L5DE M3T7 – Apply activity brief

Briefing

For this task your tutor will guide you to complete the following steps:

1. Log in to Microsoft Azure (Github Student Pack)
2. Select Azure Databricks
3. Create a new deployment (Select free trial tier)
4. Create a new cluster
5. Create a new notebook
6. Follow the exercises in the worksheet. Copy and paste relevant cells. Fill in the blanks.

Cell 1:

```
# Initialize PySpark
APP_NAME = "Debugging Prediction Problems"

# If there is no SparkSession, create the environment
try:
  sc and spark
except NameError as e:
  import findspark
  findspark.init()
  import pyspark
  import pyspark.sql

  sc = pyspark.SparkContext()
  spark = pyspark.sql.SparkSession(sc).builder… ← FILL IN

print("PySpark initiated...")
```

Before running Cell 2,

Grab the file “Example.csv”

Use DBFS file upload to place it in databricks.

The path should become /FileStore/tables/example.csv on your cluster.

Cell 2:

```
# Load the text file using the SparkContext
csv_lines = sc.text… ← FILL IN

# Map the data to split the lines into a list
data = csv_lines.map(lambda line: line.split(","))

# Collect the dataset into local RAM
data.collect()
```

What is the output:

(COMPLETE)

Research and write below the pros and cons of running collect() in Spark:

(COMPLETE)

Pros:

1.
2.
3.

Cons:

1.
2.
3.
4.

Is it a good idea to run collect()?

Cell 3: using the groupBy operator

```
# Group the records by the name of the person
csv_lines = sc.textFile("/FileStore/tables/… ← FILL IN
records = csv_lines.map(lambda line: line.split… ← FILL IN
grouped_records = records.groupBy(lambda x: x[0])

# Show the first group
grouped_records.first()

# Count the groups
job_counts = grouped_records.map(
  lambda x: {
    "name": x[0],
    "job_count": len(x[1])
  }
)

job_counts.first()

job_counts.collect()
```

What is the output?

(COMPLETE)

Explain what happened? What does the output represent in plain English?

(COMPLETE)

Cell 4: Map vs FlatMap

```
# Compute a relation of words by line
words_by_line = csv_lines\
  .map(lambda line: line.split(","))

print(words_by_line.collect())

# Compute a relation of words
flattened_words = csv_lines\
  .map(lambda line: line.split(","))\
  .flatMap(lambda x: x)

flattened_words.collect()
```

Split the cell if needed.

What is the difference between map and flatMap? Complete with examples and a plain English explanation.