pySparkPrac3

```
data = [("Alice", 25), ("Bob", 30), ("Charlie", 35), ("David", 40), ("Eva", 45)]
columns=["name", "age"]
df = spark.createDataFrame(data, schema=columns)
```

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

```
df.withColumn("IsAdult", when(df.age >= 18, True).otherwise(False)).show()
df.withColumn("IsAdult", when(df.age >= 18, 'Yes').otherwise('No')).show()
```

```
+-------+---+-------+
|   name|age|IsAdult|
+-------+---+-------+
|  Alice| 25|   true|
|    Bob| 30|   true|
|Charlie| 35|   true|
|  David| 40|   true|
|    Eva| 45|   true|
+-------+---+-------+
+-------+---+-------+
|   name|age|IsAdult|
+-------+---+-------+
|  Alice| 25|    Yes|
|    Bob| 30|    Yes|
|Charlie| 35|    Yes|
|  David| 40|    Yes|
|    Eva| 45|    Yes|
+-------+---+-------+
```

simple Agg

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import count, sum, avg, min, max
aggFunction = df.groupBy().agg(count("*").alias("count"),sum("age").alias("total_age"),avg("age").alias("avg_age"),min("age").alias("min_age"),max("age").alias("max_age"))
result.show()
```

```
+-----+---------+-------+-------+-------+
|count|total_age|avg_age|min_age|max_age|
+-----+---------+-------+-------+-------+
|    5|      175|   35.0|     25|     45|
+-----+---------+-------+-------+-------+
```

udf for isAdult

```
def adult(age):
    if age>18:
        return 'yes'
    else:
        return 'No'
udfage=udf(lambda x: adult(x),StringType())
```

```
df.withColumn("isAdult",udfage(df.age)).show()
```

```
+-------+---+-------+
|   name|age|isAdult|
+-------+---+-------+
|  Alice| 25|    yes|
|    Bob| 30|    yes|
|Charlie| 35|    yes|
|  David| 40|    yes|
|    Eva| 45|    yes|
+-------+---+-------+
```

group AGG

```
data = [("Alice", "Sales", 5000),
        ("Bob", "IT", 4000),
        ("Charlie", "Sales", 6000),
        ("David", "IT", 4500),
        ("Eva", "IT", 5500),
        ("Frank", "Sales", 7000)]
df1 = spark.createDataFrame(data, ["name", "dept", "salary"])

groupAgg = df1.groupBy("dept").agg(sum("salary").alias("total_salary"),count("name").alias("peoples"))

groupAgg.show()
```

```
+-----+------------+-------+
| dept|total_salary|peoples|
+-----+------------+-------+
|Sales|       18000|      3|
|   IT|       14000|      3|
+-----+------------+-------+
```

```
from pyspark.sql.types import *
```

```
def issubstring(x,y):
    if y in x:
        return True
    else:
        return False
contains_udf = udf(lambda x, y: issubstring(x,y), BooleanType())
```

```
df = spark.createDataFrame([("hello world", "world"), ("pySpark is awesome", "java"), ("pyspark is powerful", "pyspark")], ["string", "substring"])
df.withColumn("contains_substring", contains_udf(df.string, df.substring)).show()
```

```
+------------------+---------+------------------+
|            string|substring|contains_substring|
+------------------+---------+------------------+
|       hello world|    world|              true|
| pySpark is awesome|     java|             false|
|pyspark is powerful|  pyspark|              true|
+------------------+---------+------------------+
```

```
from datetime import datetime

timestamp_string_udf = udf(lambda x: datetime.strftime(x, '%Y-%m-%d %H:%M:%S'), StringType())

df = spark.createDataFrame([(datetime(2022, 3, 8, 13, 45, 30),), (datetime(2023, 3, 8, 13, 45, 30),)], ["timestamp"])
df.withColumn("timestamp_string", timestamp_string_udf(df.timestamp)).printSchema()
```

pySparkPrac3

```
root
 |-- timestamp: timestamp (nullable = true)
 |-- timestamp_string: string (nullable = true)
```

```
Data=[("Ram", 28, "Sales", 3000),("Meena", 33, "Sales", 4600),("Robin", 40, "Sales", 4100),("Kunal", 25, "Finance", 3000),("Ram", 28, "Sales", 3000),("Srishti", 46, "Management", 3300
),("Jeny", 26, "Finance", 3900),("Hitesh", 30, "Marketing", 3000),("Kailash", 29, "Marketing", 2000),("Sharad", 39, "Sales", 4100)]

columns = ["Employee_Name", "Age",
           "Department", "Salary"]
```

```
df=spark.createDataFrame(data=Data,schema=columns)
df.show()
```

```
+-------------+---+----------+------+
|Employee_Name|Age|Department|Salary|
+-------------+---+----------+------+
|          Ram| 28|     Sales|  3000|
|        Meena| 33|     Sales|  4600|
|        Robin| 40|     Sales|  4100|
|        Kunal| 25|   Finance|  3000|
|          Ram| 28|     Sales|  3000|
|      Srishti| 46|Management|  3300|
|         Jeny| 26|   Finance|  3900|
|       Hitesh| 30| Marketing|  3000|
|      Kailash| 29| Marketing|  2000|
|       Sharad| 39|     Sales|  4100|
+-------------+---+----------+------+
```

```
from pyspark.sql.functions import *
from pyspark.sql.window import Window
```

```
win=Window.partitionBy("Department").orderBy(desc("salary"))
```

```
df=df.withColumn("first",row_number().over(win))
```

```
df.show()
```

```
+-------------+---+----------+------+-----+
|Employee_Name|Age|Department|Salary|first|
+-------------+---+----------+------+-----+
|        Meena| 33|     Sales|  4600|    1|
|        Robin| 40|     Sales|  4100|    2|
|       Sharad| 39|     Sales|  4100|    3|
|          Ram| 28|     Sales|  3000|    4|
|          Ram| 28|     Sales|  3000|    5|
|      Srishti| 46|Management|  3300|    1|
|         Jeny| 26|   Finance|  3900|    1|
|        Kunal| 25|   Finance|  3000|    2|
|       Hitesh| 30| Marketing|  3000|    1|
|      Kailash| 29| Marketing|  2000|    2|
+-------------+---+----------+------+-----+
```