simpleData = [("James","Sales","NY",90000,34,10000),
    ("Michael","Sales","NY",86000,56,20000),
    ("Robert","Sales","CA",81000,30,23000),
    ("Maria","Finance","CA",90000,24,23000),
    ("Raman","Finance","CA",99000,40,24000),
    ("Scott","Finance","NY",83000,36,19000),
    ("Jen","Finance","NY",79000,53,15000),
    ("Jeff","Marketing","CA",80000,25,18000),
    ("Kumar","Marketing","NY",91000,50,21000)
  ]

schema = ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data=simpleData, schema = schema)
df.printSchema()
df.show(truncate=False)
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Raman        |Finance   |CA   |99000 |40 |24000|
|Scott        |Finance   |NY   |83000 |36 |19000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

```
df = df.withColumn("salary",df.salary.cast(IntegerType()))
df.printSchema()
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
```

```
 |-- salary: integer (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)
```

```
df.groupBy("department").sum("salary").alias("Totalsal").show(truncate = False)
```

```
+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|Sales     |257000     |
|Finance   |351000     |
|Marketing |171000     |
+----------+-----------+
```

```
from pyspark.sql.functions import sum
df.groupBy("department").agg(sum("salary").alias("TotalSalary")).show(truncate
= False)
```

```
+----------+-----------+
|department|TotalSalary|
+----------+-----------+
|Sales     |257000     |
|Finance   |351000     |
|Marketing |171000     |
+----------+-----------+
```

```
df.groupBy("department","state") \
    .sum("salary","bonus") \
    .show(truncate = False)
```

```
+----------+-----+-----------+----------+
|department|state|sum(salary)|sum(bonus)|
+----------+-----+-----------+----------+
|Sales     |NY   |176000     |30000     |
|Sales     |CA   |81000      |23000     |
|Finance   |CA   |189000     |47000     |
|Finance   |NY   |162000     |34000     |
|Marketing |NY   |91000      |21000     |
|Marketing |CA   |80000      |18000     |
+----------+-----+-----------+----------+
```

```
df.groupBy("department") \
    .agg(sum("salary").alias("sum_salary"), \
         avg("salary").alias("avg_salary"), \
         sum("bonus").alias("sum_bonus"), \
         max("bonus").alias("max_bonus") \
     ) \
    .show(truncate=False)
```

```
+----------+----------+-----------------+---------+---------+
|department|sum_salary|avg_salary       |sum_bonus|max_bonus|
+----------+----------+-----------------+---------+---------+
|Sales     |257000    |85666.66666666667|53000    |23000    |
|Finance   |351000    |87750.0          |81000    |24000    |
|Marketing |171000    |85500.0          |39000    |21000    |
+----------+----------+-----------------+---------+---------+
```

```
df.groupBy("department") \
    .agg(sum("salary").alias("sum_salary"), \
      avg("salary").alias("avg_salary"), \
      sum("bonus").alias("sum_bonus"), \
      max("bonus").alias("max_bonus")) \
    .where(col("sum_bonus") >= 50000) \
    .show(truncate=False)
```

```
+----------+----------+-----------------+---------+---------+
|department|sum_salary|avg_salary       |sum_bonus|max_bonus|
+----------+----------+-----------------+---------+---------+
|Sales     |257000    |85666.66666666667|53000    |23000    |
|Finance   |351000    |87750.0          |81000    |24000    |
+----------+----------+-----------------+---------+---------+
```

```
emp = [(1,"Smith",-1,"2018","10","M",3000), \
    (2,"Rose",1,"2010","20","M",4000), \
    (3,"Williams",1,"2010","10","M",1000), \
    (4,"Jones",2,"2005","10","F",2000), \
    (5,"Brown",2,"2010","40","",-1), \
      (6,"Brown",2,"2010","50","",-1) \
  ]
empColumns = ["emp_id","name","superior_emp_id","year_joined", \
       "emp_dept_id","gender","salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
empDF.printSchema()
empDF.show(truncate=False)

dept = [("Finance",10), \
    ("Marketing",20), \
    ("Sales",30), \
    ("IT",40) \
  ]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()
deptDF.show(truncate=False)
```

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+--------+---------------+-----------+-----------+------+------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |
|2     |Rose    |1              |2010       |20         |M     |4000  |
|3     |Williams|1              |2010       |10         |M     |1000  |
|4     |Jones   |2              |2005       |10         |F     |2000  |
|5     |Brown   |2              |2010       |40         |      |-1    |
|6     |Brown   |2              |2010       |50         |      |-1    |
+------+--------+---------------+-----------+-----------+------+------+

root
```

```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner") \
     .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+-------
--+--------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_na
me|dept_id|
+------+--------+---------------+-----------+-----------+------+------+-------
--+--------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance
|10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance
|10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance
|10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketi
ng|20     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT
|40     |
+------+--------+---------------+-----------+-----------+------+------+-------
--+--------+
```

```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"full") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"fullouter") \
    .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+-------
--+--------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_na
me|dept_id|
+------+--------+---------------+-----------+-----------+------+------+-------
--+--------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance
|10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance
|10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance
|10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketi
ng|20     |
```

```
|null  |null    |null           |null       |null       |null  |null  |Sales
|30     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT
|40     |
|6     |Brown   |2              |2010       |50         |      |-1    |null
|null   |
```

```
import pyspark
from pyspark.sql import SparkSession

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

simpleData = [("James","Sales","NY",90000,34,10000), \
    ("Michael","Sales","NY",86000,56,20000), \
    ("Robert","Sales","CA",81000,30,23000), \
    ("Maria","Finance","CA",90000,24,23000) \
  ]

columns= ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data = simpleData, schema = columns)
df.printSchema()
df.show(truncate=False)
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
+-------------+----------+-----+------+---+-----+
```

```
simpleData2 = [("James","Sales","NY",90000,34,10000), \
    ("Maria","Finance","CA",90000,24,23000), \
    ("Jen","Finance","NY",79000,53,15000), \
    ("Jeff","Marketing","CA",80000,25,18000), \
    ("Kumar","Marketing","NY",91000,50,21000) \
  ]
columns2= ["employee_name","department","state","salary","age","bonus"]

df2 = spark.createDataFrame(data = simpleData2, schema = columns2)

df2.printSchema()
df2.show(truncate=False)
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

```
unionDF = df.union(df2)
unionDF.show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
```

```
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

```
unionAllDF = df.unionAll(df2)
unionAllDF.show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|James        |Sales     |NY   |90000 |34 |10000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

```
disDF = df.union(df2).distinct()
disDF.show(truncate=False)
```

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|James        |Sales     |NY   |90000 |34 |10000|
|Michael      |Sales     |NY   |86000 |56 |20000|
|Robert       |Sales     |CA   |81000 |30 |23000|
|Maria        |Finance   |CA   |90000 |24 |23000|
|Jen          |Finance   |NY   |79000 |53 |15000|
|Jeff         |Marketing |CA   |80000 |25 |18000|
|Kumar        |Marketing |NY   |91000 |50 |21000|
+-------------+----------+-----+------+---+-----+
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

#Create DataFrame df1 with columns name,dept & age
data = [("James","Sales",34), ("Michael","Sales",56), \
    ("Robert","Sales",30), ("Maria","Finance",24) ]
columns= ["name","dept","age"]
df1 = spark.createDataFrame(data = data, schema = columns)
df1.printSchema()

#Create DataFrame df1 with columns name,dep,state & salary
data2=[("James","Sales","NY",9000),("Maria","Finance","CA",9000), \
    ("Jen","Finance","NY",7900),("Jeff","Marketing","CA",8000)]
columns2= ["name","dept","state","salary"]
df2 = spark.createDataFrame(data = data2, schema = columns2)
df2.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- dept: string (nullable = true)
 |-- age: long (nullable = true)

root
 |-- name: string (nullable = true)
 |-- dept: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
```

```
#Add missing columns 'state' & 'salary' to df1
from pyspark.sql.functions import lit
for column in [column for column in df2.columns if column not in df1.columns]:
    df1 = df1.withColumn(column, lit(None))

#Add missing column 'age' to df2
for column in [column for column in df1.columns if column not in df2.columns]:
    df2 = df2.withColumn(column, lit(None))



#Finally join two dataframe's df1 & df2 by name
merged_df=df1.unionByName(df2)
merged_df.show()
```

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

```
|   name|     dept| age|state|salary|
+-------+---------+----+-----+------+
|  James|    Sales|  34| null|  null|
|Michael|    Sales|  56| null|  null|
| Robert|    Sales|  30| null|  null|
|  Maria|  Finance|  24| null|  null|
|  James|    Sales|null|   NY|  9000|
|  Maria|  Finance|null|   CA|  9000|
|    Jen|  Finance|null|   NY|  7900|
|   Jeff|Marketing|null|   CA|  8000|
+-------+---------+----+-----+------+
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
data = [("James","M",60000),("Michael","M",70000),
        ("Robert",None,400000),("Maria","F",500000),
        ("Jen","",None)]

columns = ["name","gender","salary"]
df = spark.createDataFrame(data = data, schema = columns)
df.show()
```

```
+-------+------+------+
|   name|gender|salary|
+-------+------+------+
|  James|     M| 60000|
|Michael|     M| 70000|
| Robert|  null|400000|
|  Maria|     F|500000|
|    Jen|      |  null|
+-------+------+------+
```

```
from pyspark.sql.functions import when,col

df2 = df.withColumn("new_gender",when(col("gender") == "M","Male")\
                    .when(col("gender") == "F","Female")\
                    .when(col("gender").isNull(),"")\
                    .otherwise(col("gender"))
                   )
df2.show()
```

```
+-------+------+------+----------+
|   name|gender|salary|new_gender|
+-------+------+------+----------+
```

```
|  James|     M| 60000|      Male|
|Michael|     M| 70000|      Male|
| Robert|  null|400000|          |
|  Maria|     F|500000|    Female|
|    Jen|      |  null|          |
+-------+------+------+----------+
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
         .appName('SparkByExamples.com') \
         .getOrCreate()

data = [("James, A, Smith","2018","M",3000),
            ("Michael, Rose, Jones","2010","M",4000),
            ("Robert,K,Williams","2010","M",4000),
            ("Maria,Anne,Jones","2005","F",4000),
            ("Jen,Mary,Brown","2010","",-1)
            ]

columns=["name","dob_year","gender","salary"]
df=spark.createDataFrame(data,columns)
df.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- dob_year: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)
```

```
from pyspark.sql.functions import split, col
df2 = df.select(split(col("name"),",").alias("NameArray")) \
    .drop("name")
df2.printSchema()
df2.show()
```

```
root
 |-- NameArray: string (nullable = true)

+---------+
|NameArray|
+---------+
|    James|
```

```
|  Michael|
|   Robert|
|    Maria|
|      Jen|
+---------+
```

```
%sql
SELECT try_cast('a' AS INT)
```

Table

	TRY_CAST(a AS INT)
1	null

Showing 1 row.

```
df.createOrReplaceTempView("hi")
```

```
%sql
DESCRIBE EXTENDED hi
```

Table

	col_name	data_type	comment
1	name	string	null
2	dob_year	string	null
3	gender	string	null
4	salary	bigint	null

Showing all 4 rows.

```
%sql
SELECT try_cast(name AS INT) from hi
```

Table

	TRY_CAST(name AS INT)
1	null
2	null
3	null
4	null
5	null

Showing all 5 rows.

```
from pyspark.sql.functions import expr,concat_ws
data=[("James","Bond"),("Scott","Varsa")]
df=spark.createDataFrame(data).toDF("col1","col2")
df.withColumn("Name",expr(" col1 ||','|| col2")).show()
```

```
+-----+-----+-----------+
| col1| col2|       Name|
+-----+-----+-----------+
|James| Bond| James,Bond|
|Scott|Varsa|Scott,Varsa|
+-----+-----+-----------+
```

```
df.withColumn("Name",concat_ws(' ',df.col1,df.col2)).show()
```

```
+-----+-----+-----------+
| col1| col2|       Name|
+-----+-----+-----------+
|James| Bond| James Bond|
|Scott|Varsa|Scott Varsa|
+-----+-----+-----------+
```

```
from pyspark.sql.functions import expr
data = [("James","M"),("Michael","F"),("Jen","")]
columns = ["name","gender"]
df = spark.createDataFrame(data = data, schema = columns)

#Using CASE WHEN similar to SQL.
from pyspark.sql.functions import expr
df2=df.withColumn("gender", expr("CASE WHEN gender = 'M' THEN 'Male' " +
           "WHEN gender = 'F' THEN 'Female' ELSE 'unknown' END"))
df2.show()
```

```
+-------+-------+
|   name| gender|
+-------+-------+
|  James|   Male|
|Michael| Female|
|    Jen|unknown|
+-------+-------+
```

```
from pyspark.sql.functions import expr
data=[("2019-01-23",1),("2019-06-24",2),("2019-09-20",3)]
df=spark.createDataFrame(data).toDF("date","increment")

#Add Month value from another column
df.select(df.date,df.increment,
     expr("add_months(date,increment)")
  .alias("inc_date")).show()
```

```
+----------+---------+----------+
|      date|increment|  inc_date|
+----------+---------+----------+
|2019-01-23|        1|2019-02-23|
|2019-06-24|        2|2019-08-24|
|2019-09-20|        3|2019-12-20|
+----------+---------+----------+
```

```
from pyspark.sql.functions import expr
df.select(df.date,df.increment,
     expr("""add_months(date,increment) as inc_date""")
  ).show()
```

```
+----------+---------+----------+
|      date|increment|  inc_date|
```

```
+----------+---------+----------+
|2019-01-23|        1|2019-02-23|
|2019-06-24|        2|2019-08-24|
|2019-09-20|        3|2019-12-20|
+----------+---------+----------+
```

```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
data = [("111",50000),("222",60000),("333",40000)]
columns= ["EmpId","Salary"]
df = spark.createDataFrame(data = data, schema = columns)
```

```
df2 = df.select(col("EmpId"),col("Salary"),lit("1").alias("lit_value1"))
df2.show(truncate=False)
```

```
+-----+------+----------+
|EmpId|Salary|lit_value1|
+-----+------+----------+
|111  |50000 |1         |
|222  |60000 |1         |
|333  |40000 |1         |
+-----+------+----------+
```

```
from pyspark.sql.functions import when,col,lit

df3 = df2.withColumn("lit_value2", when((col("Salary") >=40000) &
(col("Salary") <= 50000),lit("100")).otherwise(lit("200")))
df3.show(truncate=False)
```

```
+-----+------+----------+----------+
|EmpId|Salary|lit_value1|lit_value2|
+-----+------+----------+----------+
|111  |50000 |1         |100       |
|222  |60000 |1         |200       |
|333  |40000 |1         |100       |
+-----+------+----------+----------+
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
         .appName('SparkByExamples.com') \
         .getOrCreate()

data = [("James, A, Smith","2018","M",3000),
            ("Michael, Rose, Jones","2010","M",4000),
            ("Robert,K,Williams","2010","M",4000),
            ("Maria,Anne,Jones","2005","F",4000),
            ("Jen,Mary,Brown","2010","",-1)
            ]

columns=["name","dob_year","gender","salary"]
df=spark.createDataFrame(data,columns)
df.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- dob_year: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)
```

```
from pyspark.sql.functions import split, col
df2 = df.select(split(col("name"),",").alias("NameArray")) \
    .drop("name")
df2.printSchema()
df2.show()
```

```
root
 |-- NameArray: array (nullable = true)
 |    |-- element: string (containsNull = false)

+--------------------+
|           NameArray|
+--------------------+
| [James,  A,  Smith]|
|[Michael,  Rose, ...|
|[Robert, K, Willi...|
|[Maria, Anne, Jones]|
|  [Jen, Mary, Brown]|
+--------------------+
```

```
df.createOrReplaceTempView("PERSON")
spark.sql("select SPLIT(name,',') as NameArray from PERSON") \
    .show()
```

```
+--------------------+
|           NameArray|
+--------------------+
| [James,  A,  Smith]|
|[Michael,  Rose, ...|
|[Robert, K, Willi...|
|[Maria, Anne, Jones]|
|  [Jen, Mary, Brown]|
+--------------------+
```

```
df.show()
```

```
+--------------------+--------+------+------+
|                name|dob_year|gender|salary|
+--------------------+--------+------+------+
|     James, A, Smith|    2018|     M|  3000|
|Michael, Rose, Jones|    2010|     M|  4000|
|   Robert,K,Williams|    2010|     M|  4000|
|    Maria,Anne,Jones|    2005|     F|  4000|
|      Jen,Mary,Brown|    2010|      |    -1|
+--------------------+--------+------+------+
```

```
df2 = df.withColumn("FirstName",split(col("name"),',').getItem(0))\
        .withColumn("MiddleName",split(col("name"),',').getItem(1))\
        .withColumn("LastName",split(col("name"),',').getItem(2)).drop("name")
df2.show()
```

```
+--------+------+------+---------+----------+--------+
|dob_year|gender|salary|FirstName|MiddleName|LastName|
+--------+------+------+---------+----------+--------+
|    2018|     M|  3000|    James|         A|   Smith|
|    2010|     M|  4000|  Michael|      Rose|   Jones|
|    2010|     M|  4000|   Robert|         K|Williams|
|    2005|     F|  4000|    Maria|      Anne|   Jones|
|    2010|      |    -1|      Jen|      Mary|   Brown|
+--------+------+------+---------+----------+--------+
```

```
spark = SparkSession.builder.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)
```

```
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          |
+----------------+------------------+------------+
```

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

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

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

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

```
from pyspark.sql.functions import substring
data = [(1,"20200828"),(2,"20180525")]
columns=["id","date"]
df=spark.createDataFrame(data,columns)
df.withColumn('year', substring('date', 1,4))\
    .withColumn('month', substring('date', 5,2))\
    .withColumn('day', substring('date', 7,2)).show()
```

```
+---+--------+----+-----+---+
| id|    date|year|month|day|
+---+--------+----+-----+---+
|  1|20200828|2020|   08| 28|
|  2|20180525|2018|   05| 25|
+---+--------+----+-----+---+
```

```
df.select('date', substring('date', 1,4).alias('year'), \
                  substring('date', 5,2).alias('month'), \
                  substring('date', 7,2).alias('day'))
```

```
Out[27]: DataFrame[date: string, year: string, month: string, day: string]
```

```
from pyspark.sql import SparkSession
spark =
SparkSession.builder.master("local[1]").appName("SparkByExamples.com").getOrCre
ate()
address = [(1,"14851 Jeffrey Rd","DE"),
    (2,"43421 Margarita St","NY"),
    (3,"13111 Siemon Ave","CA")]
df =spark.createDataFrame(address,["id","address","state"])
df.show()
```

```
+---+------------------+-----+
| id|           address|state|
+---+------------------+-----+
|  1|  14851 Jeffrey Rd|   DE|
|  2|43421 Margarita St|   NY|
|  3|  13111 Siemon Ave|   CA|
+---+------------------+-----+
```

```
from pyspark.sql.functions import regexp_replace
df.withColumn('address', regexp_replace('address', 'Rd', 'Road')) \
  .show(truncate=False)
```

```
+---+------------------+-----+
|id |address           |state|
+---+------------------+-----+
|1  |14851 Jeffrey Road|DE   |
|2  |43421 Margarita St|NY   |
|3  |13111 Siemon Ave  |CA   |
+---+------------------+-----+
```

```
from pyspark.sql.functions import when
df.withColumn('address',
    when(df.address.endswith('Rd'),regexp_replace(df.address,'Rd','Road')) \
   .when(df.address.endswith('St'),regexp_replace(df.address,'St','Street')) \
   .when(df.address.endswith('Ave'),regexp_replace(df.address,'Ave','Avenue'))
\
   .otherwise(df.address)) \
   .show(truncate=False)
```

```
+---+----------------------+-----+
|id |address               |state|
+---+----------------------+-----+
|1  |14851 Jeffrey Road    |DE   |
```

```
|2  |43421 Margarita Street|NY   |
|3  |13111 Siemon Avenue   |CA   |
+---+----------------------+-----+
```

```
#Using translate to replace character by character
from pyspark.sql.functions import translate
df.withColumn('address', translate('address', '123', 'ABC')) \
  .show(truncate=False)
```

```
+---+------------------+-----+
|id |address           |state|
+---+------------------+-----+
|1  |A485A Jeffrey Rd  |DE   |
|2  |4C4BA Margarita St|NY   |
|3  |ACAAA Siemon Ave  |CA   |
+---+------------------+-----+
```

```
from pyspark.sql.functions import *

df=spark.createDataFrame(
        data = [ ("1","2019-06-24 12:01:19.000")],
        schema=["id","input_timestamp"])
df.printSchema()

#Timestamp String to DateType
df.withColumn("timestamp",to_timestamp("input_timestamp")) \
  .show(truncate=False)

# Using Cast to convert TimestampType to DateType
df.withColumn('timestamp_string', \
         to_timestamp('timestamp').cast('string')) \
  .show(truncate=False)
```

```
df = spark.createDataFrame([[]])
```

```
df.display()
```

```
from pyspark.sql.functions import lit
colObj = lit("sparkbyexamples.com")
```

```
data=[("James",23),("Ann",40)]
df=spark.createDataFrame(data).toDF("name.fname","gender")
df.printSchema()
```

```
root
 |-- name.fname: string (nullable = true)
 |-- gender: long (nullable = true)
```

```
df.select(df.gender).show()
df.select(df["gender"]).show()
```

```
+------+
|gender|
+------+
|    23|
|    40|
+------+
```

```
+------+
|gender|
+------+
|    23|
|    40|
+------+
```

```
df.select(df["`name.fname`"]).show()
```

```
+----------+
|name.fname|
+----------+
|     James|
|       Ann|
+----------+
```

```
data=[("James","Bond","100",None),
      ("Ann","Varsa","200",'F'),
      ("Tom Cruise","XXX","400",''),
      ("Tom Brand",None,"400",'M')]
columns=["fname","lname","id","gender"]
df=spark.createDataFrame(data,columns)
```

```
#alias
from pyspark.sql.functions import col
df.select(col("fname").alias("first_name"), \
          df.lname.alias("last_name")
   ).show()
```

```
+----------+---------+
|first_name|last_name|
+----------+---------+
|     James|     Bond|
|       Ann|    Varsa|
|Tom Cruise|      XXX|
| Tom Brand|     null|
+----------+---------+
```

```
#Another example
from pyspark.sql.functions import expr
df.select(expr(" fname ||','|| lname").alias("fullName") \
   ).show()
```

```
+--------------+
|      fullName|
+--------------+
|    James,Bond|
|     Ann,Varsa|
|Tom Cruise,XXX|
|          null|
+--------------+
```

```
#asc, desc to sort ascending and descending order repsectively.
df.sort(df.fname.asc()).show()
df.sort(df.fname.desc()).show()
```

```
+----------+-----+---+------+
|     fname|lname| id|gender|
+----------+-----+---+------+
|       Ann|Varsa|200|     F|
|     James| Bond|100|  null|
| Tom Brand| null|400|     M|
|Tom Cruise|  XXX|400|      |
+----------+-----+---+------+
```

```
+----------+-----+---+------+
|     fname|lname| id|gender|
+----------+-----+---+------+
|Tom Cruise|  XXX|400|      |
| Tom Brand| null|400|     M|
|     James| Bond|100|  null|
|       Ann|Varsa|200|     F|
+----------+-----+---+------+
```

```
#contains
df.filter(df.fname.contains("T")).show()
```

```
+----------+-----+---+------+
|     fname|lname| id|gender|
+----------+-----+---+------+
|Tom Cruise|  XXX|400|      |
| Tom Brand| null|400|     M|
+----------+-----+---+------+
```

```
#startswith, endswith()
df.filter(df.fname.startswith("T")).show()
df.filter(df.fname.endswith("Cruise")).show()
```

```
+----------+-----+---+------+
|     fname|lname| id|gender|
+----------+-----+---+------+
|Tom Cruise|  XXX|400|      |
| Tom Brand| null|400|     M|
+----------+-----+---+------+

+----------+-----+---+------+
|     fname|lname| id|gender|
+----------+-----+---+------+
|Tom Cruise|  XXX|400|      |
+----------+-----+---+------+
```

```
#isNull & isNotNull
df.filter(df.lname.isNull()).show()
df.filter(df.lname.isNotNull()).show()
```

```
+---------+-----+---+------+
|    fname|lname| id|gender|
+---------+-----+---+------+
|Tom Brand| null|400|     M|
+---------+-----+---+------+

+----------+-----+---+------+
|     fname|lname| id|gender|
+----------+-----+---+------+
|     James| Bond|100|  null|
|       Ann|Varsa|200|     F|
|Tom Cruise|  XXX|400|      |
+----------+-----+---+------+
```

```
df.select(df.fname.substr(1,4).alias("substr")).show()
```

```
+------+
|substr|
+------+
|  Jame|
|   Ann|
|  Tom |
|  Tom |
+------+
```

```
from pyspark.sql.functions import when
df.select('fname','lname','gender', when(df.gender == 'M','Male')\
                           .when(df.gender == 'F','Female')\
                           .when(df.gender == None,"")\
                           .otherwise(df.gender).alias("new_gender")).show()
```

```
+----------+-----+------+----------+
|     fname|lname|gender|new_gender|
+----------+-----+------+----------+
|     James| Bond|  null|      null|
|       Ann|Varsa|     F|    Female|
|Tom Cruise|  XXX|      |          |
| Tom Brand| null|     M|      Male|
+----------+-----+------+----------+
```

```
#isin
li=["100","200"]
df.select(df.fname,df.lname,df.id) \
  .filter(df.id.isin(li)) \
  .show()
```

```
+-----+-----+---+
|fname|lname| id|
+-----+-----+---+
|James| Bond|100|
|  Ann|Varsa|200|
+-----+-----+---+
```

```
from pyspark.sql.types import
StructType,StructField,StringType,ArrayType,MapType
data=[(("James","Bond"),["Java","C#"],{'hair':'black','eye':'brown'}),
      (("Ann","Varsa"),[".NET","Python"],{'hair':'brown','eye':'black'}),
      (("Tom Cruise",""),["Python","Scala"],{'hair':'red','eye':'grey'}),
      (("Tom Brand",None),["Perl","Ruby"],{'hair':'black','eye':'blue'})]

schema = StructType([
        StructField('name', StructType([
            StructField('fname', StringType(), True),
            StructField('lname', StringType(), True)])),
        StructField('languages', ArrayType(StringType()),True),
        StructField('properties', MapType(StringType(),StringType()),True)
     ])
df=spark.createDataFrame(data,schema)
df.printSchema()
```

```
root
 |-- name: struct (nullable = true)
 |    |-- fname: string (nullable = true)
 |    |-- lname: string (nullable = true)
 |-- languages: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- properties: map (nullable = true)
 |    |-- key: string
 |    |-- value: string (valueContainsNull = true)
```

```
display(df)
```

Table

	name	languages	properties
1	{"fname": "James", "lname": "Bond"}	["Java", "C#"]	{"eye": "brown", "hair": "black"}
2	{"fname": "Ann", "lname": "Varsa"}	[".NET", "Python"]	{"eye": "black", "hair": "brown"}
	{"fname": "Tom Cruise", "lname":	["Python"	{"eye": "grey", "hair": "red"}

Showing all 4 rows.

```
#getField from MapType
df.select(df.properties.getField("hair")).show()
```

```
+----------------+
|properties[hair]|
+----------------+
|           black|
|           brown|
|             red|
|           black|
+----------------+
```

```
from pyspark.sql import Row
row=Row("James",40)
print(row[0] +","+str(row[1]))
```

```
James,40
```

```
Alice
```