**Project topic**: Improve the student performance in exams

**Data source**: The dataset, Student Performance in Exams, is from <https://www.kaggle.com/spscientist/students-performance-in-exams>

**Goal**

A high school chancellor would like to know how to improve their students’ exam performance in order to make the school better. To achieve this goal, I will use the student performance data which includes the students’ past performance on three different subjects, and three factors, family, personal, and economics. Based on these information from the dataset, I am going to present graph analysis and identify the following questions,

* How do the three factors affect student’s performance?
* How effective is the test preparation course?

After knowing how the factors affect, the school might change the strategy of teaching or provide a specific course for student in order to have a better learning.

**Graph data model**

(Change the TestGrade node because there are three columns added during data preprocessing. And I also canceled the relationship between Race-TestGrade and ParentalLevel-TestGrade because I think it is not necessary to have them.)

Diagram

Description automatically generated

**Graph projections**

**Diagram

Description automatically generated(1)**

**(2)**

**Diagram

Description automatically generated**

**(3)**

**Diagram

Description automatically generated**

**Data preprocessing**

This dataset includes the student’s test score on three subjects. Adding the letter grade columns on each score can let them easier to compare how the student’s performance in each subject.

**Neo4j database screenshot**

**A screenshot of a phone

Description automatically generated with medium confidence**

**Cypher Queries**

Overall look at the student who has at least one good grade, which is grade A or grade B, in three subjects

Code:

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Match (p:ParentalLevelOfEducation)<-[:HAS\_PARENTS\_BACKGROUND]-(s:StudentInfo)-[:CLASSIFIED\_AS]->(g:Race)

Where (t.reading\_grade = 'A' or t.reading\_grade = 'B'or t.math\_grade = 'A' or t.math\_grade = 'B' or t.writing\_grade = 'A' or t.writing = 'B')

Return t.reading\_grade as good\_reading\_grade,t.math\_grade as good\_math\_grade,t.writing\_grade as good\_writing\_grade, s.lunch as lunch\_status, s.test\_preparation\_course as test\_preparation\_course , s.gender as gender, g.group as race, p.level as parental\_level\_of\_edu

Result table

Graphical user interface, table

Description automatically generated

According to the result table, we can see that there are 299 students who have at least got a grade A or B on the test. Next, I chose gender, lunch status, and parental level of education to see how these factors affect the students.

1. Count how many female students who have good performance,

Code

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Where (t.reading\_grade = 'A' or t.reading\_grade = 'B'or t.math\_grade = 'A' or t.math\_grade = 'B' or t.writing\_grade = 'A' or t.writing = 'B') AND (s.gender = 'female')

Return count(s.gender) as count\_female\_goodgrade

Graphical user interface, text, application

Description automatically generated

1. Count how many students whose lunch status is standard,

Code

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Where (t.reading\_grade = 'A' or t.reading\_grade = 'B'or t.math\_grade = 'A' or t.math\_grade = 'B' or t.writing\_grade = 'A' or t.writing = 'B') AND (s.lunch = 'standard')

Return count(s.lunch) as count\_lunch\_standard\_goodgrade

Graphical user interface, text, application

Description automatically generated

1. Count how many students’ parents are higher educated,

Code

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Match (p:ParentalLevelOfEducation)<-[:HAS\_PARENTS\_BACKGROUND]-(s:StudentInfo)

Where (t.reading\_grade = 'A' or t.reading\_grade = 'B'or t.math\_grade = 'A' or t.math\_grade = 'B' or t.writing\_grade = 'A' or t.writing = 'B') AND (p.level = "master's degree" or p.level = "bachelor's degree")

Return count(p.level) as count\_higheredu\_goodgrade

Graphical user interface, text, application

Description automatically generated

From the above result, for the student in the good grade group, we can see that female students(56%) has a little bit better performance in exams than male student(44%); the student whose lunch status is standard(78.6%) has better performance in exams, so it seems like the economic factor doesn’t affect the student who has good grade; additionally, the student score is not affected by their parents who have higher education(22%)

Another insight for looking at how many student have low performance, which is grade D or F, on each test grade. By looking at each subject, we can know if the school have to work more on which subject in order to make students get improvement.

Code

1. Low performance on math

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Where (t.math\_grade = 'D'or t.math\_grade = 'F')

Return count(s.student\_id) as bad\_math\_grade

Graphical user interface, text, application

Description automatically generated

1. Low performance on writing

Code

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Where (t.writing\_grade = 'D'or t.writing\_grade = 'F')

Return count(s.student\_id) as bad\_writing\_grade

Graphical user interface, application

Description automatically generated

1. Low performance on reading

Code

Match (t:TestGrade)<-[:GET\_GRADE]-(s:StudentInfo)

Where (t.reading\_grade = 'D'or t.reading\_grade = 'F')

Return count(s.student\_id) as bad\_reading\_grade

Graphical user interface, text, application

Description automatically generated

From above results, the school must have new courses related to each subject. These three subjects all include half of students that have low performance.

**Apply Algorithm**

1. In order to know how the test preparation course effects on student’s exam, I apply pageRank algorithm on the data.

CALL gds.graph.create('student','StudentInfo','GET\_GRADE')

Call gds.pageRank.stream('student') YIELD nodeId, score

WITH gds.util.asNode(nodeId) AS n,score

MATCH (n)-[t:GET\_GRADE]-()

RETURN distinct n.test\_preparation\_course,score,count(t) AS course\_status

order by score desc

Graphical user interface, text, application

Description automatically generated

1. This query showed each math grade pageRank and how many students got on each grade. From the result, about 33% of students failed on math exam.

CALL gds.pageRank.stream({

nodeProjection: "TestGrade",

relationshipProjection: "GET\_GRADE",

maxIterations: 20,

dampingFactor: 0.85

})

YIELD nodeId, score

WITH gds.util.asNode(nodeId) AS n,score

MATCH (n)-[t:GET\_GRADE]-()

RETURN distinct n.math\_grade,score, count(t) AS interactions

order by score desc

Graphical user interface

Description automatically generated

1. This query showed each writing grade pageRank and how many students got on each grade. From the result, about 30% of students failed on writing exam.

CALL gds.pageRank.stream({

nodeProjection: "TestGrade",

relationshipProjection: "GET\_GRADE",

maxIterations: 20,

dampingFactor: 0.85

})

YIELD nodeId, score

WITH gds.util.asNode(nodeId) AS n,score

MATCH (n)-[t:GET\_GRADE]-()

RETURN distinct n.writing\_grade,score, count(t) AS interactions

order by score desc

Graphical user interface, application

Description automatically generated

1. This query showed each reading grade pageRank and how many students got on each grade. From the result, about 27.5% of students failed on writing exam.

CALL gds.pageRank.stream({

nodeProjection: "TestGrade",

relationshipProjection: "GET\_GRADE",

maxIterations: 20,

dampingFactor: 0.85

})

YIELD nodeId, score

WITH gds.util.asNode(nodeId) AS n,score

MATCH (n)-[t:GET\_GRADE]-()

RETURN distinct n.reading\_grade,score, count(t) AS interactions

order by score desc

Graphical user interface, application, email

Description automatically generated