Åbo Akademi University

Advance Course on Database

Weekly Report 5

Luis Araújo(2004624)

May 27, 2021

Contents

Chapter 1

Introduction

This weeks topic is **Query optimization**. In the first part of the assignment, we aboard the execution plan and the explanations of how the queries is executed. Following the second part, we had to transform relational algebra expressions into equivalent expressions using the transformation rules for relational algebra expressions.
Once Charles Oredola stoped answering our texts, for this fifth assignment, only myself and Filipe Felicio provided the following answers.

Chapter 2

Exercises

2.1 1) Formulate SQL queries that answer the following questions and generate the execution plan for each of them. Study the execution plans and write verbal explanations of how the queries is executed. Include pictures of the execution plans, and a short explanation of how each query is executed.

2.1.1 a) How many credits have the student with name 'Conti' passed? The result should contain student ID, name of the student and the total number of credits.

```
SELECT
    student.ID, student.name, SUM(course.credits) AS total_cred
FROM
    student
        JOIN
    takes ON student.ID = takes.ID
        JOIN
    course ON takes.course_id = course.course_id
WHERE
    student.name = 'Conti'
GROUP BY (student.ID);
```

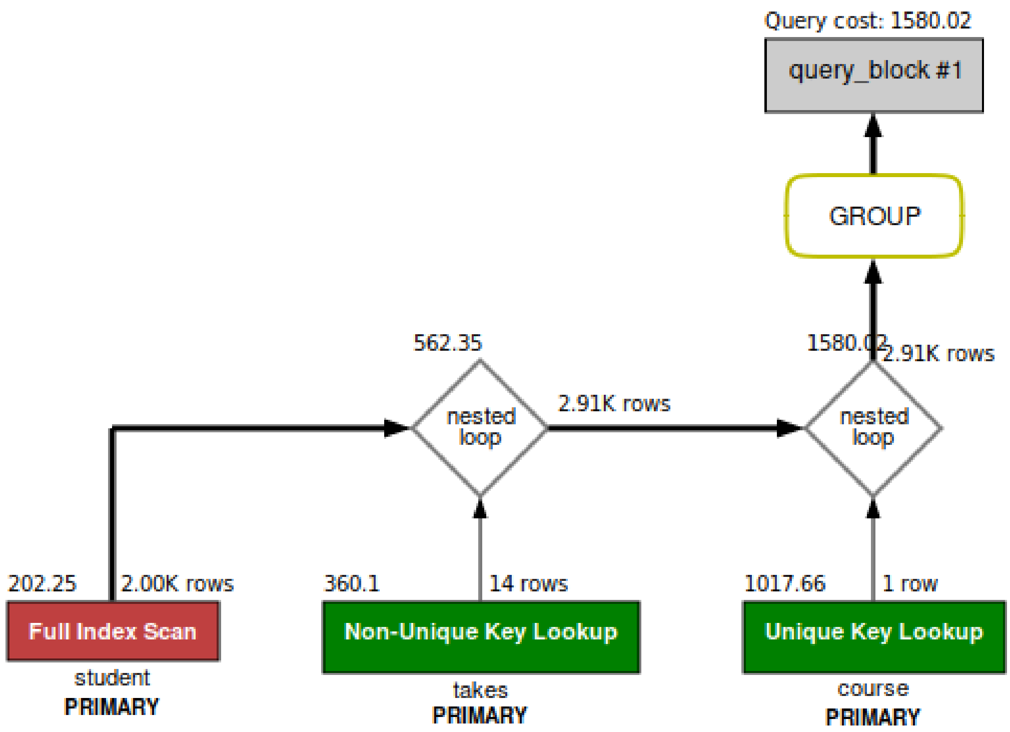

Figure 2.1: Execution Plan

Review

Correct exercise.

2.1.2 b) Which courses have the student named 'Conti' taken? The result should include student ID,student name, course title, year the course was taken, grade of the course and the number of credit points of each course.

```sql
SELECT
    student.ID,
    student.name,
    course.title,
    takes.year,
    course.credits
FROM
    student
        JOIN
    takes ON student.ID = takes.ID
        JOIN
    course ON takes.course_id = course.course_id
WHERE
    student.name = 'Conti';
```

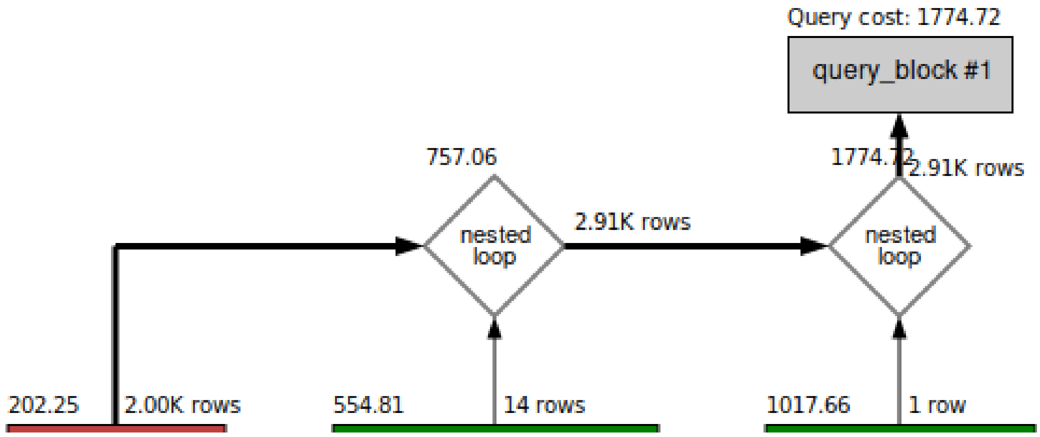

Figure 2.2: Execution Plan

Review

Correct exercise.

2.1.3 c) Which courses have the student with ID '82301' taken? The result should include the student ID, student name, course title, year the course was taken, grade of the course and the number of credit points of each course. The result will be identical to the previous query since Conti has student ID '82301'. However, the execution plan will be different. Explain how it differs from the previous query.

```
SELECT
    student.ID,
    student.name,
    course.title,
    takes.year,
    course.credits
FROM
    student
        JOIN
    takes ON student.ID = takes.ID
        JOIN
    course ON takes.course_id = course.course_id
WHERE
    student.ID = '82301';
```

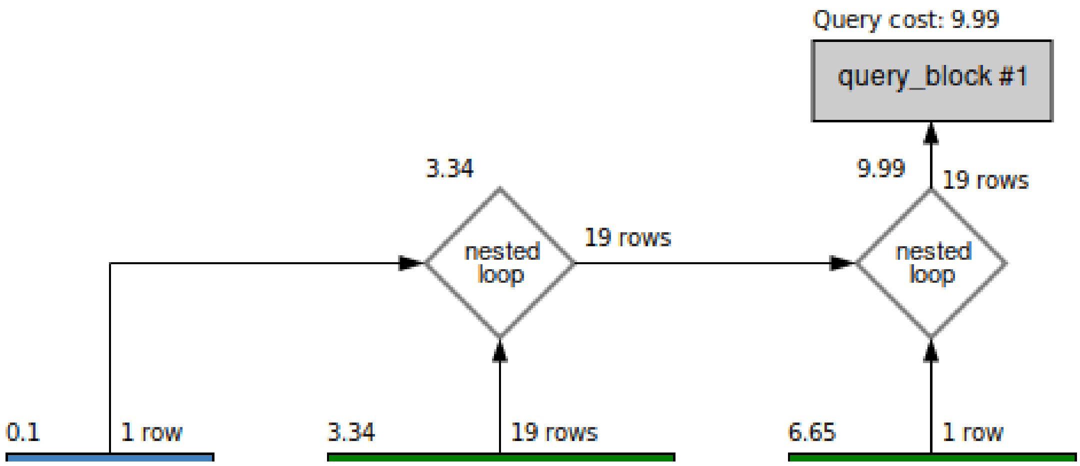

Figure 2.3: Execution Plan

Review

Correct exercise.

2.1.4 d) Create a list of the students in the Biology department and the number of credit points each of these have passed. The result should include the student ID, the student name and the sum of credits points for the courses the student has passed.The result should be in descending order of the number of credit points the student has (that is, the student with most credit points first).

```
SELECT
    student.ID, COUNT(course.credits) AS total_cred
FROM
    student
        JOIN
    takes ON student.ID = takes.ID
        JOIN
    course ON takes.course_id = course.course_id
GROUP BY student.ID
ORDER BY total_cred DESC;
```

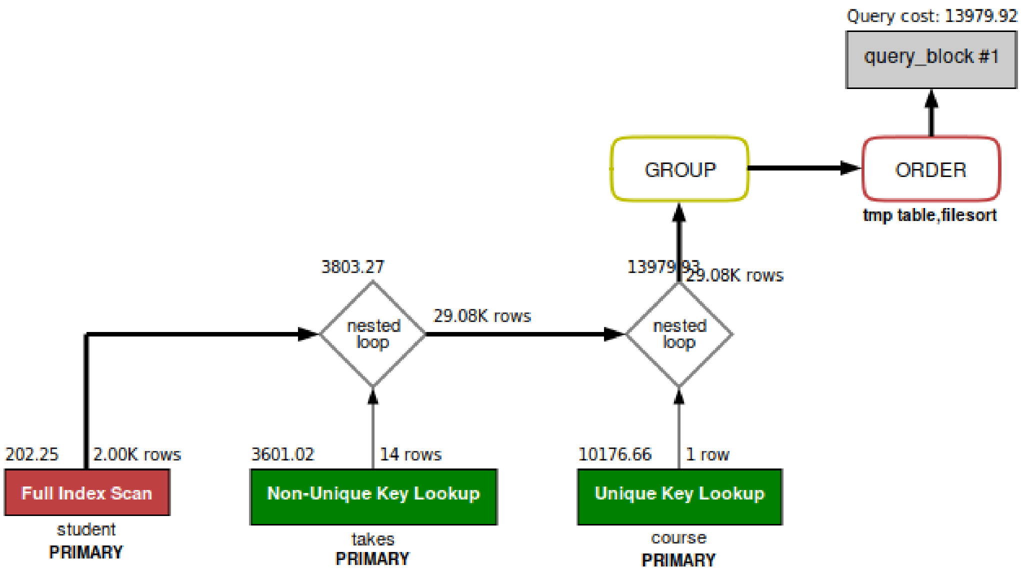

Figure 2.4: Execution Plan

Review

Correct exercise.

2.1.5 e) For each instructor, for how many students does the instructor act as an advisor. The result should contain the ID of the instructor, the name and department of the instructor and the number of students this instructor is advisor for. The result should be sorted by the number of students.

```sql
SELECT
    instructor.ID,
    instructor.dept_name,
    COUNT(s_ID) AS num_students
FROM
    instructor
        JOIN
    advisor ON instructor.ID = i_ID
GROUP BY instructor.ID
ORDER BY num_students DESC;
```

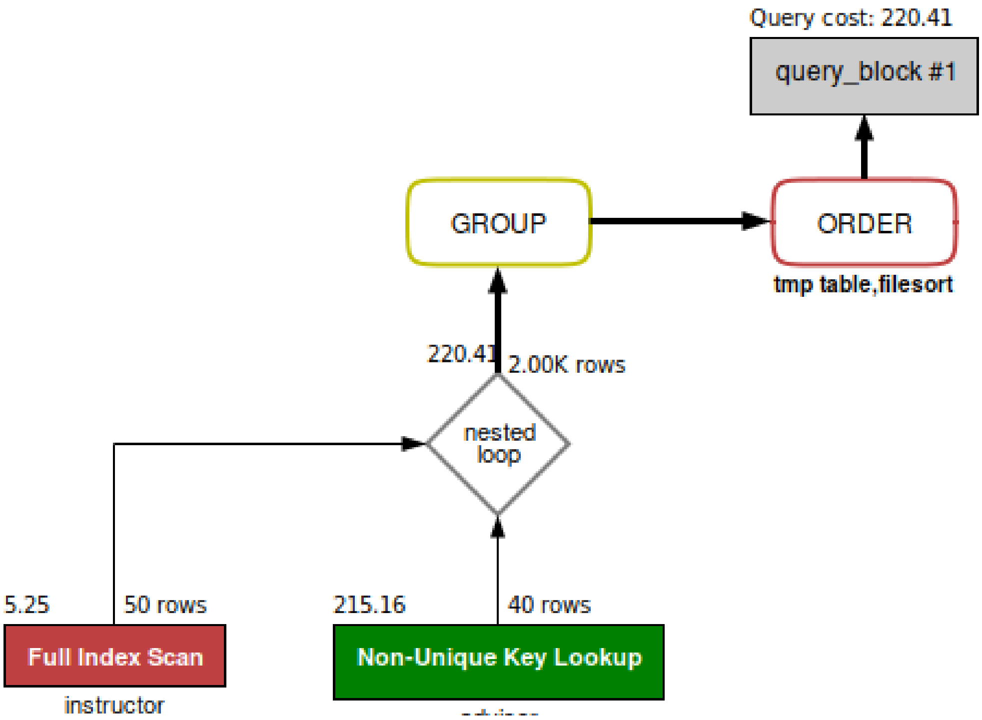

Figure 2.5: Execution Plan

Review

Correct exercise.

2.1.6 f) Which department pays the largest total salary sum to its instructors?

```
SELECT
    instructor.dept_name, SUM(instructor.salary) AS total_salary
FROM
    instructor
GROUP BY instructor.dept_name
ORDER BY total_salary DESC
LIMIT 1;
```

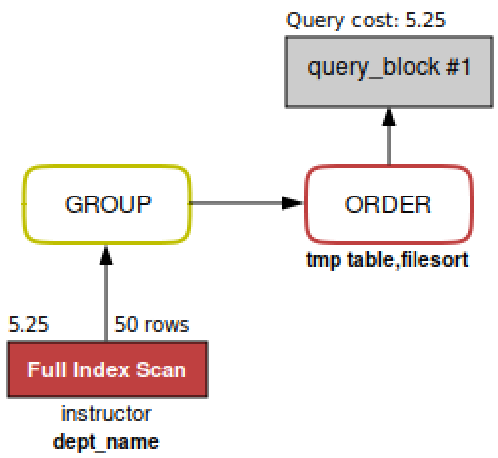

Figure 2.6: Execution Plan

Review

Correct exercise.

2.2 2) Choose two queries in question 1 above and change the order in which tables are joined by using STRAIGHT_JOIN so that the order becomes different than in the original formulation. Compare the execution plans to the original (optimized) plans and check how this join order affects the total cost of the query

2.2.1 d)

```
SELECT
    student.ID, COUNT(course.credits) AS total_cred
FROM
    student
        STRAIGHT_JOIN
    takes ON student.ID = takes.ID
        STRAIGHT_JOIN
    course ON takes.course_id = course.course_id
GROUP BY student.ID
ORDER BY total_cred DESC;
```

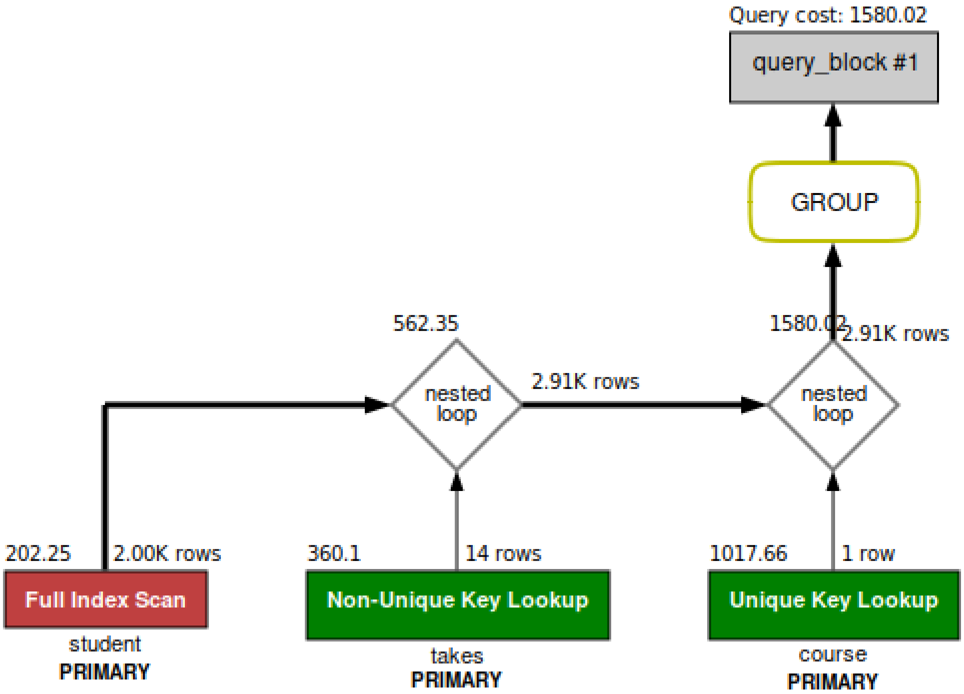

Figure 2.7: Execution Plan

Review

Correct exercise.

2.2.2 e)

```
SELECT
    instructor.ID,
    instructor.dept_name,
    COUNT(s_ID) AS num_students
FROM
    instructor
        STRAIGHT_JOIN
    advisor ON instructor.ID = i_ID
GROUP BY instructor.ID
ORDER BY num_students DESC;
```

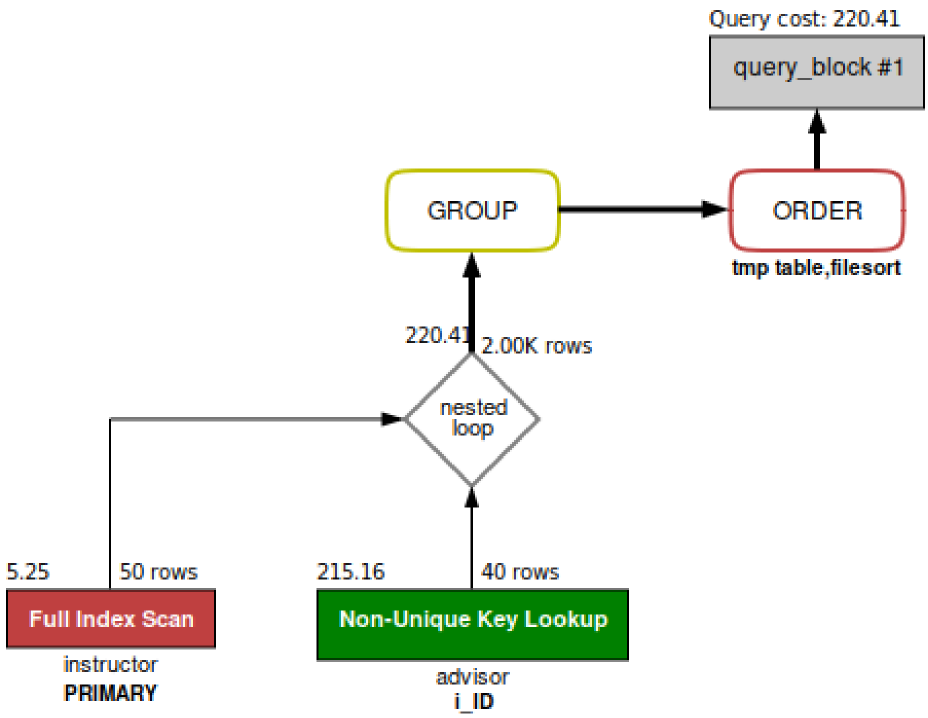

Figure 2.8: Execution Plan

Review

Correct exercise.

2.2.3 3) Assume we have a relation r = (A, B, C) and a relation s = (C, D, E) where the primary keys are underlined. Transform the following relational algebra expressions into equivalent expressions using the transformation rules for relational algebra expressions.

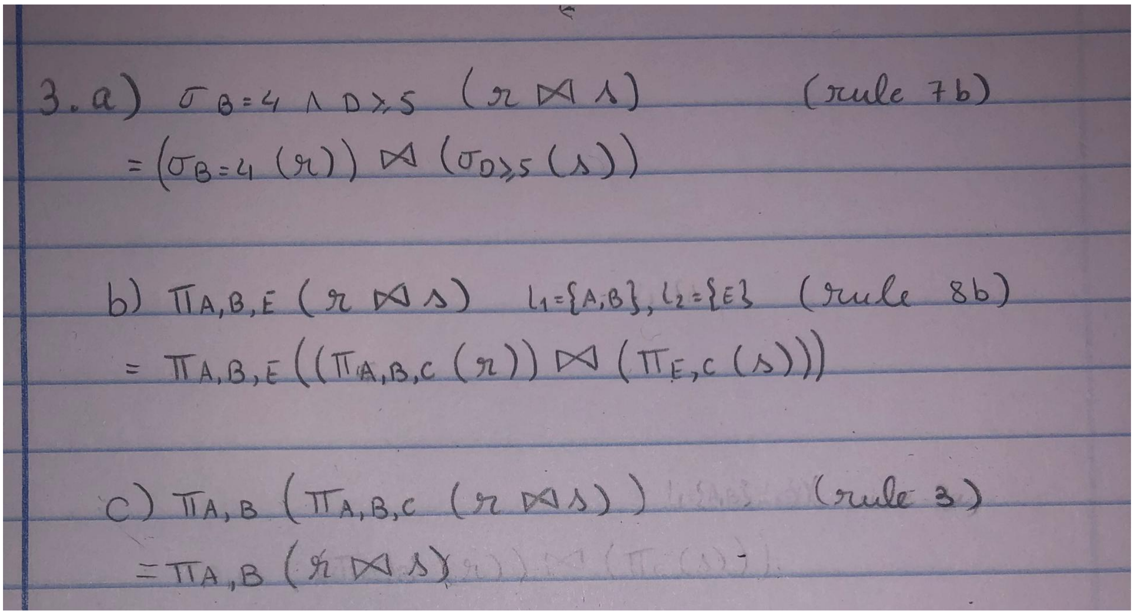

Figure 2.9: Exercise 3

Review

Having a better transformation, we could have used the rule 8b in the exercise c).

2.3 4) Consider the relations r = (A, B, C, D) and s = (D, E, F, G) where the primary keys are underlined. Transform the following relational algebra expressions into equivalent expressions, which can be executed more efficiently.

2.3.1 a) The equivalent expression is executed more efficiently because it reduces the number of projections.

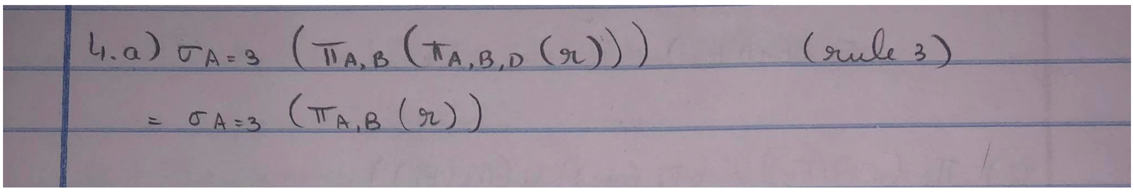

Figure 2.10: Exercise 4.a

Review

Correct exercise.

2.3.2 b) Since the join operator is a costly operation, minimizing this operation is a good way of increasing the efficiency of the query. The equivalent expression makes this operation more efficient because, before operating the join, it will first decrease the rows of the tables.

b) $\Pi_{A,B,C,D}(\sigma_{B=7 \wedge E \geq 5}(r \bowtie s))$ (rule 7b)

$= \Pi_{A,B,C,D}(\sigma_{B=7}(r) \bowtie \sigma_{E \geq 5}(s))$

Figure 2.11: Exercise 4.b

Review

For a better transformation, the team should have used first the rule 1, then 7a, and end it with the rule 8a.

2.3.3 c) For the same reason as the previous exercise, the equivalent expression is more efficient.

c) $\Pi_{A,B,E}(r \bowtie s)$ $L_1 = \{A,B\}$, $L_2 = \{E\}$ (rule 8b)

$= \Pi_{A,B,E}((\Pi_{A,B,C}(r)) \bowtie (\Pi_{E,C}(s)))$

Figure 2.12: Exercise 4.c

Review

Correct exercise.

2.3.4 d) From the group's point of view, the equivalent expression obtained is more efficient. Once the attribute D is a primary key, the search for an attribute D will be very cost-efficient. So, from the first expression, we can observe that the select operation will pass through the whole $r \cup s$ table, while the equivalent expression will first select a primary key from the $r \cup s$ table, making this table smaller for the next selection.

d) $\Pi_{A,C,D}\,(\sigma_{A<15 \ \wedge \ D=2010}\,(r \cup s))$

$= \Pi_{A,C,D}\,(\sigma_{A<15}\,(\sigma_{D=2010}\,(r \cup s)))$

Figure 2.13: Exercise 4.d

Review

The rule that the group should have used was the projection distributes over union (rule 12).

Chapter 3

Conclusion

Even though, coming up to this weeks assignment, the group had never heard of query optimization, it was really interesting, in particular, to solve that questions about the transformation of the algebra expressions into a more efficient one. In my opinion, the group achieved the level and understandings required to solve the assignment.