**Glue - Laboratory**

Example - 1

|  |
| --- |
| 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() |

Example – 2

|  |
| --- |
| 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) |

Example – 3

|  |
| --- |
| spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()  data = [("James","Smith","USA","CA"),  ("Michael","Rose","USA","NY"),  ("Robert","Williams","USA","CA"),  ("Maria","Jones","USA","FL")  ]  columns = ["firstname","lastname","country","state"]  df = spark.createDataFrame(data = data, schema = columns)  df.show(truncate=False) |

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

|  |
| --- |
| # Select All columns from List  df.select(\*columns).show()  # Select All columns  df.select([col for col in df.columns]).show()  df.select("\*").show() |

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

Example – 4

|  |
| --- |
| data = [  (("James",None,"Smith"),"OH","M"),  (("Anna","Rose",""),"NY","F"),  (("Julia","","Williams"),"OH","F"),  (("Maria","Anne","Jones"),"NY","M"),  (("Jen","Mary","Brown"),"NY","M"),  (("Mike","Mary","Williams"),"OH","M")  ]  from pyspark.sql.types import StructType,StructField, StringType  schema = StructType([  StructField('name', StructType([  StructField('firstname', StringType(), True),  StructField('middlename', StringType(), True),  StructField('lastname', StringType(), True)  ])),  StructField('state', StringType(), True),  StructField('gender', StringType(), True)  ])  df2 = spark.createDataFrame(data = data, schema = schema)  df2.printSchema()  df2.show(truncate=False) # shows all columns |

|  |
| --- |
| df2.select("name").show(truncate=False)  df2.select("name.firstname","name.lastname").show(truncate=False)  df2.select("name.\*").show(truncate=False) |

Example – 5

|  |
| --- |
| # Prepare Data  simpleData = (("Java",4000,5), \  ("Python", 4600,10), \  ("Scala", 4100,15), \  ("Scala", 4500,15), \  ("PHP", 3000,20), \  )  columns= ["CourseName", "fee", "discount"]  # Create DataFrame  df = spark.createDataFrame(data = simpleData, schema = columns)  df.printSchema()  df.show(truncate=False) |

|  |
| --- |
| # Custom transformation 1  from pyspark.sql.functions import upper  def to\_upper\_str\_columns(df):  return df.withColumn("CourseName",upper(df.CourseName))  # Custom transformation 2  def apply\_discount(df):  return df.withColumn("discounted\_fee", \  df.new\_fee - (df.new\_fee \* df.discount) / 100) |

|  |
| --- |
| # PySpark transform() Usage  df2 = df.transform(to\_upper\_str\_columns) \  .transform(apply\_discount) |

Example – 6

|  |
| --- |
| 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) |

|  |
| --- |
| empDF.join(deptDF,empDF.emp\_dept\_id == deptDF.dept\_id,"inner") \  .show(truncate=False) |

|  |
| --- |
| 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) |

|  |
| --- |
| empDF.join(deptDF,empDF.emp\_dept\_id == deptDF.dept\_id,"right") \  .show(truncate=False)  empDF.join(deptDF,empDF.emp\_dept\_id == deptDF.dept\_id,"rightouter") \  .show(truncate=False) |

|  |
| --- |
| empDF.join(deptDF,empDF.emp\_dept\_id == deptDF.dept\_id,"leftsemi") \  .show(truncate=False) |

|  |
| --- |
| empDF.createOrReplaceTempView("EMP")  deptDF.createOrReplaceTempView("DEPT")  joinDF = spark.sql("select \* from EMP e, DEPT d where e.emp\_dept\_id == d.dept\_id") \  .show(truncate=False)  joinDF2 = spark.sql("select \* from EMP e INNER JOIN DEPT d ON e.emp\_dept\_id == d.dept\_id") \  .show(truncate=False) |

Example – 7

|  |
| --- |
| # Lambda example with single argument  square = lambda x: x \* x  print(square(4)) |

|  |
| --- |
| # Lambda function using if-else  min = lambda a, b : a if(a < b) else b  print(min(10, 20)) |

|  |
| --- |
| # Lambda with map()  numbers = [2, 4, 5, 6, 3]  squared\_result = list(map(lambda x: x\*\*2, numbers))  print(squared\_result) |

Example - 8

|  |
| --- |
| **def** addition(n):  **return** n **+** n    # We double all numbers using map()  numbers **=** (1, 2, 3, 4)  result **=** map(addition, numbers)  print(list(result)) |

|  |
| --- |
| numbers **=** (1, 2, 3, 4)  result **=** map(**lambda** x: x **+** x, numbers)  print(list(result)) |

|  |
| --- |
| numbers1 **=** [1, 2, 3]  numbers2 **=** [4, 5, 6]    result **=** map(**lambda** x, y: x **+** y, numbers1, numbers2)  print(list(result)) |

|  |
| --- |
| **def** double\_even(num):  **if** num **%** 2 **==** 0:  **return** num **\*** 2  **else**:  **return** num    # Create a list of numbers to apply the function to  numbers **=** [1, 2, 3, 4, 5]    # Use map to apply the function to each element in the list  result **=** list(map(double\_even, numbers))    # Print the result  print(result)  # [1, 4, 3, 8, 5] |