Student Performance Analysis Using SQL

1. Introduction

Student data is critical in monitoring and improving performance outcomes in educational institutions. This project focuses on analyzing student data using SQL to derive meaningful insights. The dataset includes two interconnected tables: student_profile and student_performance. By leveraging SQL queries, we aim to:

- Retrieve and aggregate data efficiently.
- Compare performance metrics across different student demographics.
- Identify trends, groupings, and outliers in performance.
- Provide actionable insights for academic interventions and decision-making.

This report details the SQL queries used in the analysis, their purposes, and the insights derived.

2. Dataset Structure

The dataset consists of the following tables:

2.1 student_profile

- **Columns:**
    - student_ID (Primary Key): Unique identifier for each student.
    - first_name: The student's first name.
    - second_name: The student's last name.
    - age: Age of the student.
    - place_of_birth: The city or place where the student was born.

2.2 student_performance

- **Columns:**
    - student_ID (Foreign Key): Links to the student_profile table.
    - average_performance: The average performance score of the student (0 to 100).

3. Objectives

The primary objectives of the SQL analysis are:

1. To combine data from the two tables and create comprehensive reports.
2. To identify patterns in student performance based on demographics.
3. To rank and categorize students based on their performance.
4. To provide actionable insights through advanced grouping and filtering.

SQL Queries and Their Insights

- **Query for place of birth:**

1. Retrieves all students and their place of birth.
2. Count students from each place of birth.
3. Group students by place of birth and average age.

- **Query for the age of the students:**

1. Retrieves all student and their age
2. Group students by age and count students in each group
3. Find the youngest students
4. Find the oldest students
5. Find the students older than 17

- **Query for student's average performance:**

1. Retrieves students and their average performance
2. Group students by ranges of 10 and count how many belong to each range
3. Retrieve students with an average performance greater than 70
4. Finds the lowest average performance among all students
5. Finds the highest average performance among all students
6. This query calculates the overall average student performance

- **Query for performance analysis:**

1. Group students by age and calculate their average performance
2. Classify students into performance categories and group them by age.
3. Get the average performance for students grouped by their place of birth.
4. Retrieve students whose performance is below a specific threshold of 50%.
5. Rank students based on their average performance, where the highest performance gets rank 1

5. Conclusion

This project demonstrates the effective use of SQL for analyzing student data. The queries provide insights into demographic trends, performance distribution, and individual rankings, enabling data-driven decision-making in educational settings.

Key takeaways include:

- Understanding performance trends by age and geography.
- Identifying top performers and students needing intervention.
- Aggregating and categorizing data for meaningful analysis.

By using SQL as a tool, this analysis empowers educational institutions to make informed decisions to enhance student outcomes.