CASE STUDY ON
SCHOOL DATABASE MANAGEMENT

WRITTEN BY : ABINAYA BABU

MENTOR: DR. JUNAID QAZI

Table of Contents

1. Introduction

1.1 Purpose of the Database Design

The primary goal of the School Management Database System is to enhance the efficiency of school operations by effectively managing student, faculty, exam, and financial data. This ensures seamless data storage, retrieval, and reporting while maintaining data integrity and minimizing redundancy.

1.2 Overview of School Database Management System

This system is designed to manage student records, faculty information, fee transactions, exam results, and subject enrollments. By integrating various entities, it enables smooth school operations, data-driven decision-making, and effective academic tracking.

2. Mission and Objectives

2.1 Mission Statement

“To enhance school management efficiency, improve data-driven decisions, and simplify student, faculty, and financial record management through an integrated database solution.”

2.2 Business Objectives

- Centralize student, faculty, and financial data for easy access.

- Reduce administrative overhead through automation.
- Improve accuracy and consistency in record-keeping.
- Ensure secure and structured storage of all school-related information.

3. Database Design Overview

3.1 Identified Tables

The database consists of the following tables:

- **Department**: Stores details of different school departments.
- **Student**: Contains personal and academic details of students.
- **Instructor**: Maintains faculty details.
- **Subject**: Lists all subjects offered by the school.
- **Exam**: Records exam schedules and maximum marks.
- **Enrollment**: Tracks student registrations in subjects.
- **Fees**: Manages student fee transactions.
- **PaymentMode**: Tracks payment modes and amounts.
- **Grade**: Stores grading criteria and scales.
- **Results**: Maintains student exam scores and grades.

4. Tables and Their Attributes

4.1 Department Table

Description: Stores department details. Each department manages multiple subjects and instructors.

Fields:

- department_id
- department_name

4.2 Student Table

Description: Stores student details including personal information and academic level.

Fields:

- student_id
- first_name
- last_name
- dob
- gender
- address
- contact_number
- email
- grade_level
- grade_id

4.3 Instructor Table

Description: Stores instructor details including contact information and assigned department.

Fields:

- instructor_id
- first_name
- last_name
- contact_number
- email
- department_id

4.4 Subject Table

Description: Stores details of subjects available for enrollment.

Fields:

- subject_id
- subject_name
- department_id
- instructor_id

4.5 Exam Table

Description: Stores information about exams conducted for different subjects.

Fields:

- exam_id

- subject_id
- exam_date
- max_mark

4.6 Enrollment Table

Description: Tracks student enrollments in subjects.

Fields:

- enrollment_id
- student_id
- subject_id
- enrollment_date
- grade_id

4.7 Fees Table

Description: Manages fee transactions for students.

Fields:

- fees_id
- student_id
- amount
- payment_mode_id
- payment_date
- due_date
- status

4.8 PaymentMode Table

Description: Stores payment transaction details like method and amount.

Fields:

- payment_id
- fees_id
- payment_method
- total_amount
- payment_date

4.9 Grade Table

Description: Manages grading levels assigned to students and results.

Fields:

- grade_id
- grade_level

4.10 Results Table

Description: Stores exam results for students.

Fields:

- result_id
- student_id
- exam_id
- marks_obtained

- grade (VARCHAR)

4.11 Classroom Table

Description: Stores details of classrooms and student allocations.

Fields:

- classroom_id
- student_id
- room_number
- capacity

5. Relationships Between Tables

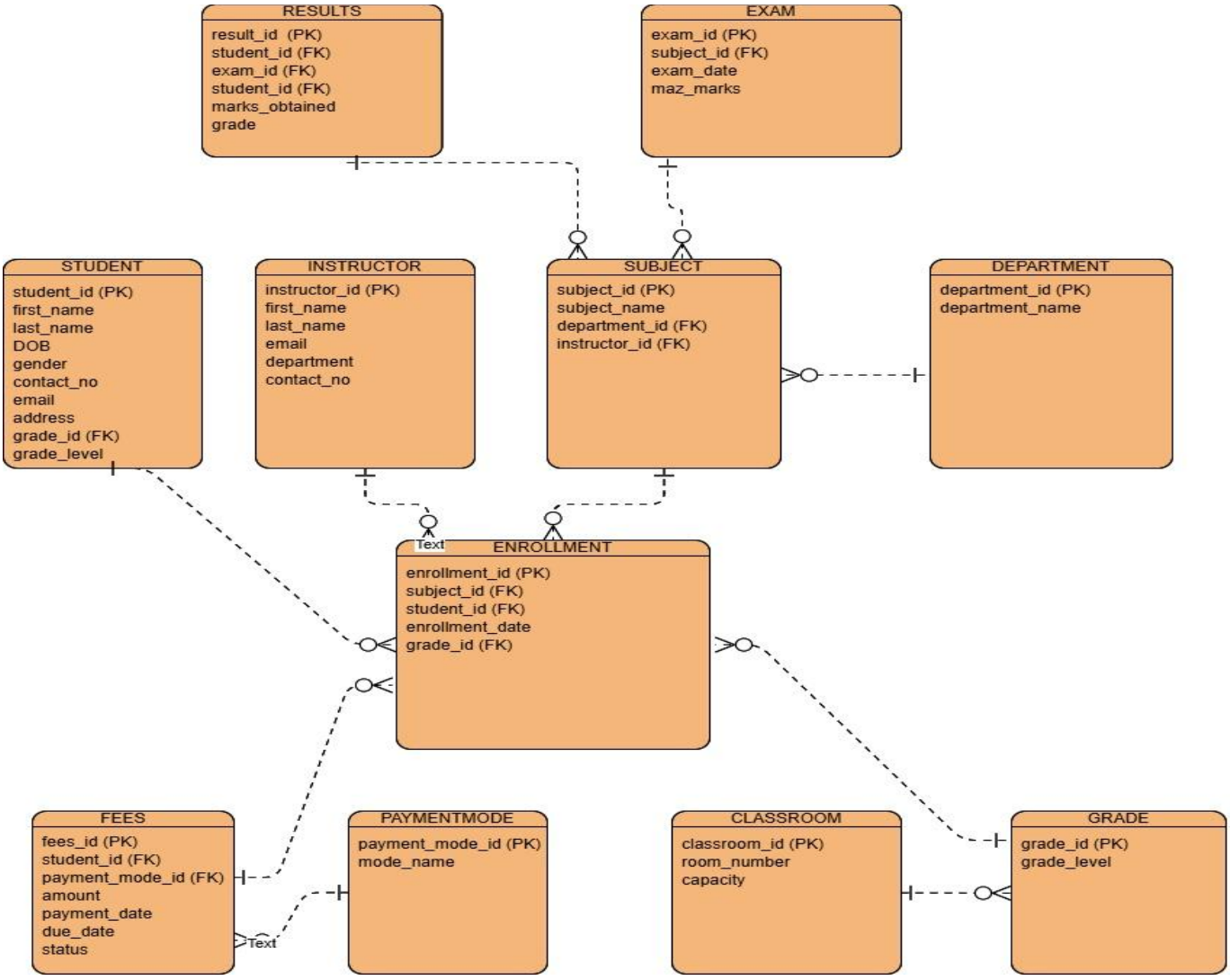
- **Department → Students & Instructors** (One-to-Many)
- **Students → Enrollment** (One-to-Many)
- **Subjects → Enrollment** (One-to-Many)
- **Instructors → Subjects** (One-to-Many)
- **Students → Fees** (One-to-Many)
- **Fees → Payments** (One-to-Many)
- **Students → Results** (One-to-Many)
- **Exams → Results** (One-to-Many)

6. Entity Relationship Diagram (ERD)

6.1 Overview of the ERD

The ERD visually represents how different tables interact, defining primary and foreign key relationships.

6.2 ER-diagram



7. Conclusion

The School Management Database System is designed to enhance efficiency by automating administrative and academic processes. The structured approach improves data integrity, ensures quick data retrieval, and facilitates accurate reporting for decision-making.

8. Appendix

8.1 Table and Description

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.

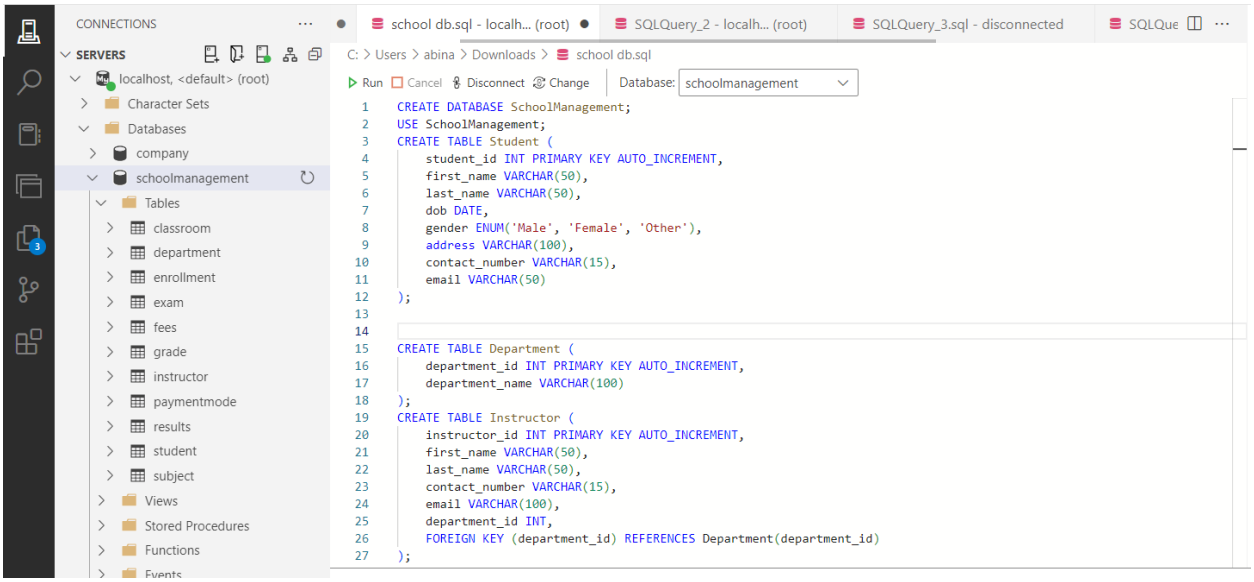
Table 9: Fees

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.

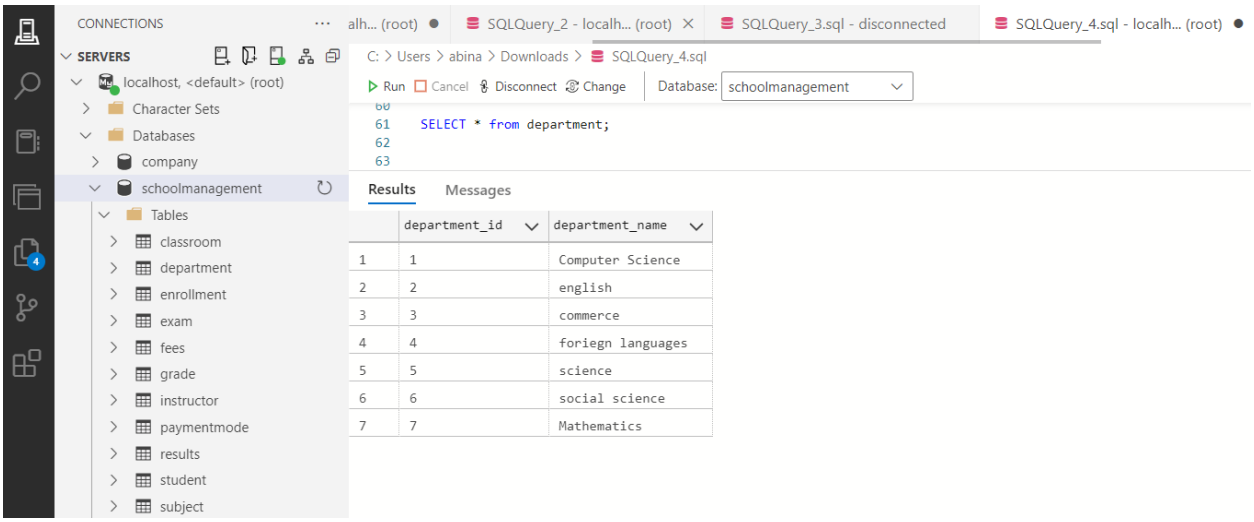
Table 10: Payment Mode

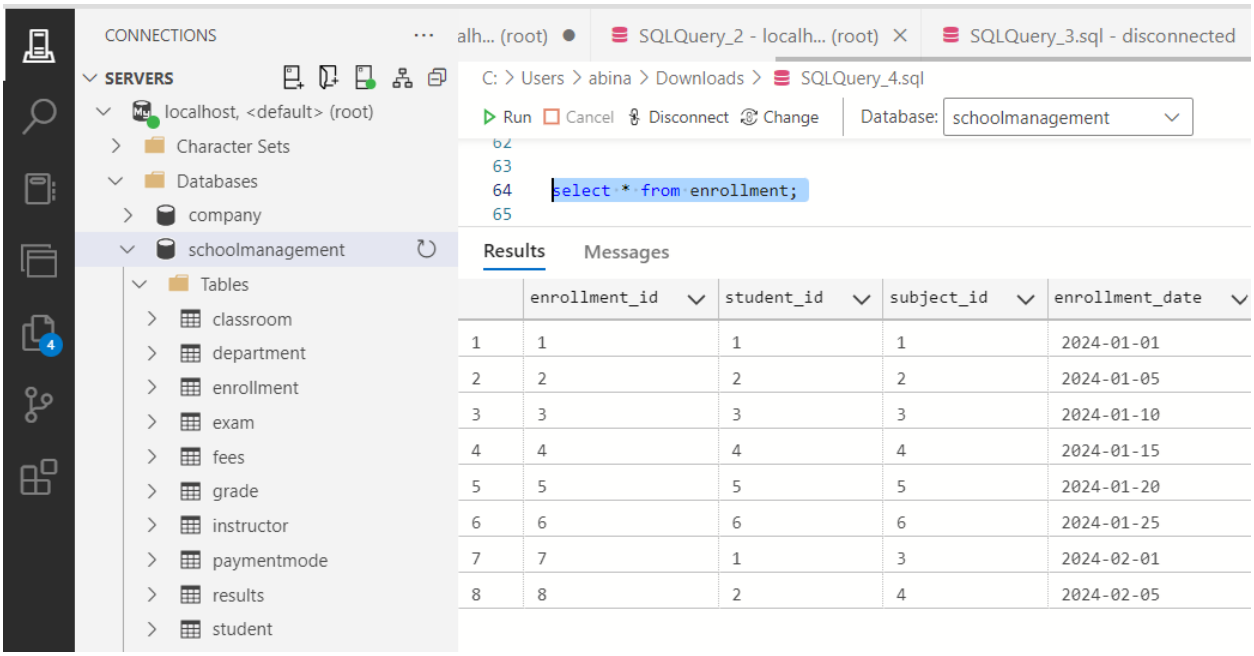
Attribute	Data Type	PK	FK	Description
payment_mode_id	INT	✓	-	Unique identifier for payment methods.
mode_name	VARCHAR(50)	-	-	Type of payment (e.g., Credit Card, Cash).

Database Overview:

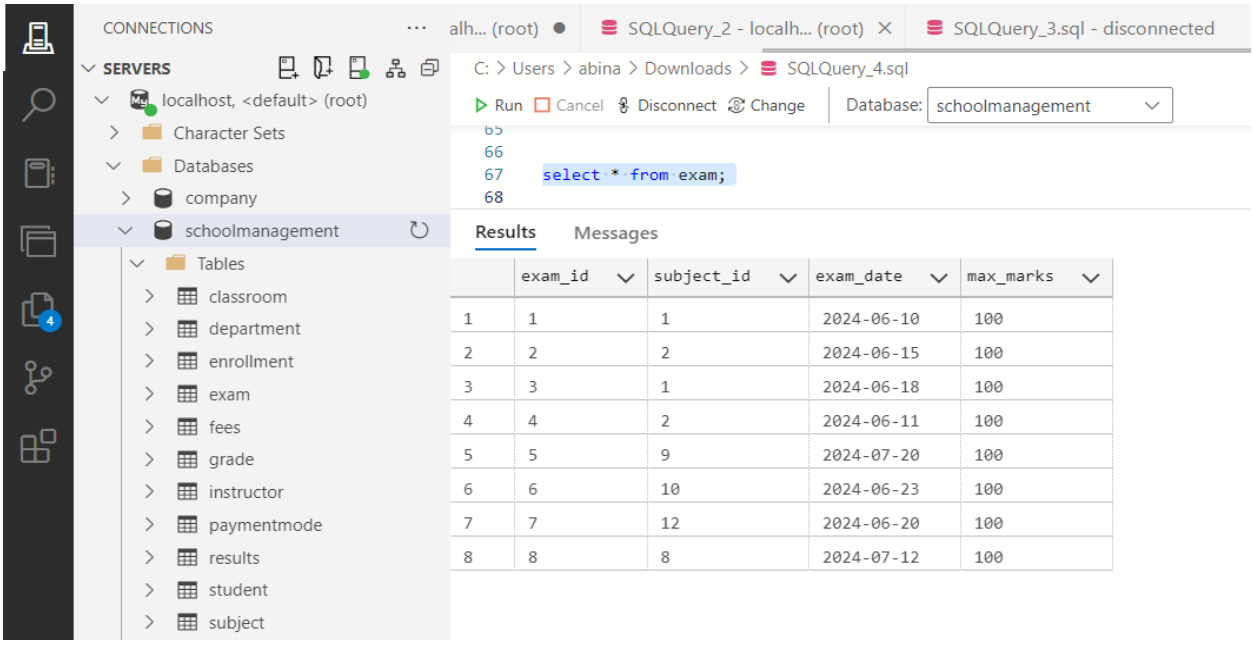

CONNECTIONS
school db.sql - localh... (root)
SQLQuery_2 - localh... (root)
SQLQuery_3.sql - disconnected
SERVERS
localhost, <default> (root)
Character Sets
Databases
company
schoolmanagement
Tables
classroom
department
enrollment
exam
fees
grade
instructor
paymentmode
results
student
subject
Views
Stored Procedures
Functions
C: > Users > abina > Downloads > school db.sql
Run
Cancel
Disconnect
Change
Database: schoolmanagement
CREATE DATABASE SchoolManagement;
USE SchoolManagement;
CREATE TABLE Student (
student_id INT PRIMARY KEY AUTO_INCREMENT,
first_name VARCHAR(50),
last_name VARCHAR(50),
dob DATE,
gender ENUM('Male', 'Female', 'Other'),
address VARCHAR(100),
contact_number VARCHAR(15),
email VARCHAR(50)
);
CREATE TABLE Department (
department_id INT PRIMARY KEY AUTO_INCREMENT,
department_name VARCHAR(100)
);
CREATE TABLE Instructor (
instructor_id INT PRIMARY KEY AUTO_INCREMENT,
first_name VARCHAR(50),
last_name VARCHAR(50),
contact_number VARCHAR(15),
email VARCHAR(100),
department_id INT,
FOREIGN KEY (department_id) REFERENCES Department(department_id)
);

Tables:

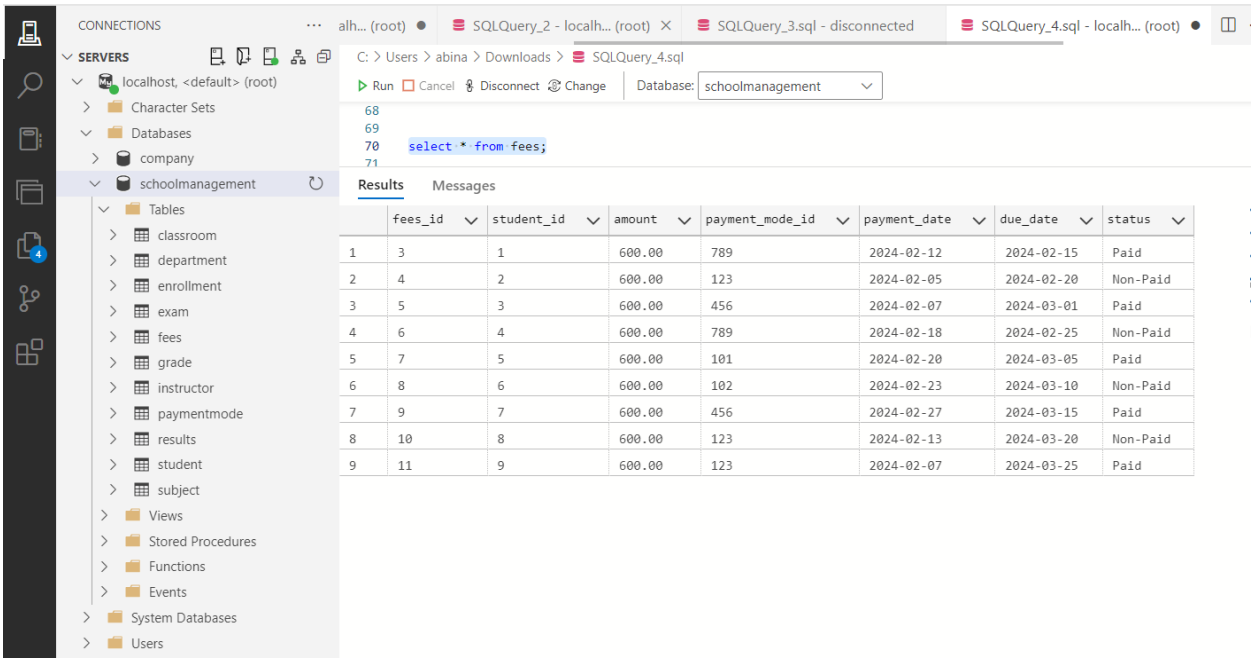

CONNECTIONS
SQLQuery_2 - localh... (root)
SQLQuery_3.sql - disconnected
SQLQuery_4.sql - localh... (root)
SERVERS
localhost, <default> (root)
Character Sets
Databases
company
schoolmanagement
Tables
classroom
department
enrollment
exam
fees
grade
instructor
paymentmode
results
student
subject
C: > Users > abina > Downloads > SQLQuery_4.sql
Run
Cancel
Disconnect
Change
Database: schoolmanagement
SELECT * from department;
Results
Messages
department_id
department_name
1 1 Computer Science
2 2 english
3 3 commerce
4 4 foriegn languages
5 5 science
6 6 social science
7 7 Mathematics



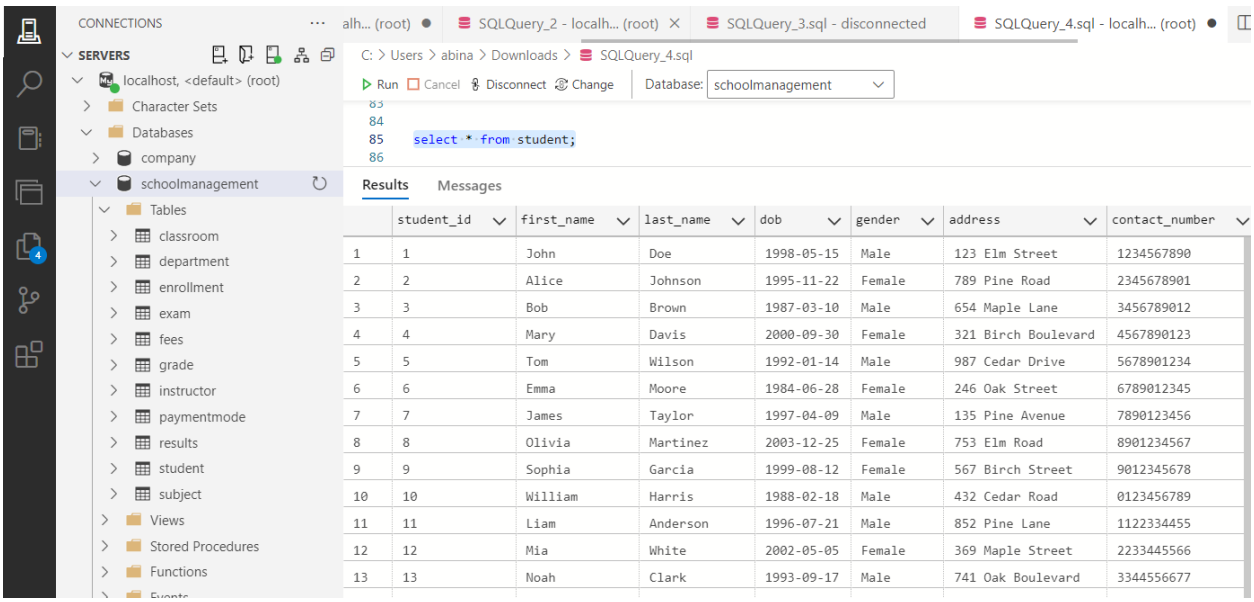
	enrollment_id	student_id	subject_id	enrollment_date
1	1	1	1	2024-01-01
2	2	2	2	2024-01-05
3	3	3	3	2024-01-10
4	4	4	4	2024-01-15
5	5	5	5	2024-01-20
6	6	6	6	2024-01-25
7	7	1	3	2024-02-01
8	8	2	4	2024-02-05



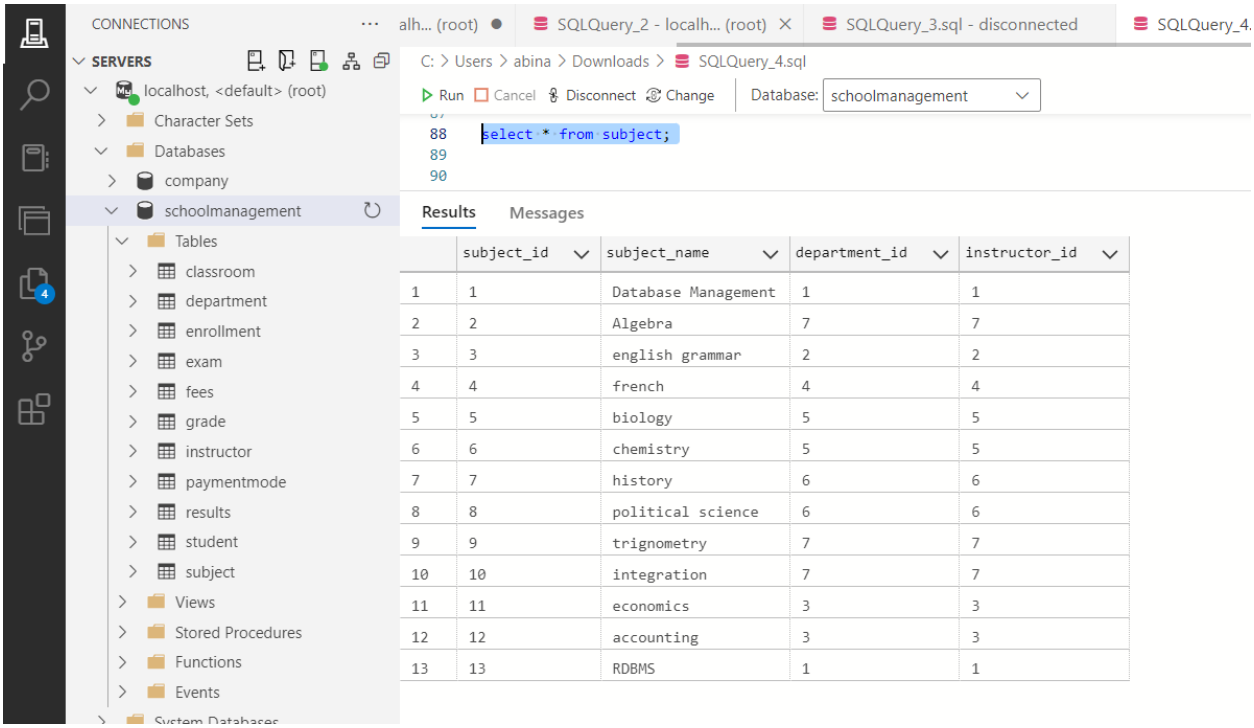
	exam_id	subject_id	exam_date	max_marks
1	1	1	2024-06-10	100
2	2	2	2024-06-15	100
3	3	1	2024-06-18	100
4	4	2	2024-06-11	100
5	5	9	2024-07-20	100
6	6	10	2024-06-23	100
7	7	12	2024-06-20	100
8	8	8	2024-07-12	100



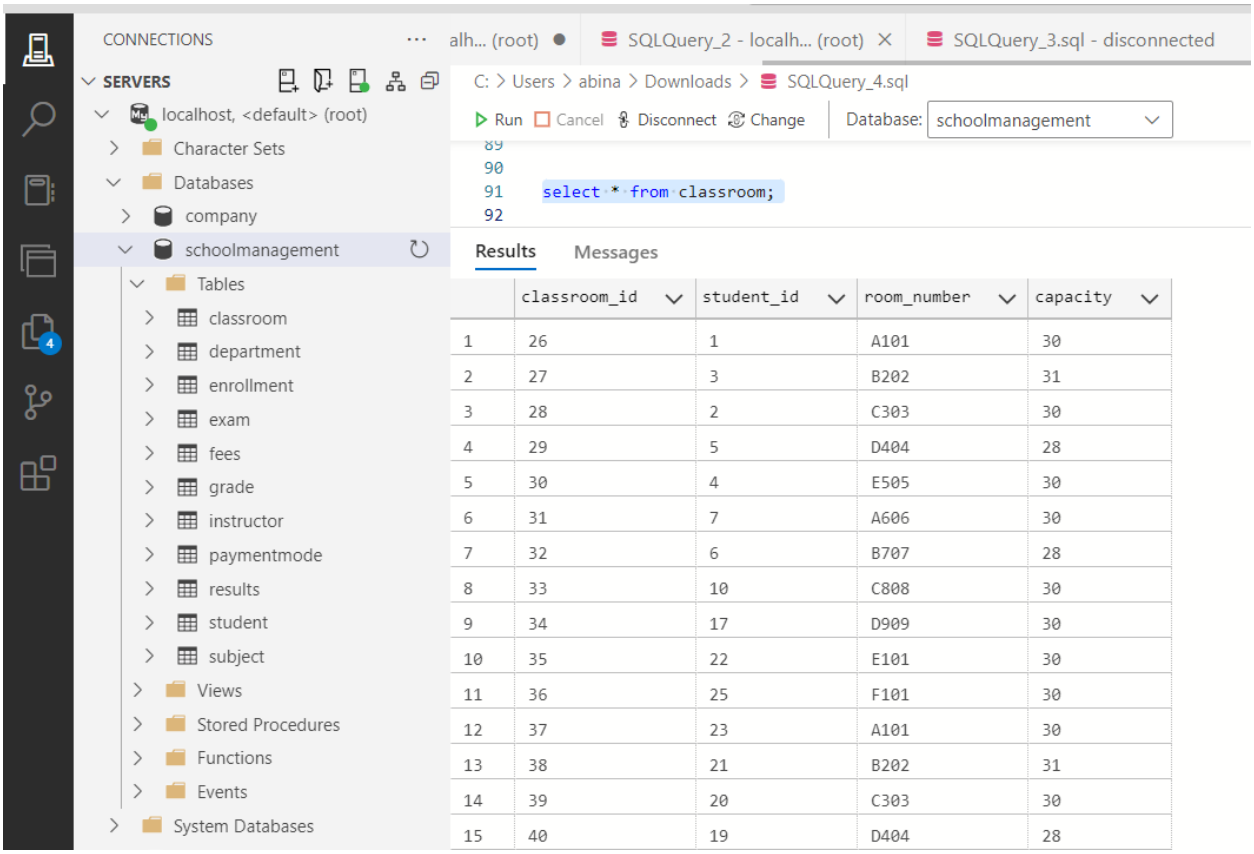
	fees_id	student_id	amount	payment_mode_id	payment_date	due_date	status
1	3	1	600.00	789	2024-02-12	2024-02-15	Paid
2	4	2	600.00	123	2024-02-05	2024-02-20	Non-Paid
3	5	3	600.00	456	2024-02-07	2024-03-01	Paid
4	6	4	600.00	789	2024-02-18	2024-02-25	Non-Paid
5	7	5	600.00	101	2024-02-20	2024-03-05	Paid
6	8	6	600.00	102	2024-02-23	2024-03-10	Non-Paid
7	9	7	600.00	456	2024-02-27	2024-03-15	Paid
8	10	8	600.00	123	2024-02-13	2024-03-20	Non-Paid
9	11	9	600.00	123	2024-02-07	2024-03-25	Paid



	student_id	first_name	last_name	dob	gender	address	contact_number
1	1	John	Doe	1998-05-15	Male	123 Elm Street	1234567890
2	2	Alice	Johnson	1995-11-22	Female	789 Pine Road	2345678901
3	3	Bob	Brown	1987-03-10	Male	654 Maple Lane	3456789012
4	4	Mary	Davis	2000-09-30	Female	321 Birch Boulevard	4567890123
5	5	Tom	Wilson	1992-01-14	Male	987 Cedar Drive	5678901234
6	6	Emma	Moore	1984-06-28	Female	246 Oak Street	6789012345
7	7	James	Taylor	1997-04-09	Male	135 Pine Avenue	7890123456
8	8	Olivia	Martinez	2003-12-25	Female	753 Elm Road	8901234567
9	9	Sophia	Garcia	1999-08-12	Female	567 Birch Street	9012345678
10	10	William	Harris	1988-02-18	Male	432 Cedar Road	0123456789
11	11	Liam	Anderson	1996-07-21	Male	852 Pine Lane	1122334455
12	12	Mia	White	2002-05-05	Female	369 Maple Street	2233445566
13	13	Noah	Clark	1993-09-17	Male	741 Oak Boulevard	3344556677



	subject_id	subject_name	department_id	instructor_id
1	1	Database Management	1	1
2	2	Algebra	7	7
3	3	english grammar	2	2
4	4	french	4	4
5	5	biology	5	5
6	6	chemistry	5	5
7	7	history	6	6
8	8	political science	6	6
9	9	trignometry	7	7
10	10	integration	7	7
11	11	economics	3	3
12	12	accounting	3	3
13	13	RDBMS	1	1



	classroom_id	student_id	room_number	capacity
1	26	1	A101	30
2	27	3	B202	31
3	28	2	C303	30
4	29	5	D404	28
5	30	4	E505	30
6	31	7	A606	30
7	32	6	B707	28
8	33	10	C808	30
9	34	17	D909	30
10	35	22	E101	30
11	36	25	F101	30
12	37	23	A101	30
13	38	21	B202	31
14	39	20	C303	30
15	40	19	D404	28

8.2 Queries, Views

Sample queries, views for quick data access, and test cases verifying database performance.

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;
```

Results | Messages

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

Results | Messages

	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

VIEWS:

1. This view will provide an overview of students, including their grade level, enrolled subjects, and the total fees they owe.

```
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

2. This view will show the exam results for students, including the exam date, subject, marks obtained, and grade.

```sql
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

3. This view will show the payment status for each student, including the amount due and whether they have made the payment.

```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;
```

Results Messages

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