```
# Databricks notebook source


from pyspark.sql import SparkSession
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']))

# COMMAND ----------
```

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql.functions import to_timestamp, current_timestamp
from pyspark.sql.types import StructType, StructField, StringType,
IntegerType, LongType

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

schema = StructType([
            StructField("seq", StringType(), True)])

dates = ['1']

rdd=sc.parallelize([dates])

df = spark.createDataFrame(rdd, schema=schema)

df.show()

# COMMAND ----------

import pandas as pd
data = [['Scott', 50], ['Jeff', 45], ['Thomas', 54],['Ann',34]]

# Create the pandas DataFrame
pandasDF = pd.DataFrame(data, columns = ['Name', 'Age'])

# print dataframe.
print(pandasDF)

from pyspark.sql import SparkSession

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

sparkDF=spark.createDataFrame(pandasDF)
sparkDF.printSchema()
sparkDF.show()

#sparkDF=spark.createDataFrame(pandasDF.astype(str))
from pyspark.sql.types import StructType,StructField, StringType,
IntegerType
mySchema = StructType([ StructField("First Name", StringType(), True)\
                       ,StructField("Age", IntegerType(), True)])
```

```
sparkDF2 = spark.createDataFrame(pandasDF,schema=mySchema)
sparkDF2.printSchema()
sparkDF2.show()


spark.conf.set("spark.sql.execution.arrow.enabled","true")
spark.conf.set("spark.sql.execution.arrow.pyspark.fallback.enabled","t
rue")

pandasDF2=sparkDF2.select("*").toPandas
print(pandasDF2)


test=spark.conf.get("spark.sql.execution.arrow.enabled")
print(test)

test123=spark.conf.get("spark.sql.execution.arrow.pyspark.fallback.ena
bled")
print(test123)

# COMMAND ----------

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

from pyspark.sql.functions import col,expr
data=[("2019-01-23",1),("2019-06-24",2),("2019-09-20",3)]
spark.createDataFrame(data).toDF("date","increment") \
    .select(col("date"),col("increment"), \
      expr("add_months(to_date(date,'yyyy-MM-dd'),cast(increment as 
int))").alias("inc_date")) \
    .show()

# COMMAND ----------

from pyspark.sql import SparkSession

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

data = [('James','Smith','M',3000),
  ('Anna','Rose','F',4100),
  ('Robert','Williams','M',6200),
]

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

```
if 'salary1' not in df.columns:
    print("aa")

# Add new constanct column
from pyspark.sql.functions import lit
df.withColumn("bonus_percent", lit(0.3)) \
  .show()

#Add column from existing column
df.withColumn("bonus_amount", df.salary*0.3) \
  .show()

#Add column by concatinating existing columns
from pyspark.sql.functions import concat_ws
df.withColumn("name", concat_ws(",","firstname",'lastname')) \
  .show()

#Add current date
from pyspark.sql.functions import current_date
df.withColumn("current_date", current_date()) \
  .show()


from pyspark.sql.functions import when
df.withColumn("grade", \
   when((df.salary < 4000), lit("A")) \
     .when((df.salary >= 4000) & (df.salary <= 5000), lit("B")) \
     .otherwise(lit("C")) \
  ).show()

# Add column using select
df.select("firstname","salary", lit(0.3).alias("bonus")).show()
df.select("firstname","salary", lit(df.salary *
0.3).alias("bonus_amount")).show()
df.select("firstname","salary",
current_date().alias("today_date")).show()

#Add columns using SQL
df.createOrReplaceTempView("PER")
spark.sql("select firstname,salary, '0.3' as bonus from PER").show()
spark.sql("select firstname,salary, salary * 0.3 as bonus_amount from
PER").show()
spark.sql("select firstname,salary, current_date() as today_date from
PER").show()
spark.sql("select firstname,salary, " +
          "case salary when salary < 4000 then 'A' "+
          "else 'B' END as grade from PER").show()
```

```
# COMMAND ----------

import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import approx_count_distinct,collect_list
from pyspark.sql.functions import
collect_set,sum,avg,max,countDistinct,count
from pyspark.sql.functions import first, last, kurtosis, min, mean,
skewness
from pyspark.sql.functions import stddev, stddev_samp, stddev_pop,
sumDistinct
from pyspark.sql.functions import variance,var_samp,  var_pop

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

simpleData = [("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)
  ]
schema = ["employee_name", "department", "salary"]


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

print("approx_count_distinct: " + \
      str(df.select(approx_count_distinct("salary")).collect()[0][0]))

print("avg: " + str(df.select(avg("salary")).collect()[0][0]))

df.select(collect_list("salary")).show(truncate=False)

df.select(collect_set("salary")).show(truncate=False)

df2 = df.select(countDistinct("department", "salary"))
df2.show(truncate=False)
print("Distinct Count of Department &amp; Salary: "+str(df2.collect()
[0][0]))
```

```
print("count: "+str(df.select(count("salary")).collect()[0]))
df.select(first("salary")).show(truncate=False)
df.select(last("salary")).show(truncate=False)
df.select(kurtosis("salary")).show(truncate=False)
df.select(max("salary")).show(truncate=False)
df.select(min("salary")).show(truncate=False)
df.select(mean("salary")).show(truncate=False)
df.select(skewness("salary")).show(truncate=False)
df.select(stddev("salary"), stddev_samp("salary"), \
    stddev_pop("salary")).show(truncate=False)
df.select(sum("salary")).show(truncate=False)
df.select(sumDistinct("salary")).show(truncate=False)
df.select(variance("salary"),var_samp("salary"),var_pop("salary")) \
  .show(truncate=False)

# COMMAND ----------

import pyspark
from pyspark.sql import SparkSession
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)

# COMMAND ----------


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

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


arrayCol = ArrayType(StringType(),False)

data = [
 ("James,,Smith",["Java","Scala","C++"],["Spark","Java"],"OH","CA"),
 ("Michael,Rose,",["Spark","Java","C++"],["Spark","Java"],"NY","NJ"),
 ("Robert,,Williams",["CSharp","VB"],["Spark","Python"],"UT","NV")
]

schema = StructType([
    StructField("name",StringType(),True),
    StructField("languagesAtSchool",ArrayType(StringType()),True),
    StructField("languagesAtWork",ArrayType(StringType()),True),
    StructField("currentState", StringType(), True),
    StructField("previousState", StringType(), True)
  ])

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

from pyspark.sql.functions import explode
df.select(df.name,explode(df.languagesAtSchool)).show()

from pyspark.sql.functions import split
df.select(split(df.name,",").alias("nameAsArray")).show()

from pyspark.sql.functions import array
df.select(df.name,array(df.currentState,df.previousState).alias("State
s")).show()

from pyspark.sql.functions import array_contains
df.select(df.name,array_contains(df.languagesAtSchool,"Java")
    .alias("array_contains")).show()

# COMMAND ----------

import pyspark
from pyspark.sql import SparkSession


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

states = {"NY":"New York", "CA":"California", "FL":"Florida"}
broadcastStates = spark.sparkContext.broadcast(states)
```

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

def state_convert(code):
    return broadcastStates.value[code]

result = df.rdd.map(lambda x:
(x[0],x[1],x[2],state_convert(x[3]))).toDF(columns)
result.show(truncate=False)

# Broadcast variable on filter

filteDf =
df.where(df['state'].isin(list(broadcastStates.value.keys())))
filteDf.show(truncate=False)

# COMMAND ----------

import pyspark
from pyspark.sql import SparkSession

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

simpleData = [("James",34,"2006-01-01","true","M",3000.60),
    ("Michael",33,"1980-01-10","true","F",3300.80),
    ("Robert",37,"06-01-1992","false","M",5000.50)
  ]

columns =
["firstname","age","jobStartDate","isGraduated","gender","salary"]
df = spark.createDataFrame(data = simpleData, schema = columns)
df.printSchema()
df.show(truncate=False)

from pyspark.sql.functions import col
from pyspark.sql.types import StringType,BooleanType,DateType
df2 = df.withColumn("age",col("age").cast(StringType())) \
    .withColumn("isGraduated",col("isGraduated").cast(BooleanType()))
\
    .withColumn("jobStartDate",col("jobStartDate").cast(DateType()))
```

```
df2.printSchema()

df3 = df2.selectExpr("cast(age as int) age",
    "cast(isGraduated as string) isGraduated",
    "cast(jobStartDate as string) jobStartDate")
df3.printSchema()
df3.show(truncate=False)

df3.createOrReplaceTempView("CastExample")
df4 = spark.sql("SELECT
STRING(age),BOOLEAN(isGraduated),DATE(jobStartDate) from CastExample")
df4.printSchema()
df4.show(truncate=False)

# COMMAND ----------

from pyspark.sql import SparkSession
from pyspark.sql.types import DoubleType

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

simpleData = [("James","34","true","M","3000.6089"),
    ("Michael","33","true","F","3300.8067"),
    ("Robert","37","false","M","5000.5034")
]

columns = ["firstname","age","isGraduated","gender","salary"]
df = spark.createDataFrame(data=simpleData, schema=columns)
df.printSchema()
df.show(truncate=False)

from pyspark.sql.functions import col, round, expr
df.withColumn("salary", df.salary.cast(DoubleType())).printSchema()
df.withColumn("salary",
col("salary").cast(DoubleType())).printSchema()

df.selectExpr("firstname", "isGraduated", "cast(salary as double) as
salary").printSchema()

df.createOrReplaceTempView("CastExample")
spark.sql("SELECT firstname, isGraduated, CAST(salary AS double) AS
salary FROM CastExample").printSchema()

# COMMAND ----------

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

dataCollect = deptDF.collect()

print(dataCollect)

dataCollect2 = deptDF.select("dept_name").collect()
print(dataCollect2)

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


# COMMAND ----------

from pyspark.sql import SparkSession
from pyspark.sql.types import DoubleType, IntegerType
from pyspark.sql.functions import col, round

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

simpleData = [("James", "34", "true", "M", "3000.6089"),
              ("Michael", "33", "true", "F", "3300.8067"),
              ("Robert", "37", "false", "M", "5000.5034")]

columns = ["firstname", "age", "isGraduated", "gender", "salary"]
df = spark.createDataFrame(data=simpleData, schema=columns)

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

df = df.withColumn("salary", df.salary.cast(DoubleType()))
df.printSchema()

df = df.withColumn("salary", col("salary").cast(DoubleType()))
```

```
df.printSchema()

df = df.withColumn("salary", round(df.salary, 2))
df.show(truncate=False)
df.printSchema()

df.select("firstname", "isGraduated",
col("salary").alias("salary")).printSchema()

df.createOrReplaceTempView("CastExample")
spark.sql("SELECT firstname, isGraduated, CAST(salary AS DOUBLE) AS
salary FROM CastExample").printSchema()

df.select("firstname", col("age").expr, "isGraduated",
col("salary").cast("float").alias("salary")).show()


# COMMAND ----------

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

dataCollect = deptDF.collect()

print(dataCollect)

dataCollect2 = deptDF.select("dept_name").collect()
print(dataCollect2)

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


# COMMAND ----------

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

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 expr
df.select(df.fname.alias("first_name"), \
          df.lname.alias("last_name"), \
          expr(" fname ||','|| lname").alias("fullName") \
   ).show()

#asc, desc
df.sort(df.fname.asc()).show()
df.sort(df.fname.desc()).show()

#cast
df.select(df.fname,df.id.cast("int")).printSchema()

#between
df.filter(df.id.between(100,300)).show()

#contains
df.filter(df.fname.contains("Cruise")).show()

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

#eqNullSafe

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

#like , rlike
df.select(df.fname,df.lname,df.id) \
  .filter(df.fname.like("%om"))

#over

#substr
df.select(df.fname.substr(1,2).alias("substr")).show()

#when & otherwise
from pyspark.sql.functions import when
```

```
df.select(df.fname,df.lname,when(df.gender=="M","Male") \
              .when(df.gender=="F","Female") \
              .when(df.gender==None ,"") \
              .otherwise(df.gender).alias("new_gender") \
    ).show()

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

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()
#getItem()
df.select(df.languages.getItem(1)).show()

df.select(df.properties.getItem("hair")).show()

#getField from Struct or Map
df.select(df.properties.getField("hair")).show()

df.select(df.name.getField("fname")).show()

#dropFields
#from pyspark.sql.functions import col
#df.withColumn("name1",col("name").dropFields(["fname"])).show()

#withField
#from pyspark.sql.functions import lit
#df.withColumn("name",df.name.withField("fname",lit("AA"))).show()
```

```
#from pyspark.sql import Row
#from pyspark.sql.functions import lit
#df = spark.createDataFrame([Row(a=Row(b=1, c=2))])
#df.withColumn('a', df['a'].withField('b',
lit(3))).select('a.b').show()

#from pyspark.sql import Row
#from pyspark.sql.functions import col, lit
#df = spark.createDataFrame([
#Row(a=Row(b=1, c=2, d=3, e=Row(f=4, g=5, h=6)))])
#df.withColumn('a', df['a'].dropFields('b')).show()


# COMMAND ----------

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

dataDictionary = [
        ('James',{'hair':'black','eye':'brown'}),
        ('Michael',{'hair':'brown','eye':None}),
        ('Robert',{'hair':'red','eye':'black'}),
        ('Washington',{'hair':'grey','eye':'grey'}),
        ('Jefferson',{'hair':'brown','eye':''})
        ]

df = spark.createDataFrame(data=dataDictionary, schema =
['name','properties'])
df.printSchema()
df.show(truncate=False)

df3=df.rdd.map(lambda x: \
    (x.name,x.properties["hair"],x.properties["eye"])) \
    .toDF(["name","hair","eye"])
df3.printSchema()
df3.show()

df.withColumn("hair",df.properties.getItem("hair")) \
  .withColumn("eye",df.properties.getItem("eye")) \
  .drop("properties") \
  .show()

df.withColumn("hair",df.properties["hair"]) \
  .withColumn("eye",df.properties["eye"]) \
  .drop("properties") \
  .show()

# Functions
from pyspark.sql.functions import explode,map_keys,col
```

```
keysDF = df.select(explode(map_keys(df.properties))).distinct()
keysList = keysDF.rdd.map(lambda x:x[0]).collect()
keyCols = list(map(lambda x:
col("properties").getItem(x).alias(str(x)), keysList))
df.select(df.name, *keyCols).show()

# COMMAND ----------

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

spark =
SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
data = [ ("36636","Finance",3000,"USA"),
    ("40288","Finance",5000,"IND"),
    ("42114","Sales",3900,"USA"),
    ("39192","Marketing",2500,"CAN"),
    ("34534","Sales",6500,"USA") ]
schema = StructType([
     StructField('id', StringType(), True),
     StructField('dept', StringType(), True),
     StructField('salary', IntegerType(), True),
     StructField('location', StringType(), True)
     ])

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

#Convert scolumns to Map
from pyspark.sql.functions import col,lit,create_map
df = df.withColumn("propertiesMap",create_map(
        lit("salary"),col("salary"),
        lit("location"),col("location")
        )).drop("salary","location")
df.printSchema()
df.show(truncate=False)

# COMMAND ----------

from pyspark.sql import SparkSession
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 = ["Name","Dept","Salary"]
df = spark.createDataFrame(data=data,schema=columns)
df.distinct().show()
print("Distinct Count: " + str(df.distinct().count()))

# Using countDistrinct()
from pyspark.sql.functions import countDistinct
df2=df.select(countDistinct("Dept","Salary"))
df2.show()

print("Distinct Count of Department &amp; Salary: "+ str(df2.collect()
[0][0]))

df.createOrReplaceTempView("PERSON")
spark.sql("select distinct(count(*)) from PERSON").show()

# COMMAND ----------

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

dataDictionary = [
        ('James',{'hair':'black','eye':'brown'}),
        ('Michael',{'hair':'brown','eye':None}),
        ('Robert',{'hair':'red','eye':'black'}),
        ('Washington',{'hair':'grey','eye':'grey'}),
        ('Jefferson',{'hair':'brown','eye':''})
        ]

df = spark.createDataFrame(data=dataDictionary, schema =
['name','properties'])
df.printSchema()
df.show(truncate=False)

# Using StructType schema
from pyspark.sql.types import StructField, StructType, StringType,
MapType,IntegerType
schema = StructType([
    StructField('name', StringType(), True),
    StructField('properties', MapType(StringType(),StringType()),True)
])
df2 = spark.createDataFrame(data=dataDictionary, schema = schema)
```

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

df3=df.rdd.map(lambda x: \
    (x.name,x.properties["hair"],x.properties["eye"])) \
    .toDF(["name","hair","eye"])
df3.printSchema()
df3.show()

df.withColumn("hair",df.properties.getItem("hair")) \
  .withColumn("eye",df.properties.getItem("eye")) \
  .drop("properties") \
  .show()

df.withColumn("hair",df.properties["hair"]) \
  .withColumn("eye",df.properties["eye"]) \
  .drop("properties") \
  .show()

# Functions
from pyspark.sql.functions import explode,map_keys,col
keysDF = df.select(explode(map_keys(df.properties))).distinct()
keysList = keysDF.rdd.map(lambda x:x[0]).collect()
keyCols = list(map(lambda x:
col("properties").getItem(x).alias(str(x)), keysList))
df.select(df.name, *keyCols).show()

# COMMAND ----------

import pyspark
from pyspark.sql import SparkSession, Row
from pyspark.sql.types import StructType,StructField, StringType,
IntegerType
from pyspark.sql.functions import *

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

spark =
SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
rdd = spark.sparkContext.parallelize(data)

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

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

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

```
dfFromData2 = spark.createDataFrame(data).toDF(*columns)
dfFromData2.printSchema()

rowData = map(lambda x: Row(*x), data)
dfFromData3 = spark.createDataFrame(rowData,columns)
dfFromData3.printSchema()

# COMMAND ----------

import pyspark
from pyspark.sql import SparkSession, Row
from pyspark.sql.types import StructType, StructField, StringType

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

# Using List
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)

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)

# Using list of Row type
dept2 = [Row("Finance", 10),
         Row("Marketing", 20),
         Row("Sales", 30),
         Row("IT", 40)
       ]

deptDF2 = spark.createDataFrame(data=dept2, schema=deptColumns)
deptDF2.printSchema()
deptDF2.show(truncate=False)

# Convert list to RDD
```

```
rdd = spark.sparkContext.parallelize(dept)


# COMMAND ----------

from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder \
               .appName('SparkByExamples.com') \
               .getOrCreate()
data=[["1"]]
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 ss SSS")) \
  .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_timestamp") \
     .show(truncate=False)

# COMMAND ----------


from pyspark.sql import SparkSession
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)


# COMMAND ----------
```