VISION BOARD

COMPANYWISE Q&A
FRACTAL

DEVIKRISHNA R

LinkedIn: @Devikrishna R Email: visionboard044@gmail.com

1.Question: How do you parse a nested JSON column and flatten the data using PySpark?

Answer:

1. **Load Data:** Use PySpark to load the data as a DataFrame with a string column containing JSON.
2. **Parse JSON:** Use from_json and a predefined schema to parse the JSON strings.
3. **Flatten Data:** Use selectExpr or explicit select to extract fields from the JSON column.

Code Implementation:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

# Initialize Spark session
spark = SparkSession.builder.appName("ParseNestedJSON").getOrCreate()

# Data
data = [
    ("1", '{"name": "John", "age": 30, "city": "New York"}'),
    ("2", '{"name": "Jane", "city": "Los Angeles"}'),
    ("3", '{"name": "Bob", "age": 40}'),
    ("4", '{"city": "Chicago", "country": "USA"}'),
    ("5", '{"hobby": "Reading", "language": "English"}')
]
```

```
# Schema definition
schema = StructType([
    StructField("name", StringType(), True),
    StructField("age", IntegerType(), True),
    StructField("city", StringType(), True),
    StructField("country", StringType(), True),
    StructField("hobby", StringType(), True),
    StructField("language", StringType(), True)
])

# Create DataFrame
df = spark.createDataFrame(data, ["id", "json_data"])

# Parse JSON
parsed_df = df.withColumn("json_data", from_json(col("json_data"), schema))
```

```
# Flatten the DataFrame
flattened_df = parsed_df.select(
    col("id"),
    col("json_data.name"),
    col("json_data.age"),
    col("json_data.city"),
    col("json_data.country"),
    col("json_data.hobby"),
    col("json_data.language")
)

# Show results
flattened_df.show()
```

Output:

```
+---+----+----+-----------+-------+-------+--------+
| id|name| age|       city|country|  hobby|language|
+---+----+----+-----------+-------+-------+--------+
|  1|John|  30|   New York|   null|   null|    null|
|  2|Jane|null|Los Angeles|   null|   null|    null|
|  3| Bob|  40|       null|   null|   null|    null|
|  4|null|null|    Chicago|    USA|   null|    null|
|  5|null|null|       null|   null|Reading| English|
+---+----+----+-----------+-------+-------+--------+
```

---

2. Question: How do you group items by name, aggregate counts, and present the data as a list of key-value pairs in PySpark?

Answer:

1. **Group Data:** Group by name and item, and sum the quantities.
2. **Aggregate as List:** Use collect_list and struct to format the output.

Code Implementation:

```python
from pyspark.sql.functions import col, collect_list, struct, sum

# Data
data = [
    ("john", "tomato", 2),
    ("bill", "apple", 2),
    ("john", "banana", 2),
    ("john", "tomato", 3),
    ("bill", "taco", 2),
    ("bill", "apple", 2),
]
```

```python
# Create DataFrame
df = spark.createDataFrame(data, ["name", "item", "quantity"])

# Group and Aggregate
aggregated_df = df.groupBy("name", "item") \
    .agg(sum("quantity").alias("total_quantity")) \
    .groupBy("name") \
    .agg(collect_list(struct(col("item"), col("total_quantity"))).alias("item"))

# Show results
aggregated_df.show(truncate=False)
```

Output:

```
+----+--------------------------------+
|name|item                            |
+----+--------------------------------+
|bill|[{apple, 4}, {taco, 2}]         |
|john|[{tomato, 5}, {banana, 2}]      |
+----+--------------------------------+
```
