Question 1: Spark Program on Employee Dataset

Dataset contains employee information. Assume the dataset includes a Salary column (or Salary derived from PaymentTier).

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

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

# Read CSV file
df = spark.read.csv("employees.csv", header=True, inferSchema=True)

# Filter employees with salary > 50,000
filtered_df = df.filter(col("Salary") > 50000)

# Increase salary by 10%
updated_df = filtered_df.withColumn(
    "UpdatedSalary",
    col("Salary") * 1.10
)

# Show top 5 highest salaries
updated_df.orderBy(col("UpdatedSalary").desc()).show(5)

# Count qualifying employees
count = updated_df.count()
print("Total qualifying employees:", count)
```

```
+----------+------+----------+---------------+------+-------+-----------------+---------------+------------------+
|First Name|Gender|Start Date|Last Login Time|Salary|Bonus %|Senior Management|           Team|     UpdatedSalary|
+----------+------+----------+---------------+------+-------+-----------------+---------------+------------------+
| Katherine|Female| 8/13/1996|       12:21 AM|149908| 18.912|            false|        Finance|164898.80000000002|
|      Rose|Female| 5/28/2015|        8:40 AM|149903|   5.63|            false|Human Resources|164893.30000000002|
|   Cynthia|Female| 7/12/2006|        8:55 AM|149684|  7.864|            false|        Product|164652.40000000002|
|      NULL|Female| 2/23/2005|        9:50 PM|149654|  1.825|             NULL|          Sales|164619.40000000002|
|     Kathy|Female| 3/18/2000|        7:26 PM|149563| 16.991|             true|        Finance|164519.30000000002|
+----------+------+----------+---------------+------+-------+-----------------+---------------+------------------+
only showing top 5 rows
Total qualifying employees: 854
```

Question 2: Pair RDD Operations

```
# Input Data
data = [("A",10), ("B",20), ("A",30), ("B",40), ("C",50)]
rdd = spark.sparkContext.parallelize(data)

# (a) Total value per key
total_per_key = rdd.reduceByKey(lambda x, y: x + y)

# (b) Average value per key
avg_per_key = rdd.mapValues(lambda x: (x,1)) \
                 .reduceByKey(lambda a,b: (a[0]+b[0], a[1]+b[1])) \
                 .mapValues(lambda x: x[0]/x[1])

# (c) Sorted by key
avg_per_key.sortByKey().collect()
```

```
[('A', 20.0), ('B', 30.0), ('C', 50.0)]
```

Question 3: Department Marks using Spark

```
# Input Format
data = [("CS",80), ("AI",90), ("IT",70), ("IT",85), ("EE",75)]
rdd = spark.sparkContext.parallelize(data)

# (a) Max marks per department
max_marks = rdd.reduceByKey(lambda x, y: max(x, y))

# (b) Average marks per department
avg_marks = rdd.mapValues(lambda x:(x,1)) \
               .reduceByKey(lambda a,b:(a[0]+b[0], a[1]+b[1])) \
               .mapValues(lambda x:x[0]/x[1])

# (c) Departments with average > 75
result = avg_marks.filter(lambda x: x[1] > 75)

# Actions to display output
print("MAX MARKS:", max_marks.collect())
print("AVG MARKS:", avg_marks.collect())
print("RESULT:", result.collect())
```

```
MAX MARKS: [('CS', 80), ('AI', 90), ('IT', 85), ('EE', 75)]
AVG MARKS: [('CS', 80.0), ('AI', 90.0), ('IT', 77.5), ('EE', 75.0)]
RESULT: [('CS', 80.0), ('AI', 90.0), ('IT', 77.5)]
```

```
Start coding or generate with AI.
```