# Lab sheet 3 - SparkSQL

Name: J Viswaksena RollNo:

AM.EN.U4AIE21035

## Q1. Write a UDF to convert a given text to upper case.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

spark = SparkSession.builder.appName("UpperCaseConversion").getOrCreate()

data = [("Hello",), ("World",), ("Spark",), ("UDF",)]
columns = ["text"]

df = spark.createDataFrame(data, columns)

def convert_to_upper_case(text):
    return text.upper()

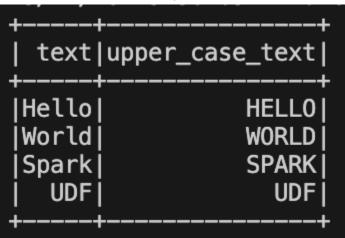
convert_to_upper_case_udf = udf(convert_to_upper_case, StringType())

df_upper_case = df.withColumn("upper_case_text", convert_to_upper_case_udf("text"))

df_upper_case.show()
```

### Output:

(base) viswaksenajayam@Viswaksenas-MacBook-Pro lab3 % spark-submit 1\_UpperCaseConversion.py



## Q2. Execute the following steps and show the output

```
# a. Import a Spark Session into Apache Spark.
from pyspark.sql import SparkSession

# b. Create a Spark Session 'spark' using the 'builder()' function.
spark = SparkSession.builder.appName("EmployeeData").getOrCreate()

# c. Import the Implicits class into 'spark' Session.
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

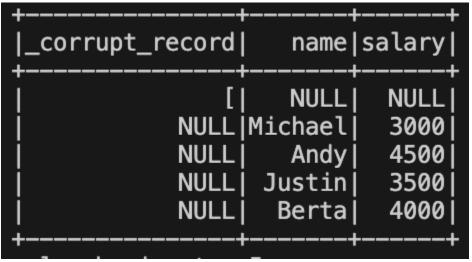
# d. Now create a DataFrame 'df' and import data from the 'employee.json' file.
file_path = "employee.json"  # Replace with the actual path to your 'employee.json'
file
df = spark.read.json(file_path)

# e. Print the schema of 'df' DataFrame.
df.printSchema()

# f. Display the DataFrame 'df'.
df.show(5)
```

#### **Output:**

(base) viswaksenajayam@Viswaksenas-MacBook-Pro lab3 % spark-submit 2\_EmployeeData.py



```
Import necessary classes and functions
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
spark = SparkSession.builder.appName("EmployeeData").getOrCreate()
file path = "employee1.json" # Replace with the actual path to your 'employee.json'
df = spark.read.json(file path)
df incremented age = df.withColumn("age", col("age") + 2)
df incremented age.show()
->OUTPUT
   _corrupt_record| age|
                                 name |
                    [ |NULL |
                                 NULL
                NULLI
                         30
                                 John
                NULLI
                         37
                                Alice
                         341
                NULLI
                                  Bob l
                         30 I
                NULLI
                                  Evel
                NULL
                         42|Charlie|
                    ] |NULL|
                                 NULLI
df_above_30 = df.filter(col("age") > 30)
df above 30.show()
->OUTPUT
  _corrupt_record|age|
                                 name I
                NULLI
                       35 I
                               Alice
                         32 I
                NULL
                                  Bob |
                NULL | 40 | Charlie |
```

```
the same.)
age_counts = df.groupBy("age").count()
age counts.show()
->OUTPUT
 +---+
 | age|count|
              11
     32 I
              2 j
 NULL
              2
     281
              11
     35 |
     40 l
df.createOrReplaceTempView("employee")
`sqlDF'.
sql_query = "SELECT * FROM employee"
sqlDF = spark.sql(sql_query)
sqlDF.show()
->OUTPUT
```

### Q4.

```
# Import necessary classes and functions
from pyspark.sql import SparkSession
from pyspark.sql import Row

# Create a Spark session
spark = SparkSession.builder.appName("EmployeeData").getOrCreate()

# a. Create a class 'Employee' to store name and age of an employee.
class Employee:
    def __init__(self, name, age):
        self.name = name
        self.age = age

# b. Assigning a Dataset 'caseClassDS' to store the record of Andrew.
andrew_record = Employee(name="Andrew", age=28)
caseClassDS = spark.createDataFrame([Row(name=andrew_record.name,
age=andrew_record.age)])

# c. Display the Dataset 'caseClassDS'.
print("Dataset 'caseClassDS':")
caseClassDS.show()
```

→Output: Dataset-1

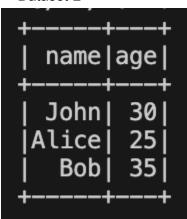


```
# d. Create a primitive Dataset to demonstrate mapping of DataFrames into Datasets.
data = [("John", 30), ("Alice", 25), ("Bob", 35)]
columns = ["name", "age"]
primitiveDS = spark.createDataFrame(data, columns)

# e. Assign the above sequence into an array.
array_of_datasets = [caseClassDS, primitiveDS]

# f. Display the result.
for index, dataset in enumerate(array_of_datasets, start=1):
    print(f"Dataset {index}:")
    dataset.show()
```

#### → Dataset-2



```
# Stop the Spark session
spark.stop()
```