

Lab sheet 3 - SparkSQL

Name: J Viswaksena
AM.EN.U4AIE21035

RollNo:

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

```
+-----+-----+
| text |upper_case_text|
+-----+-----+
|Hello|      HELLO|
|World|     WORLD|
|Spark|     SPARK|
|  UDF|       UDF|
+-----+-----+
```

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

_corrupt_record	name	salary
[NULL	NULL
NULL	Michael	3000
NULL	Andy	4500
NULL	Justin	3500
NULL	Berta	4000

Q3.

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

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

# Load data from 'employee.json' file into DataFrame 'df'
file_path = "employee1.json" # Replace with the actual path to your 'employee.json'
file
df = spark.read.json(file_path)

# a. Display the DataFrame after incrementing everyone's age by two years.
df_incremented_age = df.withColumn("age", col("age") + 2)
df_incremented_age.show()
```

→OUTPUT

_corrupt_record	age	name
[NULL	NULL	NULL
NULL	30	John
NULL	37	Alice
NULL	34	Bob
NULL	30	Eve
NULL	42	Charlie
] NULL	NULL	NULL

```
# b. Filter all the employees above age 30 and display the result.
df_above_30 = df.filter(col("age") > 30)
df_above_30.show()
```

→OUTPUT

_corrupt_record	age	name
NULL	35	Alice
NULL	32	Bob
NULL	40	Charlie

```
# c. Count the number of people with the same ages. (use the 'groupBy' function for the same.)
```

```
age_counts = df.groupBy("age").count()
```

```
age_counts.show()
```

→OUTPUT

age	count
32	1
NULL	2
28	2
35	1
40	1

```
# d. Creating a temporary view 'employee' of 'df' DataFrame.
```

```
df.createOrReplaceTempView("employee")
```

```
# e. Perform a 'select' operation on our 'employee' view to display the table into 'sqlDF'.
```

```
sql_query = "SELECT * FROM employee"
```

```
sqlDF = spark.sql(sql_query)
```

```
# f. Display the results of 'sqlDF'.
```

```
sqlDF.show()
```

→OUTPUT

name	age
Andrew	28

```
# 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

name	age
John	30
Alice	25
Bob	35

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