Name : Sudharsan Tirumal
GitHub link : https://github.com/Sudha548322

STUDENT DATABASE MANAGEMENT SYSTEM

Name : Sudharsan Tirumal

Name : Sudharsan Tirumal
GitHub link : https://github.com/Sudha548322

INDEX

Name : Sudharsan Tirumal
GitHub link : https://github.com/Sudha548322

1. INTRODUCTION

In today’s fast-paced digital world, educational institutions handle vast amounts of student data, including personal details, course enrollments, grades, attendance, exams, fee payments, and extracurricular activities. Managing this data efficiently is crucial for ensuring smooth academic and administrative operations. A Student Database Management System (DBMS) provides a structured, centralized, and secure way to store, retrieve, and manage this information.

Traditional methods such as paper records or spreadsheet-based systems are prone to errors, data redundancy, loss of information, and inefficiency in data retrieval. As institutions grow, the complexity of handling student records increases, leading to challenges in tracking student performance, fee status, and attendance records. Without an efficient system, administrators may face delays in processing information, and faculty members might struggle to access critical student data in real time.

A well-designed DBMS addresses these issues by offering a scalable, automated, and organized approach to data management. It ensures data integrity, security, and accuracy, while also allowing multiple stakeholders—students, faculty, and administrative staff—to access relevant information seamlessly. By implementing a relational database model, institutions can establish clear relationships between different entities such as students, courses, exams, and payments, thereby ensuring smooth operation and quick decision-making.

This case study focuses on designing a Student Database Management System, highlighting key elements such as database structure, entities, attributes, relationships, primary and foreign keys, and data integrity constraints. Additionally, an Entity-Relationship Diagram (ERD) is included to visually represent how different components interact within the system. By adopting this database system, educational institutions can streamline student data management, enhance efficiency, improve reporting, and provide a seamless experience for students and faculty members alike.

1.2 OBJECTIVE

The objective of this case study is to develop a Student Database Management System (DBMS) that streamlines the management of student records, improves data accuracy, and enhances operational efficiency for educational institutions. This system will ensure seamless data retrieval, secure storage, and smooth integration of various academic functions, including enrollment, exams, grading, attendance, and fee management. By implementing a structured database design with well-defined entities, relationships, and integrity constraints, the system aims to reduce redundancy, prevent data inconsistencies, and provide a user-friendly solution for students, faculty, and administrators.

1.3 PURPOSE OF THE DATABASE

The primary purpose of the Student Database Management System (DBMS) is to:

- **Simplify administrative workflows** by centralizing student, fee, and exam data into a single, structured system, reducing manual efforts and ensuring smoother operations.
- **Enable informed decision-making** with real-time insights into student performance, attendance, and financial data, aiding in better planning and resource allocation.
- **Empower students and faculty** by providing easy access to academic and financial data, ensuring they can stay up-to-date on their performance and fee status.
- **Improve financial tracking and fee collection efficiency** by automating fee payment processes and generating timely reports on fee dues, reducing errors and delays.
- **Support and growth with a scalable** and adaptable database system that can handle increasing student data and evolving administrative needs as the institution expands.

This format provides a clear and concise breakdown of the database's objectives in bullet points!

2. ENTITY IDENTIFICATION

1. Student

The Student entity stores all personal and academic details related to each student, such as their name, contact information, and enrollment history. It is essential for managing student-specific data across various functions like enrollment, attendance, and results.

2. Instructor

The Instructor entity contains details about the faculty members, including their names, contact information, and the subjects they teach. It allows the system to associate instructors with specific courses and manage their roles within the institution.

3. Subject

The Subject entity represents the courses offered by the institution, including details like course names, credits, and the department that offers them. This entity helps track the availability and enrollment in different academic subjects.

4. Department

The Department entity stores information about various academic departments within the institution, such as the department name and the head of the department. It is used to categorize subjects and assign instructors to the appropriate departments.

5. Enrollment

The Enrollment entity records the subjects in which students are enrolled, including the student ID and subject ID. It is vital for tracking student participation in courses and managing academic schedules.

6. Fees

The Fees entity manages fee records for students, including amounts due, payment statuses, and payment deadlines. It helps the institution track and manage financial transactions related to student tuition and other charges.

7. Payment Method

The Payment Method entity defines the different ways students can pay their fees, such as credit card, bank transfer, or cash. This entity supports the financial management of fee payments and provides a variety of payment options for students.

8. Grade

The Grade entity outlines the grading criteria for exams and assignments, specifying the grades awarded (e.g., A, B, C) and the corresponding ranges of marks. It is essential for evaluating and categorizing student performance.

9. Results

The Results entity stores the outcome of student assessments, such as marks obtained in exams and the corresponding grades. It is used to track student performance across various subjects and exams over time.

10. Exam

The Exam entity holds details about exams, such as exam dates, total marks, and the subjects being tested. It plays a key role in scheduling and organizing the assessment process for students in each subject.

3. ENTITIES ATTRITBUTES

Table 1: Student

Attribute	**Data Type**	**PK**	**FK**	**Description**
student_id	INT	✓	-	Unique identifier for each student.
name	VARCHAR(100)	-	-	Full name of the student.
email	VARCHAR(100)	-	-	Student’s email (unique).
dob	DATE	-	-	Date of birth.
contact_number	VARCHAR(15)	-	-	Phone number (optional).
address	VARCHAR(200)	-	-	Residential address.

Table 2: Instructor

Attribute	Data Type	PK	FK	Description
instructor_id	INT	✓	-	Unique identifier for instructors.
name	VARCHAR(100)	-	-	Full name of the instructor.
email	VARCHAR(100)	-	-	Instructor’s email (unique).
department_id	INT	-	✓	Links to the Department table.

Table 3: Subject

Attribute	Data Type	PK	FK	Description
subject_id	INT	✓	-	Unique identifier for subjects.
subject_name	VARCHAR(50)	-	-	Name of the subject (e.g., Mathematics).
department_id	INT	-	✓	Links to the Department table.

Table 4: Enrollment

Attribute	Data Type	PK	FK	Description
enrollment_id	INT	✓	-	Unique enrollment record ID.
student_id	INT	-	✓	Links to the Student table.
subject_id	INT	-	✓	Links to the Subject table.
semester	VARCHAR(10)	-	-	Academic semester (e.g., Fall 2023).

Table 5: Grade

Attribute	Data Type	PK	FK	Description
grade_id	INT	✓	-	Unique identifier for grades.
grade_label	CHAR(2)	-	-	Letter grade (e.g., A, B+).
min_score	INT	-	-	Minimum score for the grade.
max_score	INT	-	-	Maximum score for the grade.

Table 6: Exam

Attribute	Data Type	PK	FK	Description
exam_id	INT	✓	-	Unique identifier for exams.
subject_id	INT	-	✓	Links to the Subject table.
exam_date	DATE	-	-	Date of the exam.
total_marks	INT	-	-	Maximum marks for the exam.

Table 7: Results

Attribute	Data Type	PK	FK	Description
result_id	INT	✓	-	Unique result record ID.
student_id	INT	-	✓	Links to the Student table.
exam_id	INT	-	✓	Links to the Exam table.
score	INT	-	-	Marks obtained by the student.

Table 8: Department

Attribute	Data Type	PK	FK	Description
department_id	INT	✓	-	Unique identifier for departments.
department_name	VARCHAR(100)	-	-	Name of the department (e.g., Computer Science).
department_head	VARCHAR(100)	-	-	The head of the department.

Table 9: Fees

Attribute	Data Type	PK	FK	Description
fees_id	INT	✓	-	Unique identifier for fee records.
student_id	INT	-	✓	Links to the Student table.
payment_mode_id	INT	-	✓	Links to the Payment Mode table.
payment_date	DATE	-	-	Date of the fee payment.

due_date	DATE	-	-	Due date for fee payment.
status	VARCHAR(20)	-	-	Status of the fee payment (e.g., Paid, Pending).

Table 10: Payment Mode

Attribute	Data Type	PK	FK	Description
payment_mode_id	INT	✓	-	Unique identifier for payment methods.
mode_name	VARCHAR(50)	-	-	Type of payment (e.g., Credit Card, Cash).

4. KEY RELATIONSHIPS

The key relationships in the Student Database Management System establish connections between entities such as students, instructors, subjects, and results, ensuring efficient data flow and integrity. For example, a student can enroll in multiple subjects, each subject can have several exams, and instructors can teach multiple subjects, with relationships like One-to-Many and Many-to-Many connecting relevant tables. These relationships help the system manage student data, grades, exam results, and course assignments efficiently while maintaining a clear structure for data retrieval and updates.

PRIMARY TABLE	PRIMARY KEY	RELATED TABLE	FOREIGN KEY	RELATION SHIOP TYPE	DESCRIPTION
Student	student_id	Enrollment	student_id	One-to-Many	A student can enroll in multiple subjects over time.

Grade	grade_id	Enrollment	grade_id	One-to-Many	Each enrollment belongs to a specific grade level.
Exam	exam_id	Subject	subject_id	Many-to-One	Each exam belongs to a subject.
Results	result_id	Student	student_id	Many-to-One	Each result is linked to a student.
Fees	fee_id	Student	student_id	Many-to-One	Each student has a fee record.

Primary and Foreign Key Details for the Tables

Below is a detailed breakdown of the Primary Keys and Foreign Keys for each table in the Student Database Management System:

4.1 Primary Keys (PK)

Primary Table	**Primary Key**
Student	student_id
Grade	grade_id
Subject	subject_id
Exam	exam_id
Results	result_id
Fees	fee_id
Enrollment	enrollment_id

- **student_id (Student Table):**
  - Description: Unique identifier for each student. It ensures that each student is distinctly recognized in the system.
- **instructor_id (Instructor Table):**
  - Description: Unique identifier for each instructor, ensuring accurate faculty assignment and reference.
- **subject_id (Subject Table):**
  - Description: Unique identifier for each subject, allowing precise differentiation between courses**.**
- **fee_id (Fees Table):**
  - Description: Unique identifier for each fee record, ensuring that fee transactions are correctly managed.
- **exam_id (Exam Table):**
  - Description: Unique identifier for each exam, helping in organizing, scheduling, and tracking assessments.
- **enrollment_id (Enrollment Table):**
  - Description: Unique identifier for each enrollment record, ensuring student-course registrations are tracked.
- **grade_id (Grade Table):**
  - Description: Unique identifier for each grade level, ensuring grading criteria are properly categorized.
- **result_id (Results Table):**
  - Description: Unique identifier for each exam result, ensuring student performance is correctly recorded.

4.2 FOREIGN KEYS (FK):

Primary Table	**Foreign Key**
Enrollment	student_id
Enrollment	grade_id
Exam	subject_id

Results	student_id
Fees	student_id

- **student_id (Enrollment Table):**
  Description: Links each enrollment to a student, ensuring students are registered in valid courses.
- **grade_id (Enrollment Table):**
  Description: Links each enrollment to a specific grade, ensuring accurate grade assignment for courses.
- **subject_id (Exam Table):**
  Description: Links each exam to a subject, ensuring exams are associated with the correct course.
- **student_id (Results Table):**
  Description: Links each exam result to a student, ensuring performance tracking per individual.
- **student_id (Fees Table):**
  Description: Links each fee record to a student, ensuring financial records correspond to enrolled students.

5. DATA INTEGRITY CONSTRAINTS FOR EACH TABLE

5.1 UNIQUE CONSTRAINTS

Unique constraints are applied to ensure that critical identifiers such as student_id, instructor_id, subject_id, etc., are distinct across the system. This prevents data redundancy and ensures that each record can be uniquely identified and referenced in related tables.

Table	**Attribute**	**Description**
Student	student_id	Ensures each student has a unique identifier.
Instructor	instructor_id	Guarantees each instructor has a unique ID.
Subject	subject_id	Ensures each subject is uniquely identified.
Fees	fee_id	Ensures each fee record is distinct.
Exam	exam_id	Ensures each exam is uniquely identified.

5.2 NOT NULL CONSTRAINTS

The Not Null constraints ensure that essential information is always present when creating or updating records. These constraints are critical for ensuring that key data like names, dates, and amounts are not omitted, thus guaranteeing the integrity of student, exam, and fee records.

Table	**Attribute**	**Description**
Student	student_id	Ensures a student record has a unique ID.
Student	first_name, last_name	Ensures the student's first and last names are always provided.
Subject	subject_name	Ensures every subject has a name.

Fees	amount, due_date	Ensures both fee amount and due date are provided.
Exam	exam_date	Ensures every exam has a valid exam date.

5.3 FOREIGN KEY CONSTRAINTS

Foreign Key Constraints establish relationships between tables, ensuring that records are connected and consistent across the database. They enforce referential integrity by making sure that every foreign key value corresponds to a valid entry in the related primary key table.

Table	**Attribute**	**Description**
Enrollment	student_id	Links enrollment to a valid student record.
Enrollment	subject_id	Links enrollment to a valid subject.
Fees	student_id	Links fee records to a valid student.
Fees	payment_method_id	Links fee records to a valid payment method.
Results	student_id	Links results to a valid student.
Results	exam_id	Links results to a valid exam.

5.4 CHECK CONSTRAINTS

Check Constraints are used to enforce business rules on specific attributes, ensuring that the data entered meets certain conditions. For example, ensuring that marks are within a valid range or that a fee amount is positive helps maintain accurate and logical data in the system.

Table	Attribute	Description
Results	marks	Ensures marks are within a valid range (e.g., 0 to 100).
Fees	amount	Ensures the fee amount is greater than zero.
Fees	status	Ensures fee status is one of the valid options (Paid, Pending, Overdue).

5.5 REFERENTIAL INTEGRITY CONSTRAINTS

Referential Integrity Constraints guarantee that data in one table matches and corresponds to data in another table. These constraints are crucial for maintaining consistency and accuracy in the relationships between related records, such as ensuring that students, subjects, and exams are correctly linked.

Table	Attribute	Description
Enrollment	student_id	Ensures student ID in Enrollment references a valid student.
Enrollment	subject_id	Ensures subject ID in Enrollment references a valid subject.
Fees	student_id	Ensures student ID in Fees references a valid student.
Results	student_id	Ensures student ID in Results references a valid student.

Results	exam_id	Ensures exam ID in Results references a valid exam.
Instructor	department_id	Ensures department ID in Instructor references a valid department.
Subject	department_id	Ensures department ID in Subject references a valid department.

5.6 DEFAULT CONSTRAINTS

Default Constraints ensure that if a value is not explicitly provided for an attribute, the system will automatically assign a default value. This ensures that no field is left blank, providing consistency and avoiding incomplete data entries.

Table	**Attribute**	**Description**
Fees	status	Sets default status of fees to 'Pending' if not specified.

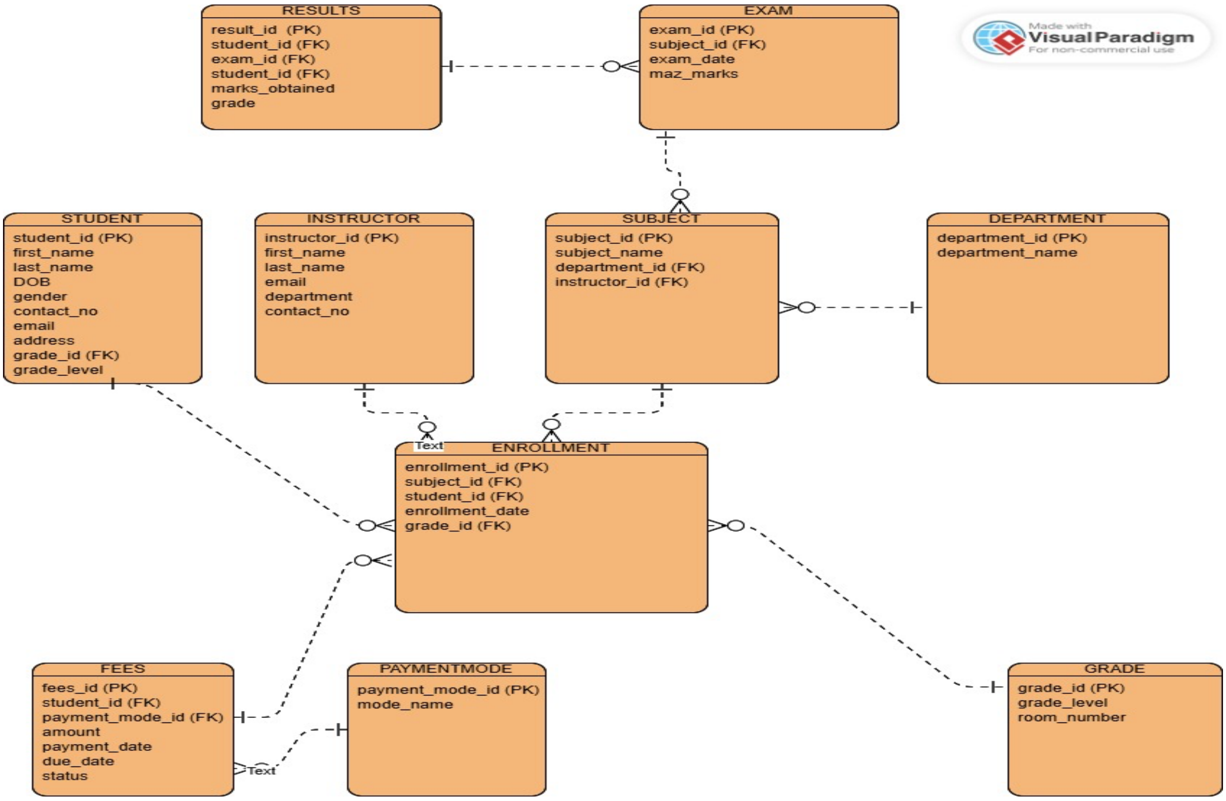
6. ENTITY-RELATIONSHIP DIAGRAM (ERD)

The **Student Database Management System** is designed to **streamline student administration** in an educational institution. This **ERD ensures** that:

- **Students** can enroll in subjects and be assigned grades.
- **Instructors** can be linked to specific departments and subjects.
- **Exam results** are systematically recorded and linked to students.
- **Fee payments** are managed with different payment modes.

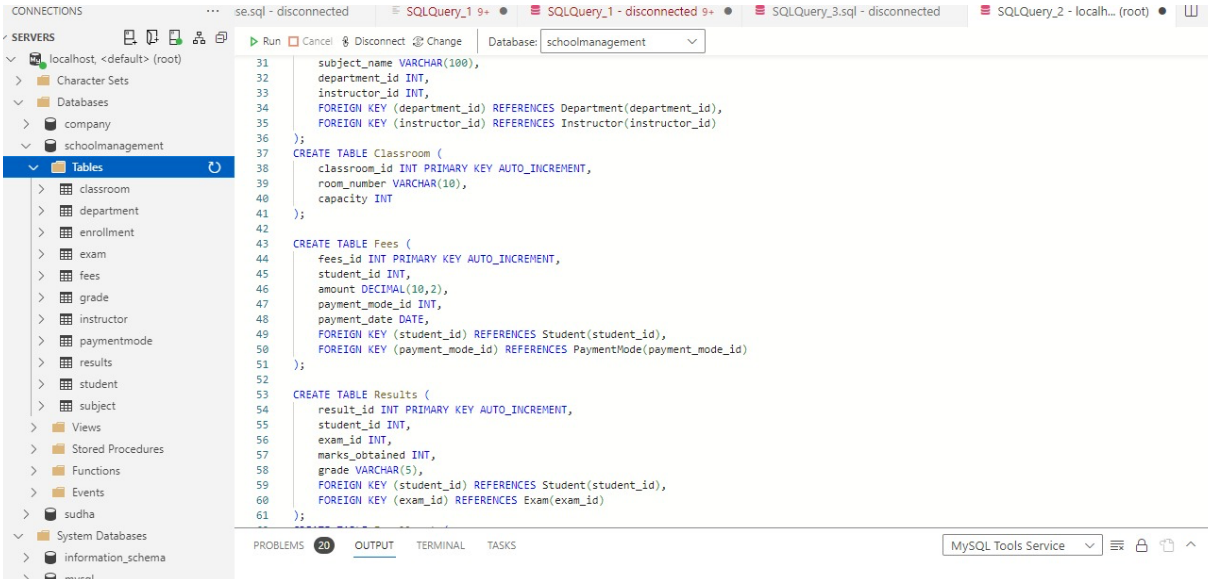
By implementing this database structure, the institution can **efficiently manage academic and financial operations**, ensuring **accuracy, consistency, and scalability**.

Name : Sudharsan Tirumal
GitHub link : https://github.com/Sudha548322



Entity Relationship Diagram

7. DATABASE



Database: schoolmanagement

```sql
    subject_name VARCHAR(100),
    department_id INT,
    instructor_id INT,
    FOREIGN KEY (department_id) REFERENCES Department(department_id),
    FOREIGN KEY (instructor_id) REFERENCES Instructor(instructor_id)
);
CREATE TABLE Classroom (
    classroom_id INT PRIMARY KEY AUTO_INCREMENT,
    room_number VARCHAR(10),
    capacity INT
);

CREATE TABLE Fees (
    fees_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    amount DECIMAL(10,2),
    payment_mode_id INT,
    payment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (payment_mode_id) REFERENCES PaymentMode(payment_mode_id)
);

CREATE TABLE Results (
    result_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    exam_id INT,
    marks_obtained INT,
    grade VARCHAR(5),
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (exam_id) REFERENCES Exam(exam_id)
);
```

Database-image

8. SAMPLE QUERIES

Query 1:

Find the Number of Students Enrolled in Each Department

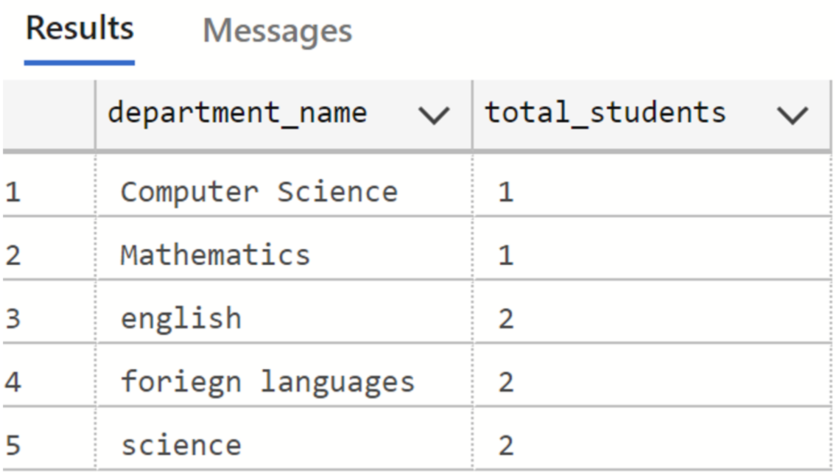
SELECT d.department_name, COUNT(e.student_id) AS total_students

FROM Enrollment e

JOIN Subject sub ON e.subject_id = sub.subject_id

JOIN Department d ON sub.department_id = d.department_id

GROUP BY d.department_name



	department_name	total_students
1	Computer Science	1
2	Mathematics	1
3	english	2
4	foriegn languages	2
5	science	2

Query 2

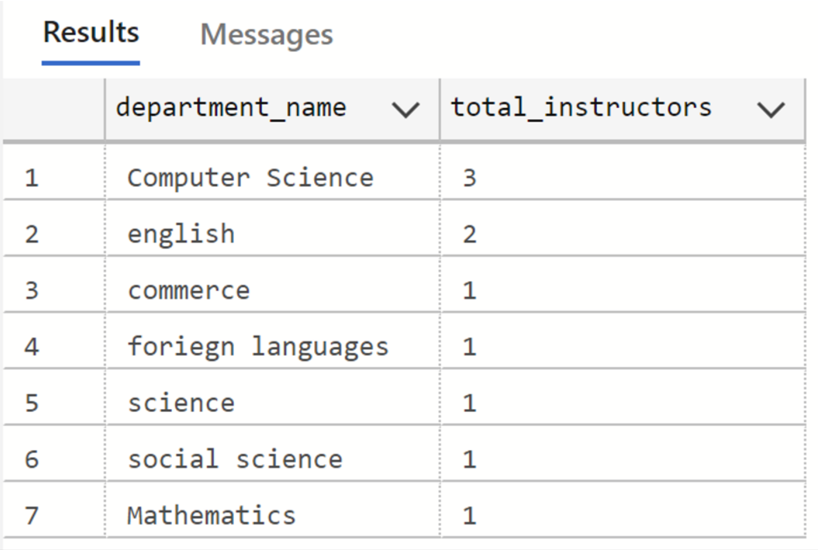
Count the Number of Instructors per Department

SELECT d.department_name, COUNT(i.instructor_id) AS total_instructors

FROM Instructor i

JOIN Department d ON i.department_id = d.department_id

GROUP BY d.department_name;



	department_name	total_instructors
1	Computer Science	3
2	english	2
3	commerce	1
4	foriegn languages	1
5	science	1
6	social science	1
7	Mathematics	1

Query 3:

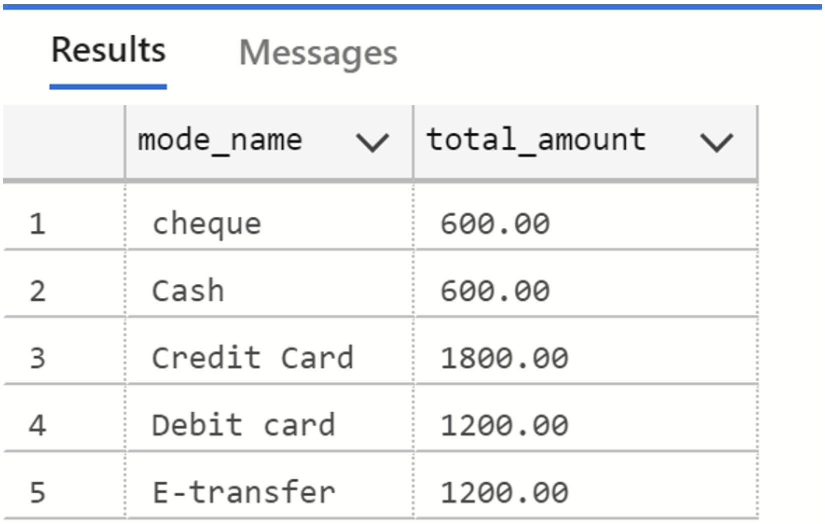
Payments made by students (by payment mode)

SELECT pm.mode_name, SUM(f.amount) AS total_amount

FROM schoolmanagement.paymentmode pm

JOIN schoolmanagement.fees f ON pm.payment_mode_id = f.payment_mode_id

GROUP BY pm.mode_name;

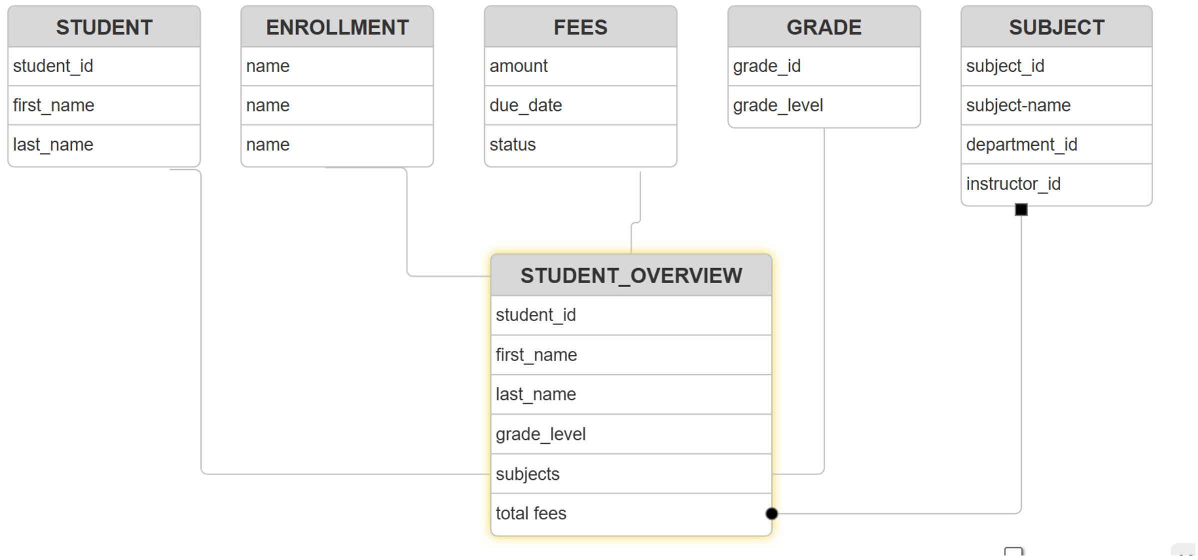


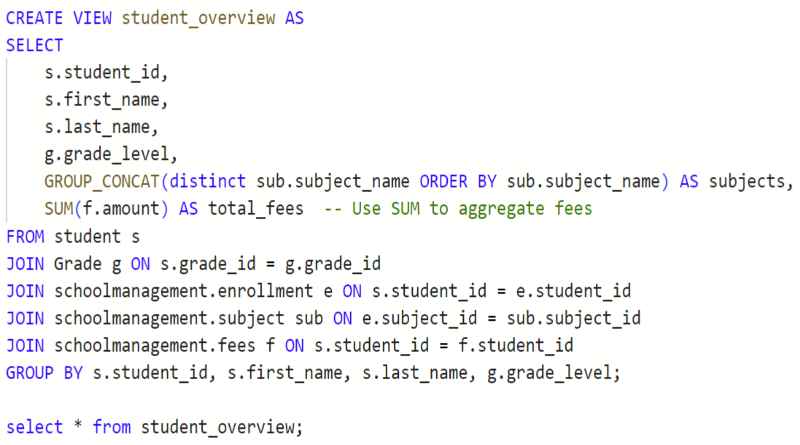
	mode_name	total_amount
1	cheque	600.00
2	Cash	600.00
3	Credit Card	1800.00
4	Debit card	1200.00
5	E-transfer	1200.00

9. VIEWS

VIEW – 1

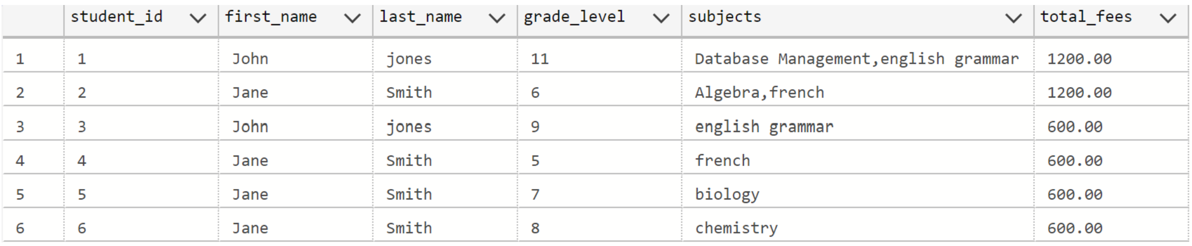
The Entity-Relationship Diagram (ERD) represents the Student Database Management System (DBMS), outlining entities like Student, Enrollment, Exam, Fees, and Results along with their relationships. It ensures structured data organization, linking students to their academic records, instructors, financial transactions, and grading systems for efficient database management.





```
CREATE VIEW student_overview AS
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    g.grade_level,
    GROUP_CONCAT(distinct sub.subject_name ORDER BY sub.subject_name) AS subjects,
    SUM(f.amount) AS total_fees  -- Use SUM to aggregate fees
FROM student s
JOIN Grade g ON s.grade_id = g.grade_id
JOIN schoolmanagement.enrollment e ON s.student_id = e.student_id
JOIN schoolmanagement.subject sub ON e.subject_id = sub.subject_id
JOIN schoolmanagement.fees f ON s.student_id = f.student_id
GROUP BY s.student_id, s.first_name, s.last_name, g.grade_level;

select * from student_overview;
```



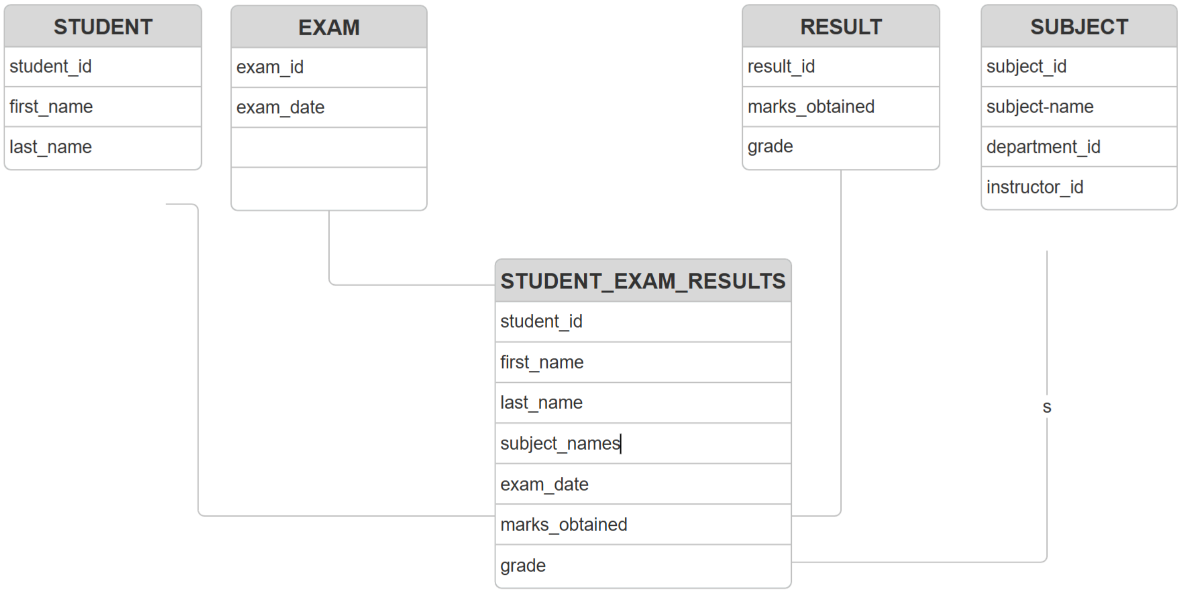
	student_id	first_name	last_name	grade_level	subjects	total_fees
1	1	John	jones	11	Database Management,english grammar	1200.00
2	2	Jane	Smith	6	Algebra,french	1200.00
3	3	John	jones	9	english grammar	600.00
4	4	Jane	Smith	5	french	600.00
5	5	Jane	Smith	7	biology	600.00
6	6	Jane	Smith	8	chemistry	600.00

VIEW 2:

This image shows documentation for a database view named "Student_exam_results" that displays student examination results. Here's the key information:

- Purpose: Shows exam results including date, subject, marks, and grades for students
- Tables used: Student, subject, results, and exam
- Fields included: Student_id, firstname, lastname, subject_name, exam_date, marks_obtained, and grade
- Combines student personal information with their exam performance data

The view appears to be designed for accessing comprehensive student examination records in an educational database system.



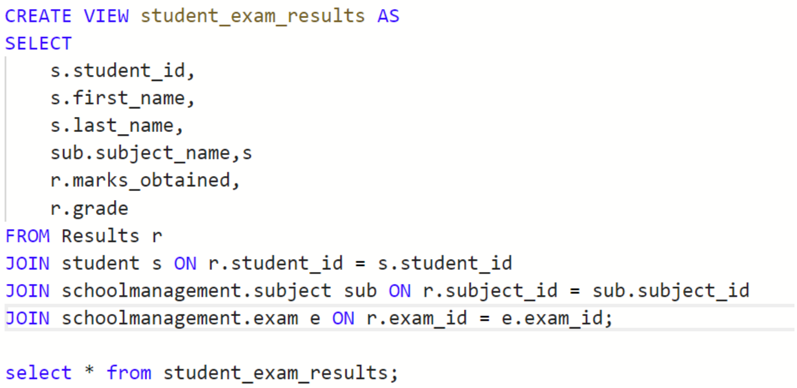
STUDENT
student_id
first_name
last_name

EXAM
exam_id
exam_date

RESULT
result_id
marks_obtained
grade

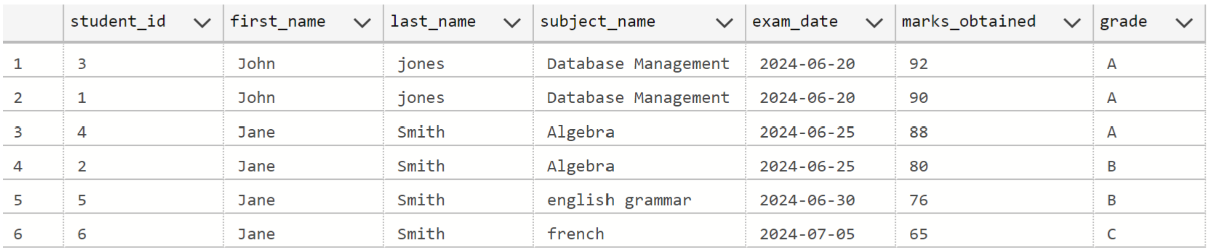
SUBJECT
subject_id
subject-name
department_id
instructor_id

STUDENT_EXAM_RESULTS
student_id
first_name
last_name
subject_names
exam_date
marks_obtained
grade



```
CREATE VIEW student_exam_results AS
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    sub.subject_name,s
    r.marks_obtained,
    r.grade
FROM Results r
JOIN student s ON r.student_id = s.student_id
JOIN schoolmanagement.subject sub ON r.subject_id = sub.subject_id
JOIN schoolmanagement.exam e ON r.exam_id = e.exam_id;

select * from student_exam_results;
```



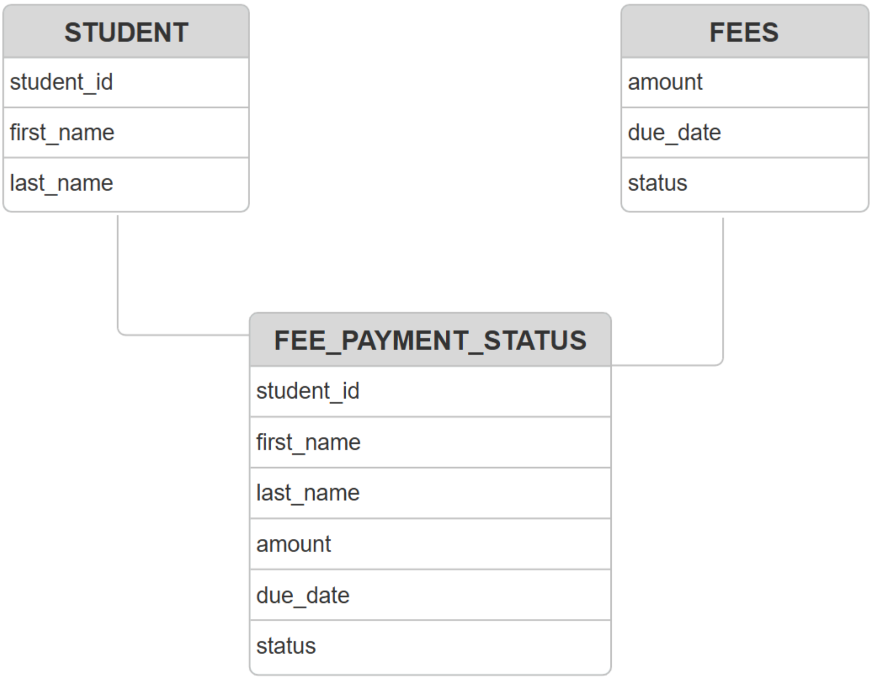
	student_id	first_name	last_name	subject_name	exam_date	marks_obtained	grade
1	3	John	jones	Database Management	2024-06-20	92	A
2	1	John	jones	Database Management	2024-06-20	90	A
3	4	Jane	Smith	Algebra	2024-06-25	88	A
4	2	Jane	Smith	Algebra	2024-06-25	80	B
5	5	Jane	Smith	english grammar	2024-06-30	76	B
6	6	Jane	Smith	french	2024-07-05	65	C

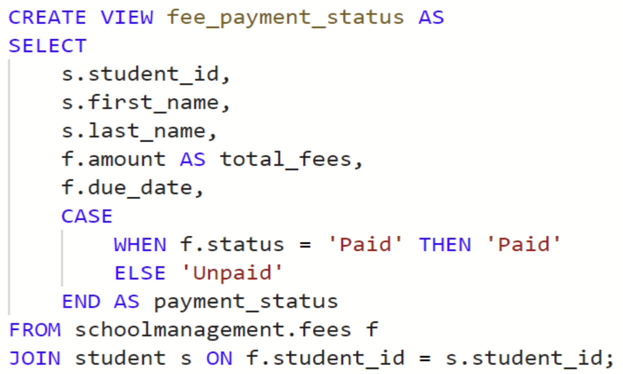
VIEW 3:

"Fee_payment_status" in a student database system. The view combines data from student and fees tables to track payment information:

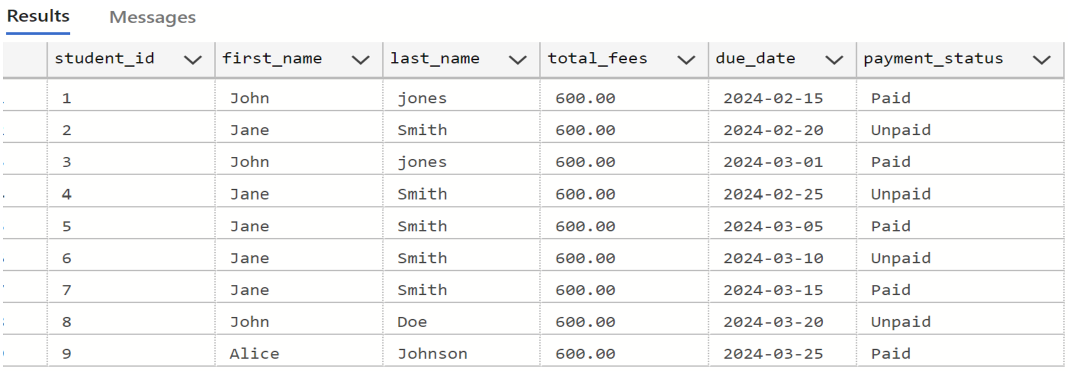
- Purpose: Displays student payment status, amounts due, and payment confirmation
- Tables: Student and fees
- Fields: Student_id, firstname, lastname, amount, due_date, and payment status
- Designed for monitoring student fee payments and tracking due dates

This view appears to be part of a financial tracking system within an educational database.





```sql
CREATE VIEW fee_payment_status AS
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    f.amount AS total_fees,
    f.due_date,
    CASE
        WHEN f.status = 'Paid' THEN 'Paid'
        ELSE 'Unpaid'
    END AS payment_status
FROM schoolmanagement.fees f
JOIN student s ON f.student_id = s.student_id;
```



student_id	first_name	last_name	total_fees	due_date	payment_status
1	John	jones	600.00	2024-02-15	Paid
2	Jane	Smith	600.00	2024-02-20	Unpaid
3	John	jones	600.00	2024-03-01	Paid
4	Jane	Smith	600.00	2024-02-25	Unpaid
5	Jane	Smith	600.00	2024-03-05	Paid
6	Jane	Smith	600.00	2024-03-10	Unpaid
7	Jane	Smith	600.00	2024-03-15	Paid
8	John	Doe	600.00	2024-03-20	Unpaid
9	Alice	Johnson	600.00	2024-03-25	Paid

10.CONCLUSION

In conclusion, a Student Management Database System provides an efficient and organized approach to managing student data. It enables easy storage, retrieval, and updating of student information such as personal details, grades, attendance, and course registrations. The system improves administrative efficiency, reduces the risk of human errors, and enhances communication between students, teachers, and staff. Ultimately, it supports academic institutions in delivering a streamlined and user-friendly experience for both students and administrators.