Student Database Analysis – Documentation

1. Database Schema (Tables)

- **Students Table**

```
CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    age INT,
    gender VARCHAR(10)
);
INSERT INTO Students (name, age, gender) VALUES
('Rahul Sharma', 20, 'Male'),
('Priya Verma', 21, 'Female'),
('Amit Patil', 22, 'Male'),
('Neha Gupta', 20, 'Female');
```

student_id	name	age	gender
1	Rahul Sharma	20	Male
2	Priya Verma	21	Female
3	Amit Patil	22	Male
4	Neha Gupta	20	Female

Creates a table named Students with columns:

- student_id → Unique ID for each student (auto-incremented).
- name → Student's name (up to 100 characters).
- age → Student's age (integer).
- gender → Student's gender (Male/Female).

Inserts 4 sample student records into the table:

- Rahul Sharma, 20, Male
- Priya Verma, 21, Female
- Amit Patil, 22, Male
- Neha Gupta, 20, Female

- **Subjects Table**

```
CREATE TABLE Subjects (
    subject_id INT PRIMARY KEY AUTO_INCREMENT,
    subject_name VARCHAR(100)
);
INSERT INTO Subjects (subject_name) VALUES
('Database Systems'),
('Data Structures'),
('Mathematics');
```

subject_id	subject_name
1	Database Systems
2	Data Structures
3	Mathematics

Creates a table named Subjects with columns:

- subject_id → Unique ID for each subject (auto-incremented).
- subject_name → Name of the subject (up to 100 characters).

Inserts 3 sample subjects into the table:

- Database Systems
- Data Structures
- Mathematics

- **Attendance Table**

```
CREATE TABLE Attendance (
    attendance_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    subject_id INT,
    attendance_percent DECIMAL(5,2),
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (subject_id) REFERENCES Subjects(subject_id)
```

);

INSERT INTO Attendance (student_id, subject_id, attendance_percent) VALUES

(1,1,85.50),

(1,2,90.00),

(2,1,78.00),

(2,3,88.00),

(3,2,70.00),

(4,1,95.00);

attendance_id	student_id	subject_id	attendance_percent
1	1	1	85.50
2	1	2	90.00
3	2	1	78.00
4	2	3	88.00
5	3	2	70.00
6	4	1	95.00

Table Creation: Creates an Attendance table with:

- attendance_id as a unique ID,
- student_id and subject_id as references to Students and Subjects tables,
- attendance_percent to store attendance as a decimal.

Data Insertion: Adds attendance records for specific students in specific subjects with their attendance percentages.

- **Grades Table**

CREATE TABLE Grades (

grade_id INT PRIMARY KEY AUTO_INCREMENT,

student_id INT,

subject_id INT,

marks DECIMAL(5,2),

FOREIGN KEY (student_id) REFERENCES Students(student_id),

FOREIGN KEY (subject_id) REFERENCES Subjects(subject_id)

);

INSERT INTO Grades (student_id, subject_id, marks) VALUES

(1,1,92.00),

(1,2,85.00),

(2,1,80.00),

(2,3,75.00),

(3,2,60.00),

(4,1,95.00);

grade_id	student_id	subject_id	marks
1	1	1	92.00
2	1	2	85.00
3	2	1	80.00
4	2	3	75.00
5	3	2	60.00
6	4	1	95.00

Creates a Grades table with columns:

- **grade_id:** unique ID for each grade (auto-incremented).
- **student_id:** links to a student in the Students table.
- **subject_id:** links to a subject in the Subjects table.
- **marks:** the score obtained (decimal).
- Foreign keys ensure student_id and subject_id exist in their respective tables.

Inserts sample grade records for students in different subjects.

2. SQL Queries for Analysis

a) Average Marks per Subject

SELECT s.subject_name, AVG(g.marks) AS avg_marks

FROM Grades g

JOIN Subjects s ON g.subject_id = s.subject_id

GROUP BY s.subject_name;

Subject	Avg Marks
Database Systems	88.5
Data Structures	72.5
Mathematics	75.0

This SQL query calculates the **average marks for each subject**.

- FROM Grades g JOIN Subjects s ON g.subject_id = s.subject_id → Combines the Grades table with Subjects to get subject names.
- GROUP BY s.subject_name → Groups the data by each subject.
- AVG(g.marks) AS avg_marks → Computes the average marks for each group (subject).

b) Attendance Below 80%

```
SELECT st.name, sub.subject_name, a.attendance_percent
FROM Attendance a
JOIN Students st ON a.student_id = st.student_id
JOIN Subjects sub ON a.subject_id = sub.subject_id
WHERE a.attendance_percent < 80;
```

Name	Subject	Attendance (%)
Priya Verma	Database Systems	78.0
Amit Patil	Data Structures	70.0

This SQL query **lists students with low attendance** (less than 80%) in each subject.

- FROM Attendance a JOIN Students st ON a.student_id = st.student_id → Links attendance records to student names.
- JOIN Subjects sub ON a.subject_id = sub.subject_id → Adds the subject name for each record.
- WHERE a.attendance_percent < 80 → Filters only those students whose attendance is below 80%.

c) Top Performer in Each Subject

```
SELECT s.subject_name, st.name, MAX(g.marks) AS top_marks
FROM Grades g
JOIN Students st ON g.student_id = st.student_id
```

JOIN Subjects s ON g.subject_id = s.subject_id

GROUP BY s.subject_name;

Subject	Top Student	Marks
Database Systems	Neha Gupta	95
Data Structures	Rahul Sharma	85
Mathematics	Priya Verma	75

This SQL query **finds the highest marks in each subject**, but there's a subtle issue. Let me explain:

- FROM Grades g JOIN Students st ON g.student_id = st.student_id JOIN Subjects s ON g.subject_id = s.subject_id → Combines grades with student names and subject names.
- GROUP BY s.subject_name → Groups data by subject.
- MAX(g.marks) AS top_marks → Finds the highest marks in each subject.

Issue:
Including st.name without aggregation can give **unexpected results** because SQL doesn't know which student's name to show when there are multiple students per subject.

d) View for Consolidated Student Performance

CREATE VIEW StudentPerformance AS

SELECT st.name, sub.subject_name, g.marks, a.attendance_percent

FROM Students st

JOIN Grades g ON st.student_id = g.student_id

JOIN Subjects sub ON g.subject_id = sub.subject_id

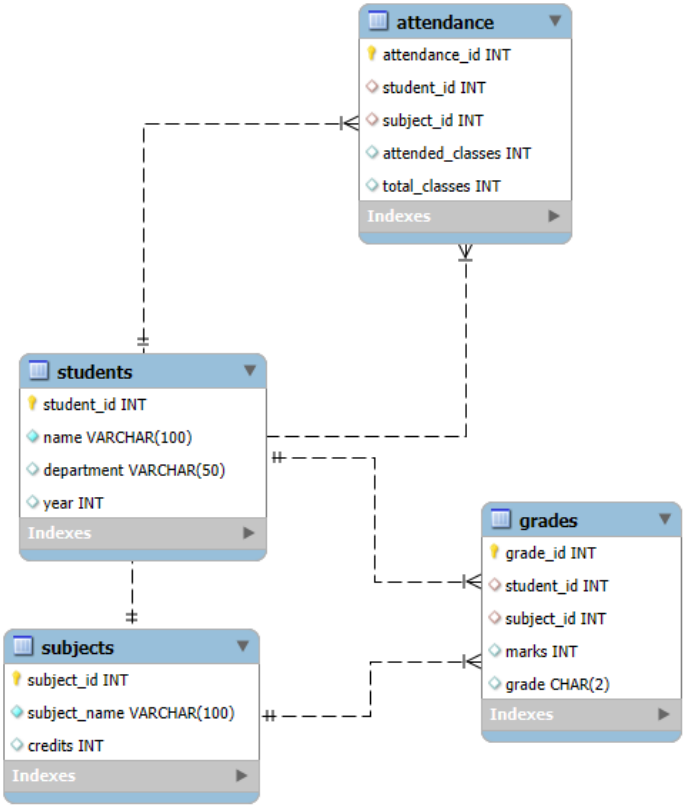
JOIN Attendance a ON st.student_id = a.student_id AND sub.subject_id = a.subject_id;

Name	Subject	Marks	Attendance (%)
Rahul Sharma	Database Systems	92	85.5
Rahul Sharma	Data Structures	85	90.0
Priya Verma	Database Systems	80	78.0
Priya Verma	Mathematics	75	88.0
Amit Patil	Data Structures	60	70.0
Neha Gupta	Database Systems	95	95.0

This SQL statement **creates a view called StudentPerformance** that combines student marks and attendance in one table.

- JOIN Grades g ON st.student_id = g.student_id → Links students with their grades.
- JOIN Subjects sub ON g.subject_id = sub.subject_id → Adds subject names.
- JOIN Attendance a ON st.student_id = a.student_id AND sub.subject_id = a.subject_id → Adds attendance for the corresponding student and subject.

3. ER Diagram



Explanation of the ER Diagram

1. Students Table

- **Attributes:**
  - student_id (Primary Key) → Unique ID for each student.
  - name → Student's full name.
  - department → Department name (e.g., Computer Engg).
  - year → Academic year (e.g., 1, 2, 3, 4).
- **Role:** Stores the core information of students.

2. Subjects Table

- **Attributes:**
  - subject_id (Primary Key) → Unique ID for each subject.
  - subject_name → Name of the subject (e.g., Database Systems).
  - credits → Credit weightage of the subject.
- **Role:** Stores information about the courses/subjects offered.

3. Attendance Table

- **Attributes:**
  - attendance_id (Primary Key) → Unique ID for each attendance record.
  - student_id (Foreign Key → Students) → Which student's attendance is being recorded.
  - subject_id (Foreign Key → Subjects) → For which subject the attendance belongs.
  - attended_classes → How many classes the student has attended.
  - total_classes → Total number of classes conducted.
- **Role:** Tracks how much a student has attended in each subject.

4. Grades Table

- **Attributes:**
  - grade_id (Primary Key) → Unique ID for each grade record.
  - student_id (Foreign Key → Students) → Which student got the grade.
  - subject_id (Foreign Key → Subjects) → Subject for which grade is given.
  - marks → Marks scored by the student.
  - grade → Final grade (A, B, C, etc.).
- **Role:** Stores performance/grades of students for each subject.

Relationships in ERD

1. **Students → Attendance**
   - One student can have multiple attendance records (one per subject).
   - **One-to-Many relationship.**
2. **Students → Grades**
   - One student can have multiple grades (one per subject).
   - **One-to-Many relationship.**

3. **Subjects → Attendance**
    - One subject can be attended by many students.
    - **One-to-Many relationship.**
4. **Subjects → Grades**
    - One subject can have grades of many students.
    - **One-to-Many relationship.**