Systems for Data Science *CS 532, Spring 2021*

HW10: SQL Queries Using Spark

For reference, the Student, Attend and Course tables have these values,

Student_ID	**Student_Name**
1	Paul
2	Ryan
3	Benedict
4	Watson
5	Holmes

Table 1: Student

Student_ID	**Course_ID**
1	1
2	2
3	1
4	2
5	3

Table 2: Attend

Course_ID	**Course_Name**
1	Physics
2	Chemistry
3	Mathematics

Table 3: Course

Hence given the query that we must print all the Student names taking course = ’Physics’, the following should be the expected result

Student_Name
Paul
Benedict

Table 4: SQL Query Result

1. Output of SparkSQL EXPLAIN command and interpretation

```
== Parsed Logical Plan ==
'Project ['Student.Name]
+- 'Filter ('Course.COURSE_NAME = Physics)
   +- 'Join Inner, ('Course.COURSE_ID = 'Attend.COURSE_ID)
      :- 'Join Inner, ('Student.ID = 'Attend.ID)
      : :- 'UnresolvedRelation [Student], [], false
      : +- 'UnresolvedRelation [Attend], [], false
      +- 'UnresolvedRelation [Course], [], false

== Analyzed Logical Plan ==
Name: string
Project [Name#1]
+- Filter (COURSE_NAME#9 = Physics)
   +- Join Inner, (COURSE_ID#8 = COURSE_ID#5)
      :- Join Inner, (ID#0 = ID#4)
      : :- SubqueryAlias student
      : : +- LogicalRDD [ID#0, NAME#1], false
      : +- SubqueryAlias attend
      : +- LogicalRDD [ID#4, COURSE_ID#5], false
      +- SubqueryAlias course
         +- LogicalRDD [COURSE_ID#8, COURSE_NAME#9], false

== Optimized Logical Plan ==
Project [Name#1]
+- Join Inner, (COURSE_ID#8 = COURSE_ID#5)
   :- Project [NAME#1, COURSE_ID#5]
   : +- Join Inner, (ID#0 = ID#4)
   : :- Filter isnotnull(ID#0)
   : : +- LogicalRDD [ID#0, NAME#1], false
   : +- Filter (isnotnull(ID#4) AND isnotnull(COURSE_ID#5))
   : +- LogicalRDD [ID#4, COURSE_ID#5], false
   +- Project [COURSE_ID#8]
      +- Filter ((isnotnull(COURSE_NAME#9) AND (COURSE_NAME#9 = Physics)) AND
          ↪ isnotnull(COURSE_ID#8))
         +- LogicalRDD [COURSE_ID#8, COURSE_NAME#9], false

== Physical Plan ==
*(9) Project [Name#1]
+- *(9) SortMergeJoin [COURSE_ID#5], [COURSE_ID#8], Inner
   :- *(6) Sort [COURSE_ID#5 ASC NULLS FIRST], false, 0
   : +- Exchange hashpartitioning(COURSE_ID#5, 200), ENSURE_REQUIREMENTS, [id
       ↪ =#118]
   : +- *(5) Project [NAME#1, COURSE_ID#5]
   : +- *(5) SortMergeJoin [ID#0], [ID#4], Inner
   : :- *(2) Sort [ID#0 ASC NULLS FIRST], false, 0
   : : +- Exchange hashpartitioning(ID#0, 200), ENSURE_REQUIREMENTS, [id=#104]
   : : +- *(1) Filter isnotnull(ID#0)
   : : +- *(1) Scan ExistingRDD[ID#0,NAME#1]
   : +- *(4) Sort [ID#4 ASC NULLS FIRST], false, 0
   : +- Exchange hashpartitioning(ID#4, 200), ENSURE_REQUIREMENTS, [id=#110]
   : +- *(3) Filter (isnotnull(ID#4) AND isnotnull(COURSE_ID#5))
   : +- *(3) Scan ExistingRDD[ID#4,COURSE_ID#5]
   +- *(8) Sort [COURSE_ID#8 ASC NULLS FIRST], false, 0
      +- Exchange hashpartitioning(COURSE_ID#8, 200), ENSURE_REQUIREMENTS, [id
          ↪ =#124]
         +- *(7) Project [COURSE_ID#8]
            +- *(7) Filter ((isnotnull(COURSE_NAME#9) AND (COURSE_NAME#9 =
                ↪ Physics)) AND isnotnull(COURSE_ID#8))
               +- *(7) Scan ExistingRDD[COURSE_ID#8,COURSE_NAME#9]
```

Interpretation: The output of the explain consists of a parsed logical plan, an analyzed logical plan, optimized logical plan and physical plan. The parsed logical plan is an unresolved plan that is extracted from the given SQL query. The analyzed logical plans transforms which translates UnresolvedRelation into fully typed objects. The optimized logical plan transforms through a set of optimization rules that results in the physical plan.

2. Discussion on how to implement the same application using only RDDs

Assume that each of the three tables, Student (`student_rdd`), Attend (`attend_rdd`) and Course (`course_rdd`) are represented as separate RDDs in Spark.

Firstly, we can join the student and attend rdds like: `student_rdd.join(attend_rdd, student_rdd.id == attend_rdd.id, how='inner')`
Assuming this result is stored in a temporary RDD named `temp_rdd`, we can then perform the next join with course rdd like: `temp_rdd.join(course_rdd, temp_rdd.course_id == course_rdd.course_id, how='inner')`
Assuming this result is stored in a temporary RDD named `temp2_rdd`, as the output for the sql query we can now perform the following command to generate the required result: `temp2_rdd.filter(lambda x: x['course_name'] == 'Physics')`.
Assuming this result is in another RDD named `temp3_rdd`, in order to display the final output we must first perform: `temp3_rdd.collect()` and later loop through reach row in the collection and display the "student_name" attribute of each row.