Project Title (Blood Bank System)

Semester Project

G-5	**Name**	**Roll Number**
	ANAS	Fa-2021/BSCS-144
	MOAWWAZ	Fa-2021/BSCS-159

Session: FA-2021

Semester: 4^{ths}

Section: D

Submitted to: Mr. Abdul Rehman

Date: 19-june-2023

Department of Computer Science

Lahore Garrison University

Introduction:

ABC School, a prestigious educational institution, utilizes a robust School Management System to effectively handle various aspects of its operations. The system incorporates several tables to manage student information, course offerings, teacher assignments, attendance tracking, fee management, salary administration, exam scheduling, and grading. Students like John Doe, assigned a student ID, are enrolled in specific courses such as Science, while Emily Smith, with a student ID pursues her studies in the English course. The Subjects table lists subjects like Mathematics, Science, and English that are taught by specialized teachers. For instance, Jane Johnson, teacher, imparts her expertise in Mathematics, while Mark Davis, teacher, handles Science, and Sarah Williams, teacher, imparts knowledge in English. Through the Enrollments table, student-course associations are established, ensuring accurate tracking of students' academic progress. The Classrooms table manages room assignments, providing details about classrooms' locations and capacities. Attendance records are maintained in the Attendance table, documenting students' presence or absence for specific courses and dates. Fees and payments are efficiently tracked through the Fees table, while the Salary table records salary information for teachers. The Exams table facilitates scheduling of exams, such as Science and English exams, and the Grades table captures students' performance through assigned grades. Overall, this School Management System ensures streamlined operations and effective management of students, teachers, courses, attendance, fees, salaries, exams, and grades at ABC School.

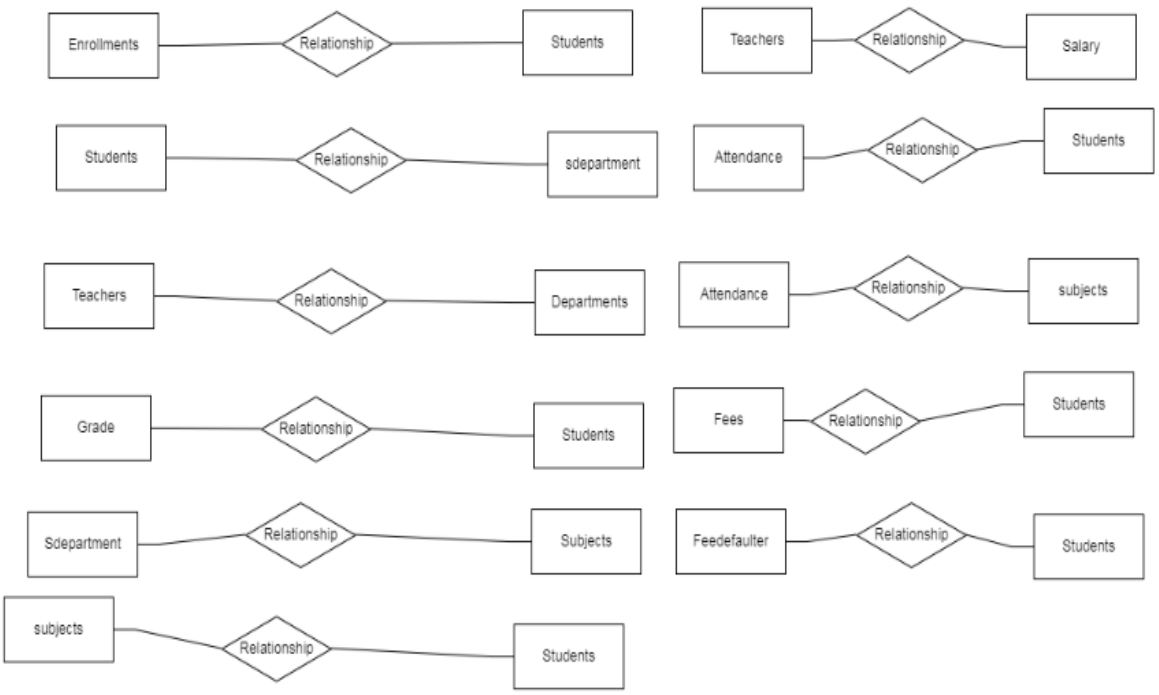
Step 1- Entity Identification

- Students
- Teachers
- Sdepartments
- Departments
- Fees
- Fee defaulter
- Grades
- Exams
- Attendance
- Enrollments
- Salary
- Subjects

Step 2- Relationship Identification

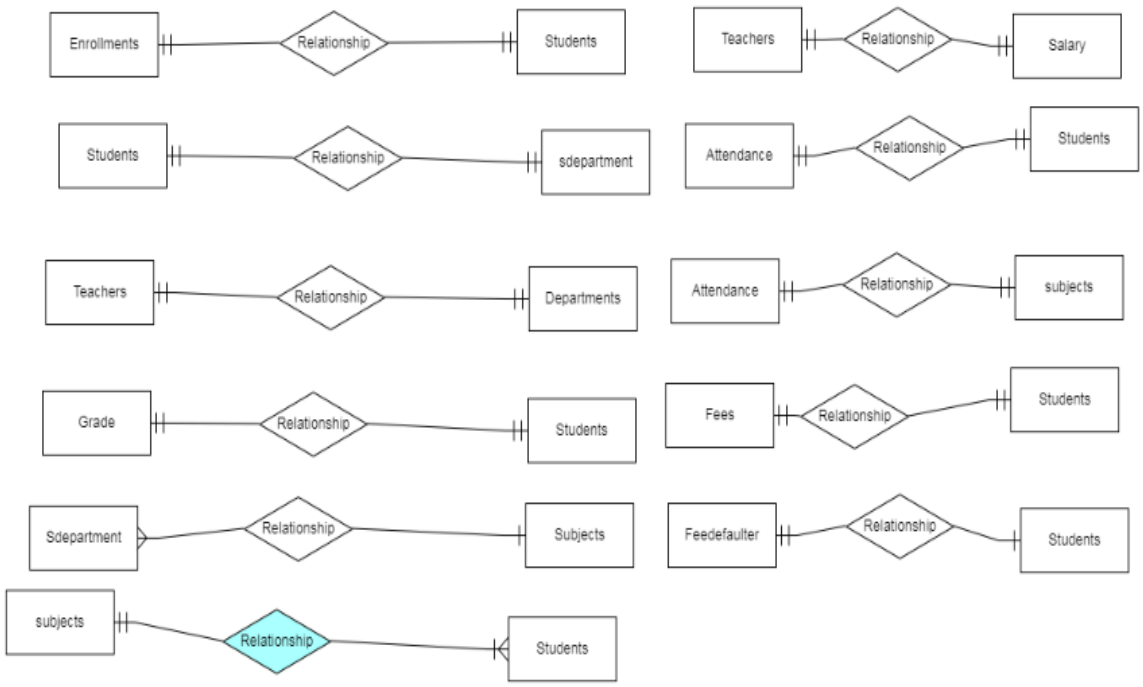
- The student table has a foreign key sdepartment_id which references to sdepartments primary key sdepartment_id.
- The teacher table has a foreign key department_id which references to departments table primary key department_id.

- The enrollment table has a foreign key student_id which references to students table primary key student_id.
- The enrollment table has a foreign key subject_id which references to subject table primary key subject_id.
- The enrollment table has a foreign key sdepartment_id which references to sdepartments table primary key sdepartment_id.
- The attendance table has a foreign key student_id which references to students table primary key student_id.
- The attendance table has a foreign key subject_id which references to subject table primary key subject_id.
- The Feedefaulter table has a foreign key student_id which references to students table primary key student_id.
- The salary table has a foreign key department_id which references to departments table primary key department_id.
- The salary table has a foreign key teacher_id which references to teachers table primary key teacher_id.
- The grades table has a foreign key student_id which references to students table primary key student_id.
- The grades table has a foreign key subject_id which references to subject table primary key subject_id.



Step 3- Cardinality Identification

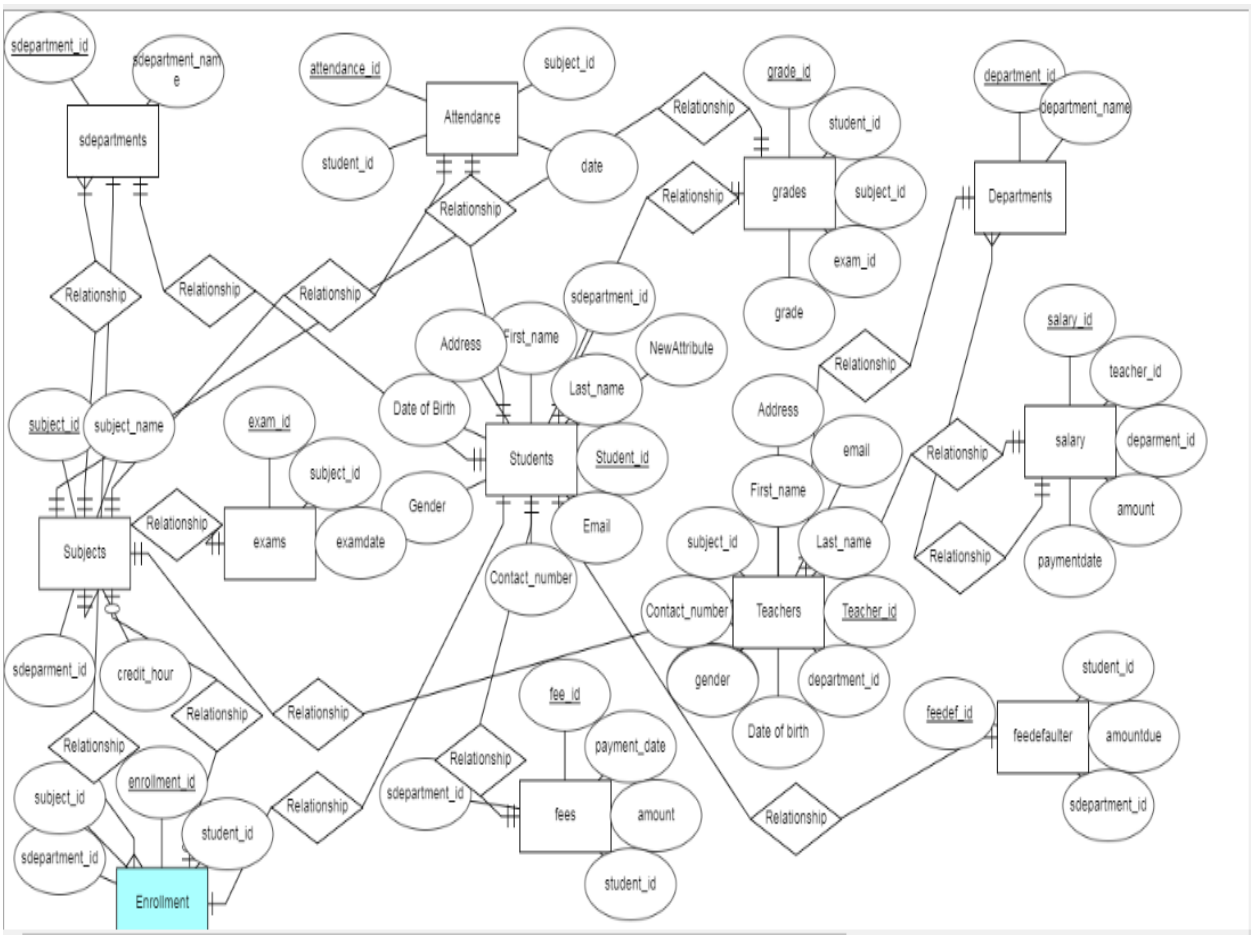
- The relationship between the student and sdeparment table is one-to-one.
- The relationship between the subject and student table is many-to-one.
- The relationship between the fees and student table is one-to-one.
- The relationship between the Feedefaulter and student table is one-to-one.
- The relationship between the exam and subject table is one-to-one.
- The relationship between the teacher and salary table is one-to-one.
- The relationship between the teacher and department table is one-to-one.
- The relationship between the enrollments and student table is one-to-one.
- The relationship between the subject and enrollment table is many-to-one.
- The relationship between the grade and student table is one-to-one.
- The relationship between the grade and subject table is one-to-one.
- The relationship between the teacher and subject table is one-to-many.
- The relationship between the attendance and student table is one-to-one.
- The relationship between the attendance and subject table is one-to-one.
- The relationship between the subject and student is many-to-one.



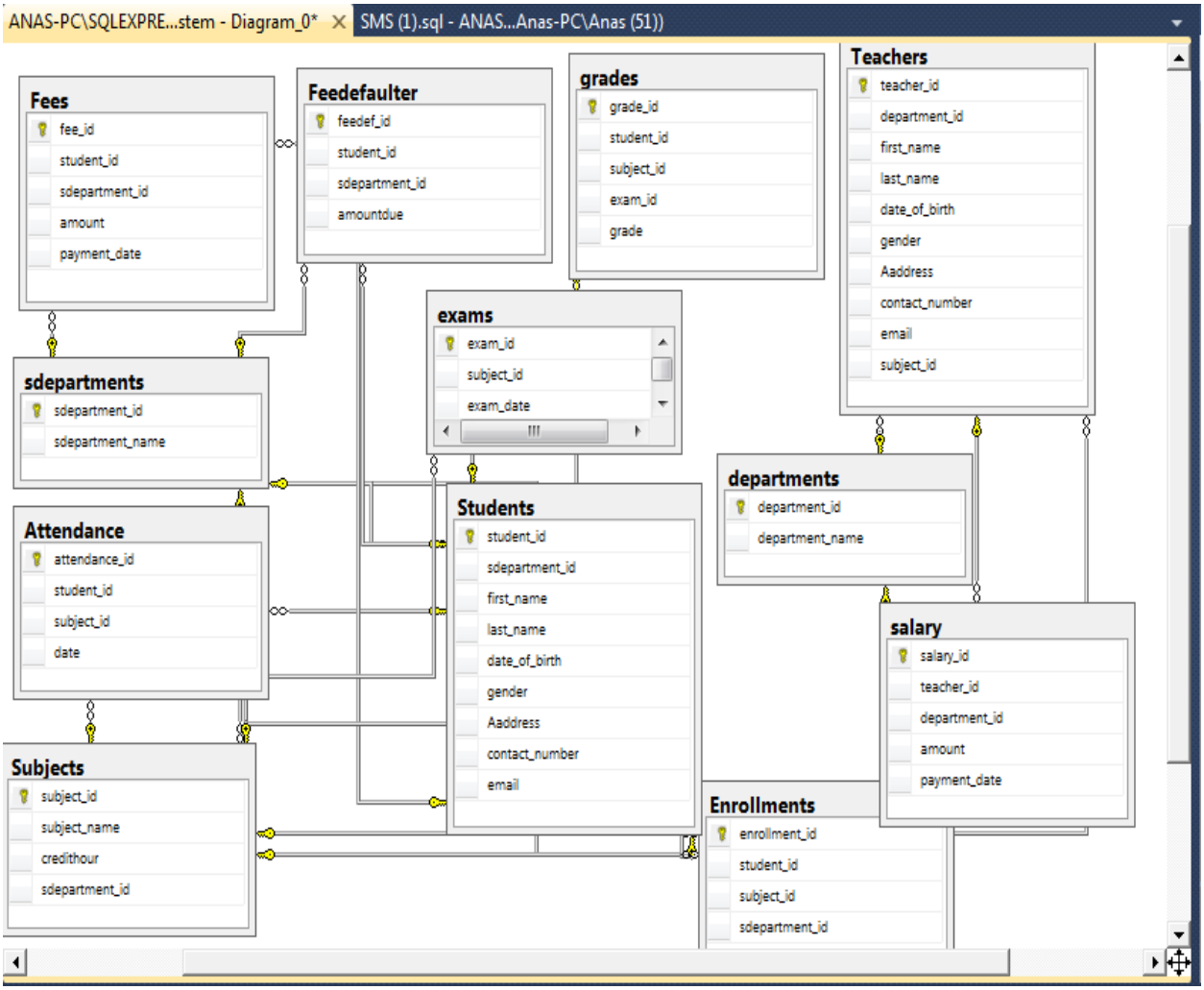
Step 4- Identify Attributes

Students: student_id, sdepartment_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email
Sdepartments: sdeparment_id, sdepartment_name
Teachers: teacher_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email, subject_id, deparment_id
Departments: department_id, department_name
Salary: salary_id, teacher_id, department_id, amount, payment_date
Subjects: `subject_id, subject_name, credithour, sdepartment_id`
Enrollments: enrollment_id, student_id, subject_id, sdepartment_id
Attendance: attendance_id, student_id, subject_id, date
Fees: fee_id, student_id, sdepartment_id, amount, payment_date
Feedefaulter: feedef_id, student_id, sdepartment_id, amountdue
Exams: exam_id, subject_id, exam_date
Grades: grade_id, subject_id, exam_id, grade, `student_id`

Step 5- Create the ERD Diagram



Step 6- Convert ERD to Tables in DBMS


ANAS-PC\SQLEXPRE...stem - Diagram_0*
SMS (1).sql - ANAS...Anas-PC\Anas (51))
Fees
fee_id
student_id
sdepartment_id
amount
payment_date
Feedefaulter
feedef_id
student_id
sdepartment_id
amountdue
grades
grade_id
student_id
subject_id
exam_id
grade
Teachers
teacher_id
department_id
first_name
last_name
date_of_birth
gender
Aaddress
contact_number
email
subject_id
exams
exam_id
subject_id
exam_date
sdepartments
sdepartment_id
sdepartment_name
departments
department_id
department_name
Attendance
attendance_id
student_id
subject_id
date
Students
student_id
sdepartment_id
first_name
last_name
date_of_birth
gender
Aaddress
contact_number
email
salary
salary_id
teacher_id
department_id
amount
payment_date
Subjects
subject_id
subject_name
credithour
sdepartment_id
Enrollments
enrollment_id
student_id
subject_id
sdepartment_id

Sr No	Topic	Queries
1.	CREATE TABLE Statement	**10**
2.	PRIMARY KEY and FOREIGN KEY	**10**
3.	AUTO INCREMENT	**10**
4.	ALTER TABLE Statement (ADD Column, MODIFY DATATYPE, RENAME COLUMN, DROP COLUMN)	**50**
5.	INSERT INTO Statement	**10**
6.	SELECT and DISTINCT Statement	**20**
7.	WHERE Clause using AND, OR and NOT Operators	**50**
8.	ORDER BY Statement	**25**
9.	ORDER BY using AND, OR and NOT Operators	**25**
10.	GROUP BY Statement	**25**
11.	GROUP BY using AND, OR, NOT Operators and Group by	**25**
12.	Subqueries	**30**
13.	Subqueries	**30**
14.	Aggregate functions MAX, MIN, SUM, COUNT, and AVG.	**20**
15.	Aggregate functions using logical Operators and Group by	**30**
16.	INNER Joins	**20**
17.	INNER Joins using logical Operators, Group by and Order by	**30**
18.	LEFT JOIN	**20**
19.	RIGHT JOIN	**20**
20.	FULL OUTER JOIN	**20**
21.	Stored Procedures without parameter	**25**
22.	Stored Procedures with parameter	**25**
23.	Stored Procedures with parameter using logical Operators and Group by	**30**
24.	DML Triggers INSERT	**20**
25.	DML Triggers UPDATE	**20**
26.	DML Triggers DELETE	**20**
27.	VIEW Statement	**10**
28.	VIEW Statement using logical Operators	**30**
29.	Single-Row Functions UPPER, LOWER, INITCAP, CONCAT, LENGTH, SUBSTR using logical operators	**50**
30.	Single-Row Functions INSTR, TRIM, REPLACE, ROUND, TRUNC using logical operators	**50**
31.	Transaction COMMIT and ROLLBACK	**20**
32.	Exception handling - Try Catch	**20**

1. CREATE TABLE Statement – 10 Queries

1	Create table Students	CREATE TABLE Students (student_id INT IDENTITY(1,1) PRIMARY KEY, sdepartment_id INT FOREIGN KEY REFERENCES sdepartments(sdepartment_id), first_name VARCHAR(50), last_name VARCHAR(50), date_of_birth DATE, gender VARCHAR(10), Aaddress VARCHAR(100), contact_number VARCHAR(20), email VARCHAR(50));
2	Create table departments	CREATE TABLE departments (department_id INT IDENTITY(1,1) PRIMARY KEY, department_name VARCHAR(50));
3	Create table Subjects	CREATE TABLE Subjects (subject_id INT IDENTITY(1,1) PRIMARY KEY, subject_name VARCHAR(50), credithour int, sdepartment_id INT FOREIGN KEY REFERENCES sdepartments(sdepartment_id));
4	Create table Teachers	CREATE TABLE Teachers (teacher_id INT IDENTITY(1,1) PRIMARY KEY, department_id INT FOREIGN KEY REFERENCES departments(department_id), first_name VARCHAR(50), last_name VARCHAR(50), date_of_birth DATE, gender VARCHAR(10), Aaddress VARCHAR(100), contact_number VARCHAR(20), email VARCHAR(50), subject_id INT FOREIGN KEY REFERENCES Subjects(subject_id),);
5	Create table salary	CREATE TABLE salary (salary_id INT IDENTITY(1,1) PRIMARY KEY, teacher_id INT FOREIGN KEY REFERENCES teachers(teacher_id), department_id INT FOREIGN KEY REFERENCES departments(department_id), amount INT, payment_date DATE);
6	Create table Enrollments	CREATE TABLE Enrollments (enrollment_id INT IDENTITY(1,1) PRIMARY KEY, student_id INT FOREIGN KEY REFERENCES Students(student_id),

		subject_id INT FOREIGN KEY REFERENCES subjects(subject_id), sdepartment_id INT FOREIGN KEY REFERENCES sdepartments(sdepartment_id));
7	Create table Attendance	CREATE TABLE Attendance (attendance_id INT IDENTITY(1,1) PRIMARY KEY, student_id INT FOREIGN KEY REFERENCES Students(student_id), subject_id INT FOREIGN KEY REFERENCES subjects(subject_id), date DATE,);
8	Create table Fees	CREATE TABLE Fees (fee_id INT IDENTITY(1,1) PRIMARY KEY, student_id INT FOREIGN KEY REFERENCES Students(student_id), sdepartment_id INT FOREIGN KEY REFERENCES sdepartments(sdepartment_id), amount INT, payment_date DATE);
9	Create table Feedefaulter	CREATE TABLE Feedefaulter(feedef_id INT IDENTITY(1,1) PRIMARY KEY, student_id INT FOREIGN KEY REFERENCES Students(student_id), sdepartment_id INT FOREIGN KEY REFERENCES sdepartments(sdepartment_id), amountdue INT);
10	Create table grades	CREATE TABLE grades (grade_id INT IDENTITY(1,1) PRIMARY KEY, student_id INT FOREIGN KEY REFERENCES students(student_id), subject_id INT FOREIGN KEY REFERENCES subjects(subject_id), exam_id INT FOREIGN KEY (exam_id) REFERENCES exams(exam_id), grade VARCHAR(20));

2. PRIMARY KEY and FOREIGN KEY – 10 Queries

1	Student primary key	student_id INT PRIMARY KEY,
2	department primary key	department_id INT PRIMARY KEY,
3	subject primary key	subject_id INT PRIMARY KEY,
4	teacher primary key	teacher_id INT PRIMARY KEY,
5	salary primary key	salary_id INT PRIMARY KEY,

6	sdepartment FOREIGN key	sdepartment_id INT FOREIGN KEY REFERENCES sdepartments(sdepartment_id),
7	department FOREIGN key	department_id INT FOREIGN KEY REFERENCES departments(department_id),
8	teacher_FOREIGN key	teacher_id INT FOREIGN KEY REFERENCES teachers(teacher_id),
9	student FOREIGN key	student_id INT FOREIGN KEY REFERENCES Students(student_id),
10	subject FOREIGN key	subject_id INT FOREIGN KEY REFERENCES subjects(subject_id),

3. AUTO INCREMENT – 10 Queries

1	Auto inc in student	student_id INT IDENTITY(1,1) PRIMARY KEY,
2	Auto inc in department	department_id INT IDENTITY(1,1) PRIMARY KEY,
3	Auto inc in sdepartment	sdepartment_id INT IDENTITY(1,1) PRIMARY KEY,
4	Auto inc in subject	subject_id INT IDENTITY(1,1) PRIMARY KEY,
5	Auto inc in teacher	teacher_id INT IDENTITY(1,1) PRIMARY KEY,
6	Auto inc in salary	salary_id INT IDENTITY(1,1) PRIMARY KEY,
7	Auto inc in enrollment	enrollment_id INT IDENTITY(1,1) PRIMARY KEY,
8	Auto inc in attendance	attendance_id INT IDENTITY(1,1) PRIMARY KEY,
9	Auto inc in fee	fee_id INT IDENTITY(1,1) PRIMARY KEY,
10	Auto inc in exam	exam_id INT IDENTITY(1,1) PRIMARY KEY,

5. INSERT INTO Statement – 10 Queries

1	Inserting records into the Students table	INSERT INTO Students (first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email) VALUES ('John', 'Doe', '2000-01-03', 'Male', '123 Main St', '123-456-7890', 'john.doe@example.com');
2	Inserting records into the Subjects table	INSERT INTO Subjects (subject_name, credithour) VALUES ('Mathematics', 3);
3	Inserting records into the departments table	INSERT INTO departments (department_name) VALUES ('Computer Science');
4	Inserting records into the sdepartments table	INSERT INTO sdepartments (sdepartment_name) VALUES ('Computer Science Department');

5	Inserting records into the Teachers table	INSERT INTO Teachers (first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email) VALUES ('Jane', 'Smith', '1999-05-10', 'Female', '456 Elm St', '987-654-3210', 'jane.smith@example.com');
6	Inserting records into the Fees table	INSERT INTO Fees (amount, payment_date) VALUES (1000, '2023-06-05');
7	Inserting records into the FeeDefaulter table	INSERT INTO FeeDefaulter (amountdue) VALUES (500);
8	Inserting records into the Salary table	INSERT INTO Salary (amount, payment_date) VALUES (5500, '2023-06-15');
9	Inserting records into the Exams table	INSERT INTO Exams (exam_date) VALUES ('2023-07-03');
10	Inserting records into the Grades table	INSERT INTO Grades (grade) VALUES ('A');

4. ALTER TABLE Statement (ADD Column, MODIFY DATATYPE, RENAME COLUMN, DROP COLUMN) – 50 Queries

1	Rename fees primary key column	EXEC sp_rename 'Fees.fee_id', 'fee_record_id', 'COLUMN';
2	Add middle name field	ALTER TABLE Students ADD middle_name VARCHAR(50);
3	Modify student department name size.	ALTER TABLE sdepartments alter column sdepartment_name VARCHAR(100);
4	Rename subject credit hour column.	exec sp_rename 'Subjects.credithour', 'credit_hour', 'COLUMN';

5	Remove unnecessary teacher address columns.	`ALTER TABLE Teachers DROP COLUMN Aaddress;`
6	Add late count column to attendance.	`ALTER TABLE attendance ADD late_count INT;`
7	Modify fees amount data type.	`ALTER TABLE fees ALTER COLUMN amount decimal;`
8	Rename fee defaulter primary key column	`EXEC sp_rename 'Feedefaulter.feedef_id', 'fees_defaulter_id', 'COLUMN';`
9	Add duration column to exams table	`ALTER TABLE departments ADD description TEXT`
10	Remove unnecessary teacher email columns.	`ALTER TABLE Teachers DROP COLUMN email;`

11	Add duration column to exams.	`ALTER TABLE exams ADD duration INT;`
12	Add column 'previousnumbers' to table 'sdepartments' as INT.	`ALTER TABLE sdepartments ADD previousnumbers INT;`
13	Modify last name data type	`ALTER TABLE Students alter column last_name VARCHAR(100);`
14	. Rename department name column	`EXEC sp_rename 'departments.department_name', 'dept_name', 'COLUMN';`
15	. Remove sdepartment name column	`ALTER TABLE sdepartments DROP COLUMN sdepartment_name;`
16	. Add semester column to subjects	`ALTER TABLE Subjects ADD semester VARCHAR(20);`
17	. Modify amount due data type.	`ALTER TABLE Feedefaulter alter column amountdue DECIMAL;`
18	Rename teacher ID column	`EXEC sp_rename 'Teachers.teacher_id', 'instructor_id', 'COLUMN';`
19	. Remove date column from attendance	`ALTER TABLE attendance DROP COLUMN date;`

20	. Add joining date column to teachers.	`ALTER TABLE Teachers ADD joining_date DATE;`
21	Add payment mode column (fees table	`ALTER TABLE fees ADD payment_mode VARCHAR(50);`
22	) Modify salary amount data type	`ALTER TABLE salary alter column amount DECIMAL;`
23	Rename subject name to course name	`EXEC sp_rename 'Subjects.subject_name', 'course_name', 'COLUMN';`
24	Drop exam date column (exams table)	`ALTER TABLE exams DROP COLUMN exam_date;`
25	Add grade percentage column (grades table)	`ALTER TABLE grades ADD grade_percentage DECIMAL;`
26	Modify student gender data type	`ALTER TABLE Students alter column gender varchar(20)`
27	Rename department id to dept id	`EXEC sp_rename 'departments.department_id', 'dept_id', 'COLUMN';`
28	Drop contact number column (Teachers table)	`ALTER TABLE Teachers DROP COLUMN contact_number;`
29	Add start time column (attendance table)	`ALTER TABLE attendance ADD start_time TIME;`
30	Add credit limit column (fees table)	`ALTER TABLE fees ADD credit_limit INT;`
31	Modify amount due data type (Feedefaulter table)	`ALTER TABLE Feedefaulter alter column amountdue DECIMAL;`
32	Rename grade id to grade record id	`EXEC sp_rename 'Grades.grade_id', 'grade_record_id', 'COLUMN';`
33	Drop payment date column (salary table	`ALTER TABLE salary DROP COLUMN payment_date;`
34	) Add max marks column (exams table	`ALTER TABLE exams ADD max_marks INT;`
35	) Add grade date column (grades table)	`ALTER TABLE grades ADD grade_date DATE;`
36	Modify student last name data type	`ALTER TABLE Students alter column last_name VARCHAR(100);`
37	Rename exam id to exam record id	`EXEC sp_rename 'Exams.exam_id', 'exam_record_id', 'COLUMN';`
38	Drop sdepartment name column (sdepartments table)	`ALTER TABLE sdepartments DROP COLUMN sdepartment_name;`
39	Add semester column (Subjects table)	`ALTER TABLE Subjects ADD semester VARCHAR(20);`
40	Modify amount due data type (Feedefaulter table)	`ALTER TABLE Feedefaulter alter column amountdue DECIMAL;`
41	Rename attendance id to attendance record id	`EXEC sp_rename 'Attendance.attendance_id', 'attendance_record_id', 'COLUMN';`
42	Drop date column from attendance table	`ALTER TABLE attendance DROP COLUMN date;`

43	Add joining date column to Teachers table	`ALTER TABLE Teachers ADD joining_date DATE;`
44	Add payment mode column to fees table	`ALTER TABLE fees ADD payment_mode VARCHAR(50);`
45	Modify amount data type in salary table	`ALTER TABLE salary alter column amount DECIMAL;`
46	Rename enrollment id to enrollment record id	`EXEC sp_rename 'Enrollments.enrollment_id', 'enrollment_record_id', 'COLUMN';`
47	Drop exam date column from exams table	`ALTER TABLE exams DROP COLUMN exam_date;`
48	Add grade percentage column to grades table	`ALTER TABLE grades ADD grade_percentage DECIMAL;`
49	Modify amount data type in fees table	`ALTER TABLE fees alter column amount decimal;`
50	Rename sdepartment name to student department name	`EXEC sp_rename 'StudentDepartments.sdepartment_name', 'student_department_name', 'COLUMN';`

6. SELECT and DISTINCT Statement – 20 Queries

1	Students: Retrieve first name and last name..	`SELECT first_name, last_name FROM Students;`
2	Subjects: Retrieve subject name and credit hour.	`SELECT subject_name, credithour FROM Subjects;`
3	Departments: Retrieve department id and department name.	`SELECT department_id, department_name FROM departments;`
4	Sdepartments: Retrieve sdepartment id and sdepartment name.	`SELECT sdepartment_id, sdepartment_name FROM sdepartments;`
5	Teachers: Retrieve first name, last name, and email.	`SELECT first_name, last_name, email FROM Teachers;`
6	Enrollments: Retrieve enrollment id and student id.	`SELECT enrollment_id, student_id FROM Enrollments;`
7	Attendance: Retrieve attendance id, student id, and subject id.	`SELECT attendance_id, student_id, subject_id FROM Attendance;`
8	Fees: Retrieve fee id, student id, sdepartment id, and amount	`SELECT fee_id, student_id, sdepartment_id, amount FROM Fees;`
9	FeeDefaulter: Retrieve feedef id, student id, and sdepartment id.	`SELECT feedef_id, student_id, sdepartment_id FROM FeeDefaulter;`

10	Salary: Retrieve salary id, teacher id, department id, and amount	`SELECT salary_id, teacher_id, department_id, amount FROM salary;`
11	Students: Retrieve distinct sdepartment_id	`SELECT DISTINCT sdepartment_id FROM Students;`
12	Subjects: Retrieve distinct subject_id.	`SELECT DISTINCT subject_id FROM Subjects;`
13	Departments: Retrieve distinct department_name.	`SELECT DISTINCT department_name FROM departments;`
14	Sdepartments: Retrieve distinct sdepartment_name.	`SELECT DISTINCT sdepartment_name FROM sdepartments;`
15	Teachers: Retrieve distinct gender.	`SELECT DISTINCT gender FROM Teachers;`
16	Enrollments: Retrieve distinct student_id.	`SELECT DISTINCT student_id FROM Enrollments;`
17	Attendance: Retrieve distinct subject_id	`SELECT DISTINCT subject_id FROM Attendance;`
18	Fees: Retrieve distinct sdepartment_id.	`SELECT DISTINCT sdepartment_id FROM Fees;`
19	FeeDefaulter: Retrieve distinct amountdue	`SELECT DISTINCT amountdue FROM FeeDefaulter;`
20	Salary: Retrieve distinct department_id	`SELECT DISTINCT department_id FROM salary;`

7. WHERE Clause using AND, OR and NOT Operators – **50 Queries**

1	Students: Retrieve information about students named John Smith	`SELECT * FROM Students WHERE first_name = 'John' AND last_name = 'Smith';`
2	. Students: Retrieve students from the Computer Science department	`SELECT * FROM Students WHERE sdepartment_id IN (SELECT sdepartment_id FROM sdepartments WHERE sdepartment_name = 'Computer Science');`
3	. Students: Retrieve students born on January 1, 2000.	`SELECT * FROM Students WHERE date_of_birth = '2000-01-01';`
4	Students: Retrieve male students.	`SELECT * FROM Students WHERE gender = 'Male';`
5	Students: Retrieve students with last names starting with 'S' or 'T'	`SELECT * FROM Students WHERE last_name LIKE 'S%' OR last_name LIKE 'T%';`
6	. Students: Retrieve students with addresses containing the word 'Street'	`SELECT * FROM Students WHERE Aaddress LIKE '%Street%';`

7	. Students: Retrieve students with contact numbers starting with '1' and email addresses containing '@example.com'.	`SELECT * FROM Students WHERE contact_number LIKE '1%' AND email LIKE '%@example.com';`
8	Students: Retrieve students with a date of birth that is not December 25, 2002	`SELECT * FROM Students WHERE date_of_birth <> '2002-12-25';`
9	. Students: Retrieve students named Sarah or with the last name Johnson.	`SELECT * FROM Students WHERE first_name = 'Sarah' OR last_name = 'Johnson';`
10	Students: Retrieve students not from the Physics department.	`SELECT * FROM Students WHERE sdepartment_id NOT IN (SELECT sdepartment_id FROM sdepartments WHERE sdepartment_name = 'Physics');`
11	Students: Retrieve students whose contact numbers do not contain the digit 5.	`SELECT * FROM Students WHERE contact_number NOT LIKE '%5%';`
12	Students: Retrieve students with a non-null email address.	`SELECT * FROM Students WHERE email IS NOT NULL;`
13	Students: Retrieve male students born between January 1, 1995, and December 31, 2000	`SELECT * FROM Students WHERE gender = 'Male' AND date_of_birth BETWEEN '1995-01-01' AND '2000-12-31';`
14	. Students: Retrieve students whose first names start with 'A' or 'B' and last names end with 'son'.	`SELECT * FROM Students WHERE (first_name LIKE 'A%' OR first_name LIKE 'B%') AND last_name LIKE '%son';`
15	Students: Retrieve students with addresses that do not contain the word 'Apartment'.	`SELECT * FROM Students WHERE Aaddress NOT LIKE '%Apartment%';`
16	Students: Retrieve students with a null contact number.	`SELECT * FROM Students WHERE contact_number IS NULL;`
17	Students: Retrieve students not from the Computer Science or Mathematics departments.	`SELECT * FROM Students WHERE sdepartment_id NOT IN (SELECT sdepartment_id FROM sdepartments WHERE sdepartment_name IN ('Computer Science', 'Mathematics'));`
18	Students: Retrieve students whose first names contain the letter 'o' and last names contain the letter 'e'.	`SELECT * FROM Students WHERE first_name LIKE '%o%' AND last_name LIKE '%e%';`
19	Students: Retrieve students with a null date of birth.	`SELECT * FROM Students WHERE date_of_birth IS NULL;`
20	Students: Retrieve students whose contact numbers do not start with '1' or email addresses do not contain '@example.com'.	`SELECT * FROM Students WHERE contact_number NOT LIKE '1%' OR email NOT LIKE '%@example.com';`
21	Students: Retrieve female students born before January 1, 2003	`SELECT * FROM Students WHERE gender = 'Female' AND date_of_birth < '2003-01-01';`
22	. Students: Retrieve students with email addresses that do not contain 'gmail' and have a non-null contact number	`SELECT * FROM Students WHERE email NOT LIKE '%gmail%' AND contact_number IS NOT NULL;`
23	. Students: Retrieve students whose last names are not 'Smith' or 'Johnson'.	`SELECT * FROM Students WHERE last_name <> 'Smith' AND last_name <> 'Johnson';`

24	Students: Retrieve students whose first names are not 'John' and last names are not 'Doe'	`SELECT * FROM Students WHERE first_name <> 'John' AND last_name <> 'Doe';`
25	. Students: Retrieve male students born after January 1, 1999	`SELECT * FROM Students WHERE gender = 'Male' AND date_of_birth > '1999-01-01';`
26	. Students: Retrieve students with a non-null contact number and null email address.	`SELECT * FROM Students WHERE contact_number IS NOT NULL AND email IS NULL;`
27	Students: Retrieve students from the Physics department with a non-null contact number.	`SELECT * FROM Students WHERE sdepartment_id IN (SELECT sdepartment_id FROM sdepartments WHERE sdepartment_name = 'Physics') AND contact_number IS NOT NULL;`
28	Students: Retrieve students whose last names are not 'Taylor' or have a null email address.	`SELECT * FROM Students WHERE last_name <> 'Taylor' OR email IS NULL;`
29	Students: Retrieve students whose first names are not 'Daniel' and have a non-null contact number	`SELECT * FROM Students WHERE first_name <> 'Daniel' AND contact_number IS NOT NULL;`
30	. Students: Retrieve female students born after January 1, 1997.	`SELECT * FROM Students WHERE gender = 'Female' AND date_of_birth > '1997-01-01';`
31	Students: Retrieve students with a non-null email address and born before January 1, 1993.	`SELECT * FROM Students WHERE email IS NOT NULL AND date_of_birth < '1993-01-01';`
32	Departments: Retrieve departments whose names start with 'C' and have more than 50 students enrolled.	`SELECT * FROM departments WHERE department_name LIKE 'C%' AND department_id IN (SELECT sdepartment_id FROM Students GROUP BY sdepartment_id HAVING COUNT(*) > 50);`
33	Departments: Retrieve departments with IDs not equal to 5 and names containing 'Science'.	`SELECT * FROM departments WHERE department_id <> 5 AND department_name LIKE '%Science%';`
34	Student Departments: Retrieve student departments whose names end with 'ics' and have corresponding entries in the Students table.	`SELECT * FROM sdepartments WHERE sdepartment_name LIKE '%ics' AND sdepartment_id IN (SELECT sdepartment_id FROM Students);`
35	Subjects: Retrieve subjects with credit hours less than or equal to 3 and names starting with 'M'.	`SELECT * FROM Subjects WHERE credithour <= 3 AND subject_name LIKE 'M%';`
36	Teachers: Retrieve teachers whose last names contain 'a' and contact numbers start with '9'.	`SELECT * FROM Teachers WHERE last_name LIKE '%a%' AND contact_number LIKE '9%';`
37	Teachers: Retrieve teachers born between January 1, 1980, and December 31, 1990, who are not assigned to teach subject ID 3.	`SELECT * FROM Teachers WHERE date_of_birth BETWEEN '1980-01-01' AND '1990-12-31' AND teacher_id NOT IN (SELECT teacher_id FROM Subjects WHERE subject_id = 3);`
38	Teachers: Retrieve teachers whose gender is not 'Male' or last names are not 'Smith'.	`SELECT * FROM Teachers WHERE gender <> 'Male' OR last_name <> 'Smith';`
39	Teachers: Retrieve teachers with a null contact number or email	`SELECT * FROM Teachers WHERE contact_number IS NULL OR email LIKE '%@example.com';`

	addresses containing '@example.com'.	
40	Teachers: Retrieve female teachers assigned to departments with IDs 1, 2, or 3.	`SELECT * FROM Teachers WHERE department_id IN (1, 2, 3) AND gender = 'Female';`
41	Teachers: Retrieve teachers born before January 1, 1985, who are assigned to teach subject ID 2.	`SELECT * FROM Teachers WHERE date_of_birth < '1985-01-01' AND teacher_id IN (SELECT teacher_id FROM Subjects WHERE subject_id = 2);`
42	Enrollments: Retrieve enrollment records for student ID 1 and subject ID 2.	`SELECT * FROM Enrollments WHERE student_id = 1 AND subject_id = 2;`
43	Enrollments: Retrieve enrollment records for students with IDs 4, 5, or 6, excluding enrollment IDs 1, 2, and 3.	`SELECT * FROM Enrollments WHERE enrollment_id NOT IN (1, 2, 3) AND student_id IN (4, 5, 6);`
44	Enrollments: Retrieve enrollment records for students with IDs 1, 2, or 3, excluding subject IDs 4, 5, and 6.	`SELECT * FROM Enrollments WHERE student_id IN (1, 2, 3) AND subject_id NOT IN (4, 5, 6);`
45	Enrollments: Retrieve enrollment records for students not with IDs 1, 2, or 3, or with subject IDs 4, 5, or 6.	`SELECT * FROM Enrollments WHERE student_id NOT IN (1, 2, 3) OR subject_id IN (4, 5, 6);`
46	Attendance: Retrieve attendance records for student ID 1 and subject ID 2.	`SELECT * FROM attendance WHERE student_id = 1 AND subject_id = 2;`
47	Attendance: Retrieve attendance records for students with IDs 1, 2, or 3, excluding subject IDs 4, 5, and 6.	`SELECT * FROM attendance WHERE student_id IN (1, 2, 3) AND subject_id NOT IN (4, 5, 6);`
48	Attendance: Retrieve attendance records for students not with IDs 1, 2, or 3, or with subject IDs 4, 5, or 6.	`SELECT * FROM attendance WHERE student_id NOT IN (1, 2, 3) OR subject_id IN (4, 5, 6);`
49	Fees: Retrieve fee records for student ID 1 and student department ID 2.	`SELECT * FROM fees WHERE student_id = 1 AND sdepartment_id = 2;`
50	Fees: Retrieve fee records for students with IDs 1, 2, or 3, with an amount greater than 100.	`SELECT * FROM fees WHERE student_id IN (1, 2, 3) AND amount > 100;`

8. ORDER BY Statement – 25 Queries

1	Students: Retrieve all student records sorted by first name in ascending order.	`SELECT *` `FROM Students` `ORDER BY first_name ASC;`

2	Students: Retrieve all student records sorted by last name in descending order.	`SELECT * FROM Students ORDER BY last_name DESC;`
3	Subjects: Retrieve all subject records sorted by subject name in ascending order.	`SELECT * FROM Subjects ORDER BY subject_name ASC;`
4	Subjects: Retrieve all subject records sorted by credit hours in descending order.	`SELECT * FROM Subjects ORDER BY credithour DESC;`
5	Departments: Retrieve all department records sorted by department name in ascending order.	`SELECT * FROM departments ORDER BY department_name ASC;`
6	Departments: Retrieve all department records sorted by department ID in descending order.	`SELECT * FROM departments ORDER BY department_id DESC;`
7	SDepartments: Retrieve all student department records sorted by student department name in ascending order.	`SELECT * FROM sdepartments ORDER BY sdepartment_name ASC;`
8	SDepartments: Retrieve all student department records sorted by student department ID in descending order.	`SELECT * FROM sdepartments ORDER BY sdepartment_id DESC;`
9	Teachers: Retrieve all teacher records sorted by first name in ascending order.	`SELECT * FROM Teachers ORDER BY first_name ASC;`
10	Teachers: Retrieve all teacher records sorted by last name in descending order.	`SELECT * FROM Teachers ORDER BY last_name DESC;`
11	Enrollments: Retrieve all enrollment records sorted by enrollment ID in ascending order.	`SELECT * FROM Enrollments ORDER BY enrollment_id ASC;`
12	Enrollments: Retrieve all enrollment records sorted by student ID in descending order.	`SELECT * FROM Enrollments ORDER BY student_id DESC;`
13	Attendance: Retrieve all attendance records sorted by attendance ID in ascending order.	`SELECT * FROM Attendance ORDER BY attendance_id ASC;`
14	Attendance: Retrieve all attendance records sorted by student ID in descending order.	`SELECT * FROM Attendance ORDER BY student_id DESC;`
15	Fees: Retrieve all fee records sorted by fee ID in ascending order.	`SELECT * FROM Fees`

		`ORDER BY fee_id ASC;`
16	Fees: Retrieve all fee records sorted by amount in descending order.	`SELECT * FROM Fees ORDER BY amount DESC;`
17	FeeDefaulter: Retrieve all fee defaulter records sorted by fee defaulter ID in ascending order.	`SELECT * FROM FeeDefaulter ORDER BY feedef_id ASC;`
18	FeeDefaulter: Retrieve all fee defaulter records sorted by amount due in descending order.	`SELECT * FROM FeeDefaulter ORDER BY amountdue DESC;`
19	Salary: Retrieve all salary records sorted by salary ID in ascending order	`SELECT * FROM salary ORDER BY salary_id ASC;`
20	. Salary: Retrieve all salary records sorted by amount in descending order.	`SELECT * FROM salary ORDER BY amount DESC;`
21	Exams: Retrieve all exam records sorted by exam ID in ascending order.	`SELECT * FROM exams ORDER BY exam_id ASC;`
22	Exams: Retrieve all exam records sorted by exam date in descending order.	`SELECT * FROM exams ORDER BY exam_date DESC;`
23	Grades: Retrieve all grade records sorted by grade ID in ascending order.	`SELECT * FROM grades ORDER BY grade_id ASC;`
24	Grades: Retrieve all grade records sorted by student ID in descending order.	`SELECT * FROM grades ORDER BY student_id DESC;`
25	Grades: Retrieve all grade records sorted by exam ID in ascending order.	`SELECT * FROM grades ORDER BY exam_id ASC;`

9. ORDER BY using AND, OR and NOT Operators– 25 Queries

1	Students: Retrieve all student records where gender is 'Female' or sdepartment_id is 2, ordered by date_of_birth in ascending order.	`SELECT * FROM Students WHERE gender = 'Female' OR sdepartment_id = 2 ORDER BY date_of_birth ASC;`

2	Departments: Retrieve all department records where department_name is 'Science' or department_id is 3, ordered by department_id in ascending order.	`SELECT * FROM departments WHERE department_name = 'Science' OR department_id = 3 ORDER BY department_id ASC;`
3	Students: Retrieve all student records where gender is 'Male' and sdepartment_id is 1, ordered by last_name in ascending order.	`SELECT * FROM Students WHERE gender = 'Male' AND sdepartment_id = 1 ORDER BY last_name ASC;`
4	Departments: Retrieve all department records where department_id is 2 or 4, ordered by department_name in ascending order.	`SELECT * FROM departments WHERE department_id = 2 OR department_id = 4 ORDER BY department_name ASC;`
5	SDepartments: Retrieve all student department records where sdepartment_id is not 3, ordered by sdepartment_name in ascending order	`SELECT * FROM sdepartments WHERE NOT sdepartment_id = 3 ORDER BY sdepartment_name ASC;`
6	. Teachers: Retrieve all teacher records where gender is 'Female' and department_id is 2, ordered by last_name in descending order.	`SELECT * FROM Teachers WHERE gender = 'Female' AND department_id = 2 ORDER BY last_name DESC;`
7	Enrollments: Retrieve all enrollment records where enrollment_id is 100 or student_id is in the list of student_ids from the Students table where sdepartment_id is 3, ordered by student_id in ascending order.	`SELECT * FROM Enrollments WHERE enrollment_id = 100 OR student_id IN (SELECT student_id FROM Students WHERE sdepartment_id = 3) ORDER BY student_id ASC;`
8	Attendance: Retrieve all attendance records where neither student_id is 1 nor subject_id is 5, ordered by attendance_id in ascending order.	`SELECT * FROM Attendance WHERE NOT (student_id = 1 OR subject_id = 5) ORDER BY attendance_id ASC;`
9	Fees: Retrieve all fee records where payment_date is '2023-06-15' or sdepartment_id is 2, ordered by amount in descending order.	`SELECT * FROM Fees WHERE payment_date = '2023-06-15' OR sdepartment_id = 2 ORDER BY amount DESC;`

10	FeeDefaulter: Retrieve all fee defaulter records where neither student_id is 4 nor sdepartment_id is 6, ordered by feedef_id in ascending order.	`SELECT * FROM FeeDefaulter WHERE NOT (student_id = 4 OR sdepartment_id = 6) ORDER BY feedef_id ASC;`
11	Salary: Retrieve all salary records where payment_date is '2023-01-31' and teacher_id is 5, ordered by amount in descending order.	`SELECT * FROM salary WHERE payment_date = '2023-01-31' AND teacher_id = 5 ORDER BY amount DESC;`
12	Exams: Retrieve all exam records where subject_id is not in the list [10, 15], ordered by exam_date in ascending order.	`SELECT * FROM exams WHERE subject_id NOT IN (10, 15) ORDER BY exam_date ASC;`
13	Grades: Retrieve all grade records where subject_id is 5 and exam_id is 10, ordered by student_id in descending order.	`SELECT * FROM grades WHERE subject_id = 5 AND exam_id = 10 ORDER BY student_id DESC;`
14	Grades: Retrieve all grade records where neither student_id is 10 nor subject_id is 20, ordered by grade in descending order.	`SELECT * FROM grades WHERE NOT (student_id = 10 OR subject_id = 20) ORDER BY grade DESC;`
15	Subjects: Retrieve all subject records where subject_name is not 'Mathematics', ordered by credithour in descending order.	`SELECT * FROM Subjects WHERE NOT subject_name = 'Mathematics' ORDER BY credithour DESC;`
16	Teachers: Retrieve all teacher records where gender is 'Male' or department_id is 3, ordered by date_of_birth in ascending order.	`SELECT * FROM Teachers WHERE gender = 'Male' OR department_id = 3 ORDER BY date_of_birth ASC;`
17	Enrollments: Retrieve all enrollment records where neither student_id is 15 nor enrollment_id is 4, ordered by enrollment_id in descending order.	`SELECT * FROM Enrollments WHERE NOT (student_id = 15 AND Enrollment_id = 4) ORDER BY enrollment_id DESC;`
18	Attendance: Retrieve all attendance records where student_id is 7 or subject_id is 12, ordered by attendance_id in descending order.	`SELECT * FROM Attendance WHERE student_id = 7 OR subject_id = 12 ORDER BY attendance_id DESC;`

19	Fees: Retrieve all fee records where student_id is not in the list [5, 8, 10], ordered by payment_date in ascending order.	SELECT * FROM Fees WHERE student_id NOT IN (5, 8, 10) ORDER BY payment_date ASC;
20	FeeDefaulter: Retrieve all fee defaulter records where student_id is 12 and sdepartment_id is 4, ordered by amountdue in descending order.	SELECT * FROM FeeDefaulter WHERE student_id = 12 AND sdepartment_id = 4 ORDER BY amountdue DESC;
21	Salary: Retrieve all salary records where neither teacher_id is 8 nor department_id is 5, ordered by salary_id in ascending order	SELECT * FROM salary WHERE NOT (teacher_id = 8 AND department_id = 5) ORDER BY salary_id ASC;
22	. Exams: Retrieve all exam records where exam_date is '2023-07-10' or subject_id is 25, ordered by subject_id in descending order.	SELECT * FROM exams WHERE exam_date = '2023-07-10' OR subject_id = 25 ORDER BY subject_id DESC;
23	Students: Retrieve all student records where gender is 'Female' or email does not contain '@example.com', ordered by first_name in ascending order.	SELECT * FROM Students WHERE gender = 'Female' OR email NOT LIKE '%@example.com' ORDER BY first_name ASC;
24	Subjects: Retrieve not all subject records where subject_name is 'Physics' and credithour is 3, ordered by subject_id in descending order.	SELECT * FROM Subjects WHERE subject_name = 'Physics' AND credithour != 3 ORDER BY subject_id DESC;
25	Departments: Retrieve all department records where department_id is 3 or department_name does not start with 'A', ordered by department_name in ascending order.	SELECT * FROM departments WHERE department_id = 3 OR department_name NOT LIKE 'A%' ORDER BY department_name ASC;

10. GROUP BY Statement – 25 Queries

1	Students: Retrieve gender and count of students grouped by gender.	SELECT gender, COUNT(*) AS total_students FROM Students GROUP BY gender;

2	Subjects: Retrieve credithour and average credithour grouped by credithour.	`SELECT credithour, AVG(credithour) AS average_credit_hour FROM Subjects GROUP BY credithour;`
3	Departments: Retrieve department_name and count of departments grouped by department_name.	`SELECT department_name, COUNT(*) AS total_departments FROM departments GROUP BY department_name;`
4	SDepartments: Retrieve sdepartment_id and count of students grouped by sdepartment_id.	`SELECT sdepartment_id, COUNT(*) AS total_students FROM sdepartments GROUP BY sdepartment_id;`
5	Teachers: Retrieve gender and maximum date_of_birth grouped by gender.	`SELECT gender, MAX(date_of_birth) AS max_date_of_birth FROM Teachers GROUP BY gender;`
6	Enrollments: Retrieve student_id and count of enrollments grouped by student_id.	`SELECT student_id, COUNT(*) AS total_enrollments FROM Enrollments GROUP BY student_id;`
7	Attendance: Retrieve student_id, subject_id, and count of attendances grouped by student_id and subject_id.	`SELECT student_id, subject_id, COUNT(*) AS total_attendances FROM Attendance GROUP BY student_id, subject_id;`
8	Fees: Retrieve student_id and total amount of fees grouped by student_id.	`SELECT student_id, SUM(amount) AS total_amount FROM Fees GROUP BY student_id;`
9	FeeDefaulter: Retrieve student_id and average amount due grouped by student_id	`SELECT student_id, AVG(amountdue) AS average_amount_due FROM FeeDefaulter GROUP BY student_id;`
10	. Salary: Retrieve department_id and minimum amount of salary grouped by department_id.	`SELECT department_id, MIN(amount) AS min_amount FROM salary GROUP BY department_id;`
11	Exams: Retrieve subject_id and maximum exam_date grouped by subject_id..	`SELECT subject_id, MAX(exam_date) AS max_exam_date FROM exams GROUP BY subject_id;`
12	Grades: Retrieve student_id and average grade grouped by student_id.	`SELECT student_id, AVG(grade) AS average_grade FROM grades GROUP BY student_id;`
13	Students: Retrieve sdepartment_id and count of students grouped by sdepartment_id.	`SELECT sdepartment_id, COUNT(*) AS total_students FROM Students GROUP BY sdepartment_id;`

14	Subjects: Retrieve subject_name and maximum credithour grouped by subject_name.	SELECT subject_name, MAX(credithour) AS max_credit_hour FROM Subjects GROUP BY subject_name;
15	Departments: Retrieve department_id and minimum length of department_name grouped by department_id	SELECT department_id, MIN(LEN(department_name)) AS min_name_length FROM departments GROUP BY department_id;
16	Salary: Retrieve gender and average salary grouped by gender.	SELECT gender, AVG(amount) AS average_salary FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.Teacher_id GROUP BY gender;
17	Teachers: Retrieve department_id and count of teachers grouped by department_id	SELECT department_id, COUNT(*) AS total_teachers FROM Teachers GROUP BY department_id;
18	. Enrollments: Retrieve enrollment_id and maximum student_id grouped by enrollment_id.	SELECT enrollment_id, MAX(student_id) AS max_student_id FROM Enrollments GROUP BY enrollment_id;
19	Attendance: Retrieve attendance_id and minimum student_id grouped by attendance_id.	SELECT attendance_id, MIN(student_id) AS min_student_id FROM Attendance GROUP BY attendance_id;
20	Fees: Retrieve fee_id and total amount grouped by fee_id.	SELECT fee_id, SUM(amount) AS total_amount FROM Fees GROUP BY fee_id;
21	Students: Retrieve date_of_birth and count of students grouped by date_of_birth.	SELECT date_of_birth, COUNT(*) AS total_students FROM Students GROUP BY date_of_birth;
22	Grades: Retrieve grade_id and average grade grouped by grade_id.	SELECT grade_id, AVG(grade) AS average_grade FROM grades GROUP BY grade_id;
23	Exams: Retrieve exam_id and maximum exam_date grouped by exam_id.	SELECT exam_id, MAX(exam_date) AS max_exam_date FROM exams GROUP BY exam_id;
24	Salary: Retrieve salary_id and minimum amount grouped by salary_id	SELECT salary_id, MIN(amount) AS min_amount FROM salary GROUP BY salary_id;
25	. FeeDefaulter: Retrieve feedef_id and average amount due grouped by feedef_id.	SELECT feedef_id, AVG(amountdue) AS average_amount_due FROM FeeDefaulter GROUP BY feedef_id;

11. -GROUP BY using AND, OR, NOT Operators and Group by – 25 Queries

1	Students: Retrieve gender, sdepartment_id, and average age of male students in specific sdepartment_ids.	`SELECT gender, sdepartment_id, AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students WHERE gender = 'Male' AND (sdepartment_id = 1 OR sdepartment_id = 2) GROUP BY gender, sdepartment_id;`
2	Subjects: Retrieve subject_id and count of subjects with a specific subject_name and credit hour greater than 3.	`SELECT subject_id, COUNT(*) AS total_subjects FROM Subjects WHERE subject_name = 'Engineering' AND credithour > 3 GROUP BY subject_id;`
3	Teachers: Retrieve gender, department_id, and count of female teachers in specific department_ids.	`SELECT gender, department_id, COUNT(*) AS total_teachers FROM Teachers WHERE gender = 'Female' AND (department_id = 1 OR department_id = 3) GROUP BY gender, department_id;` `SELECT student_id, subject_id, max(grade) AS max_grade FROM grades WHERE student_id IN ( SELECT student_id FROM Students WHERE gender = 'Female' AND (sdepartment_id = 1 OR sdepartment_id = 2) ) AND grade > 80 GROUP BY student_id, subject_id;`
4	Subjects: Retrieve sdepartment_id and average credit hour grouped by sdepartment_id.	`SELECT sdepartment_id, AVG(credithour) AS average_credit_hour FROM Subjects GROUP BY sdepartment_id;`
5	Teachers: Retrieve gender, department_id, and count of female teachers in a specific department_id	`SELECT gender, department_id, COUNT(*) AS count FROM Teachers WHERE gender = 'Female' AND department_id = 1 GROUP BY gender, department_id;`
6	. Students: Retrieve sdepartment_id and count of male students in specific sdepartment_ids or with a specific first_name.	`SELECT sdepartment_id, COUNT(*) AS count FROM Students WHERE gender = 'Male' AND (sdepartment_id = 1 OR first_name = 'ali') GROUP BY sdepartment_id;`
7	Teachers: Retrieve gender, department_id, and count of female teachers in a specific department_id and with a date_of_birth after a certain date.	`SELECT gender, department_id, COUNT(*) AS count FROM Teachers WHERE gender = 'Female' AND department_id = 'Science' AND date_of_birth >= '1980-01-01' GROUP BY gender, department_id;`

8	Subjects: Retrieve sdepartment_id and count of subjects with credit hours between 2 and 4, grouped by sdepartment_id	`SELECT sdepartment_id, COUNT(*) AS count FROM Subjects WHERE credithour BETWEEN 2 AND 4 GROUP BY sdepartment_id;`
9	. Teachers: Retrieve gender, department_id, and count of male teachers in a specific department_id and with a date_of_birth after a certain date.	`SELECT gender, department_id, COUNT(*) AS count FROM Teachers WHERE gender = 'Male' AND department_id = (123) AND date_of_birth > '1985-01-01' GROUP BY gender, department_id;`
10	Subjects: Retrieve sdepartment_id and total credit hour of subjects excluding specific sdepartment_ids.	`SELECT sdepartment_id, SUM(credithour) AS total_credit_hour FROM Subjects WHERE sdepartment_id NOT IN (1, 2) GROUP BY sdepartment_id;`
11	Fees: Retrieve student_id and total fees paid by students with amounts above 500 or payment dates before a specific date.	`SELECT student_id, SUM(amount) AS total_fees_paid FROM Fees WHERE amount > 500 OR payment_date < '2022-01-01' GROUP BY student_id;`
12	FeeDefaulter: Retrieve student_id and total amount due for students with amounts due above 100 and in a specific sdepartment_id.	`SELECT student_id, SUM(amountdue) AS total_amount_due FROM Feedefaulter WHERE amountdue > 100 AND sdepartment_id = 1 GROUP BY student_id;`
13	Grades: Retrieve student_id, subject_id, and count of grades for students not in a specific sdepartment_id.	`SELECT student_id, subject_id, count(grade) AS grade FROM grades WHERE student_id NOT IN (SELECT student_id FROM Students WHERE sdepartment_id = 1) GROUP BY student_id, subject_id;`
14	Students: Retrieve sdepartment_id, gender, and count of students with date_of_birth before a specific date or last_name starting with 'S'.	`SELECT sdepartment_id, gender, COUNT(*) AS student_count FROM Students WHERE date_of_birth < '2000-01-01' OR last_name LIKE 'S%' GROUP BY sdepartment_id, gender;`
15	Students: Retrieve student_id and count of enrollments for students in a specific sdepartment_id with last_name starting with 'A' or contact_number containing '123'.	`SELECT student_id, COUNT(*) AS enrollment_count FROM Students WHERE sdepartment_id = 1 AND (last_name LIKE 'A%' OR contact_number LIKE '%123') GROUP BY student_id;`
16	Attendance: Retrieve subject_id and count of	`SELECT subject_id, COUNT(*) AS total_attendance FROM Attendance WHERE subject_id = 1 OR subject_id = 4 GROUP BY subject_id;`

	attendances for specific subject_ids.	
17	Fees: Retrieve student_id and total fees paid by students with amounts above 500 and payment dates within a specific range.	`SELECT student_id, SUM(amount) AS total_fees_paid FROM Fees WHERE amount > 500 AND payment_date BETWEEN '2022-01-01' AND '2022-12-31' GROUP BY student_id;`
18	FeeDefaulter: Retrieve student_id and total amount due for students with amounts due above 100 or in a specific sdepartment_id.	`SELECT student_id, SUM(amountdue) AS total_amount_due FROM Feedefaulter WHERE amountdue > 100 OR sdepartment_id = 1 GROUP BY student_id;`
19	Exams: Retrieve subject_id and minimum exam_date for subject_ids outside a specific range.	`SELECT subject_id, MIN(exam_date) AS min_exam_date FROM exams WHERE subject_id < 5 OR subject_id > 10 GROUP BY subject_id;`
20	Students: Retrieve sdepartment_id, gender, and average age of male students excluding a specific sdepartment_id	`SELECT sdepartment_id, gender, AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students WHERE gender = 'Male' AND sdepartment_id <> 1 GROUP BY sdepartment_id, gender;`
21	. Enrollments: Retrieve student_id and count of enrollments for students in a specific sdepartment_id with enrollment_id above 100..	`SELECT student_id, COUNT(*) AS enrollment_count FROM Enrollments WHERE sdepartment_id = 1 AND enrollment_id > 100 GROUP BY student_id;`
22	Attendance: Retrieve subject_id and count of attendances for subject_ids not in a specific range	`SELECT subject_id, COUNT(*) AS attendance_count FROM Attendance WHERE subject_id NOT IN (2, 4) GROUP BY subject_id;`
23	Fees: Retrieve student_id and total fees paid by students with amounts above 100 in specific sdepartment_ids.	`SELECT student_id, SUM(amount) AS total_fees_paid FROM Fees WHERE amount > 100 AND (sdepartment_id = 1 OR sdepartment_id = 1) GROUP BY student_id;`
24	Salary: Retrieve department_id and maximum salary amount for department_ids excluding specific values	`SELECT department_id, MAX(amount) AS max_salary_amount FROM salary WHERE department_id NOT IN (1, 3) GROUP BY department_id;`
25	. Grades: Retrieve student_id, subject_id, and minimum grade for students in a specific sdepartment_id with grades above 80.	`SELECT student_id, subject_id, MIN(grade) AS min_grade FROM grades WHERE student_id IN (SELECT student_id FROM Students WHERE sdepartment_id = 1) AND grade > 80 GROUP BY student_id, subject_id;`

12. Subqueries– 30 Queries

1	Subjects: Retrieve subject_name based on subject_id associated with a specific teacher_id.	SELECT subject_name FROM Subjects WHERE subject_id IN (SELECT subject_id FROM Teachers WHERE Teacher_id = teacher_id);
2	Subjects: Retrieve subject_name based on subject_id associated with a specific department_id.	SELECT subject_name FROM Subjects WHERE subject_id IN (SELECT subject_id FROM departments WHERE department_id = department_id);
3	Teachers: Retrieve full_name and email of a teacher based on subject_id.	SELECT CONCAT(first_name, ' ', last_name) AS full_name,email FROM Teachers WHERE Teacher_id = (SELECT Teacher_id FROM Subjects WHERE subject_id = subject_id);
4	Enrollments: Retrieve the count of enrollments for a specific student in their respective sdepartment_id.	SELECT COUNT(*) AS enrollment_count FROM Enrollments WHERE student_id IN (SELECT student_id FROM Students WHERE sdepartment_id = sdepartment_id);
5	Fees: Retrieve the total fee amount for a specific student based on their sdepartment_id.	SELECT SUM(amount) AS total_fee_amount FROM Fees WHERE student_id = (SELECT student_id FROM Students WHERE sdepartment_id = sdepartment_id);
6	Departments: Retrieve the department_name associated with a specific department_id.	SELECT department_name FROM departments WHERE department_id = (SELECT department_id FROM Teachers WHERE Teacher_id = teacher_id);
7	Students: Retrieve the count of male students enrolled in a subject based on subject_id.	SELECT COUNT(*) AS male_student_count FROM Students WHERE gender = 'Male' AND student_id IN (SELECT student_id FROM Enrollments

		WHERE subject_id = subject_id);
8	Subjects: Retrieve subject_name based on subject_id associated with male teachers	SELECT subject_name FROM Subjects WHERE subject_id IN (SELECT subject_id FROM Teachers WHERE gender = 'Male');
9	. Fees: Retrieve the total fee amount paid by male students. sdepartment_id in the Fees table.	SELECT SUM(amount) AS total_fee_amount FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Male');
10	Subjects: Retrieve subject_name based on subject_id associated with female teachers.	SELECT subject_name FROM Subjects WHERE subject_id IN (SELECT subject_id FROM Teachers WHERE gender = 'Female');
11	Fees: Retrieve the count of students associated with a specific	SELECT COUNT(*) AS student_count FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE sdepartment_id = sdepartment_id);
12	Departments: Retrieve the department_name based on department_id associated with a teacher_id.	SELECT department_name FROM departments WHERE department_id IN (SELECT department_id FROM Teachers WHERE Teacher_id = teacher_id);
13	Teachers: Retrieve the full_name of a teacher based on department_id associated with department_id.	SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM Teachers WHERE Teacher_id = (SELECT Teacher_id FROM departments WHERE department_id = department_id);
14	Fees: Retrieve the total fee amount paid by female students.	SELECT SUM(amount) AS total_fee_amount FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Female');
15	Students: Retrieve the average age of female students in the Science department.	SELECT AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students WHERE gender = 'Female' AND sdepartment_id = (SELECT department_id FROM departments WHERE department_name = 'Science');

16	Students: Retrieve the count of unpaid students in a specific sdepartment_id.	SELECT COUNT(*) AS unpaid_count FROM Students WHERE student_id NOT IN (SELECT student_id FROM Fees WHERE sdepartment_id = sdepartment_id);
17	Students: Retrieve student_id of students who have not defaulted on fees	SELECT student_id FROM Students WHERE student_id NOT IN (SELECT student_id FROM FeeDefaulter);
18	. Subjects: Retrieve subject_name for subjects with a credit hour greater than 3 or associated with female teachers.	SELECT subject_name FROM Subjects WHERE credithour > 3 OR subject_id IN (SELECT subject_id FROM Teachers WHERE gender = 'Female');
19	Subjects: Retrieve subject_name for subjects associated with male teachers or with a credit hour less than 4.	SELECT subject_name FROM Subjects WHERE subject_id IN (SELECT subject_id FROM Teachers WHERE gender = 'Male') OR credithour < 4;
20	Students: Retrieve email of students in the Arts department who have not paid fees	SELECT email FROM Students WHERE sdepartment_id = (SELECT department_id FROM departments WHERE department_name = 'Arts') AND student_id NOT IN (SELECT student_id FROM Fees);
21	. Departments: Retrieve department_name for department_ids associated with teachers	SELECT department_name FROM departments WHERE department_id IN (SELECT department_id FROM Teachers GROUP BY department_id);
22	. Students: Retrieve student_id of students who have not attended any classes and are not defaulters.	SELECT student_id FROM Students WHERE student_id NOT IN (SELECT student_id FROM Attendance) AND student_id NOT IN (SELECT student_id FROM Feedefaulter);
23	Students: Retrieve student_id of students who are enrolled in both the Science and Arts departments.	SELECT student_id FROM Students WHERE student_id IN (SELECT student_id FROM Enrollments WHERE sdepartment_id = (SELECT department_id FROM departments WHERE department_name = 'Science')) AND student_id IN (SELECT student_id FROM Enrollments WHERE sdepartment_id = (SELECT department_id FROM departments WHERE department_name = 'Arts'));
24	Teachers: Retrieve Teacher_id of teachers who have not received a salary.	SELECT Teacher_id FROM Teachers WHERE Teacher_id NOT IN (SELECT teacher_id FROM salary);
25	Departments: Retrieve department_name for department_ids with more than 100 enrollments.	SELECT department_name FROM departments WHERE department_id IN (SELECT sdepartment_id FROM Enrollments GROUP BY sdepartment_id HAVING COUNT(student_id) > 100);

26	Students: Retrieve student_id of students who have not enrolled in any courses.	SELECT student_id FROM Students WHERE student_id NOT IN (SELECT student_id FROM Enrollments);
27	Students: Retrieve student_id of students who are defaulters but have not attended any classes	SELECT student_id FROM Students WHERE student_id IN (SELECT student_id FROM Feedefaulter) AND student_id NOT IN (SELECT student_id FROM Attendance);
28	Departments: Retrieve department_name for department_ids associated with male teachers and have more male teachers than female teachers.	SELECT department_name FROM departments WHERE department_id IN (SELECT department_id FROM Teachers WHERE gender = 'Male' GROUP BY department_id HAVING COUNT(Teacher_id) > (SELECT COUNT(Teacher_id) FROM Teachers WHERE gender = 'Female' GROUP BY department_id));
29	Subjects: Retrieve subject_id and subject_name for subjects with a credit hour greater than 3 or associated with teachers with the last name 'Smith'.	SELECT subject_id, subject_name FROM Subjects WHERE credithour > 3 OR subject_id IN (SELECT subject_id FROM Teachers WHERE last_name = 'Smith');
30	. Subjects: Retrieve subject_id and subject_name for subjects that have not been enrolled in and have a credit hour greater than 3.	SELECT subject_id, subject_name FROM Subjects WHERE subject_id NOT IN (SELECT subject_id FROM Enrollments) AND credithour > 3;

13. Subqueries with logical operators– 30 Queries

1	Retrieve students who are enrolled in a specific subject:	SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Enrollments WHERE subject_id = 1);
2	Retrieve students who are enrolled in more than one subject:	SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Enrollments GROUP BY student_id HAVING COUNT(*) > 1);
3	Retrieve students who have not enrolled in any subject:	SELECT * FROM Students WHERE student_id NOT IN (SELECT student_id FROM Enrollments);
4	retrieves all the subjects that have a credit hour value less than or equal to 3	SELECT * FROM Subjects WHERE subject_id not IN (SELECT subject_id FROM Subjects WHERE credithour > 3);

5	Retrieve subjects offered in a specific department:	`SELECT * FROM Subjects WHERE subject_id IN (SELECT subject_id FROM Enrollments WHERE sdepartment_id = 3);`
6	Retrieve departments with more than five subjects:	`SELECT * FROM Departments WHERE department_id IN (SELECT department_id FROM Subjects GROUP BY department_id HAVING COUNT(*) > 5);`
7	query will return all students who belong to the department with department_id '1'.	`SELECT * FROM Students WHERE sdepartment_id IN ( SELECT sdepartment_id FROM Departments WHERE department_id = '1'`
8	query will return all teachers who belong to the department with department_name '2'.	`SELECT * FROM Teachers WHERE department_id = ( SELECT department_id FROM Departments WHERE department_name = '2' );`
9	Retrieve students who are enrolled in a subject and belong to a specific department:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Enrollments WHERE subject_id = 1 AND sdepartment_id = 1);`
10	Retrieve students who have not attended a specific exam:	`SELECT * FROM Students WHERE student_id NOT IN (SELECT student_id FROM Attendance);`
11	Retrieve students who have not attended any exams:	`SELECT * FROM Students WHERE student_id NOT IN (SELECT student_id FROM Attendance);`
12	Retrieve students who have not defaulted on fees:	`SELECT * FROM Students WHERE student_id NOT IN (SELECT student_id FROM Fees WHERE payment_date = '2022-11-20');`
13	Retrieve students who have not defaulted on fees:	`SELECT * FROM Students WHERE student_id NOT IN (SELECT student_id FROM FeeDefaulter);`
14	Retrieve students who have defaulted on fees in a specific department:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM FeeDefaulter WHERE sdepartment_id = 44);`
15	Retrieve teachers who have a salary greater than a certain amount:	`SELECT * FROM Teachers WHERE teacher_id IN (SELECT teacher_id FROM Salary WHERE amount > 100);`
16	Retrieve teachers who have received a salary on a specific date:	`SELECT * FROM Teachers WHERE teacher_id IN (SELECT teacher_id FROM Salary WHERE payment_date ='2022-11-20');`

17	The subquery that retrieves the subject with subject_id 3 from the Subjects table.	`SELECT * FROM Exams WHERE subject_id NOT IN ( SELECT subject_id FROM Subjects WHERE subject_id = 3 );`
18	y will return all exams with an exam_date greater than the maximum exam date for the subject with subject_id = 3.	`SELECT * FROM Exams WHERE exam_date > ( SELECT MAX(exam_date) FROM Exams WHERE subject_id = 3 );`
19	Filter Grades by Subject ID.	`SELECT * FROM Grades WHERE subject_id IN ( SELECT subject_id FROM Subjects WHERE subject_name = 'YourSubjectName' );`
20	Filter grades by subject ID 3.	`SELECT * FROM Grades WHERE subject_id IN ( SELECT subject_id FROM Subjects WHERE subject_id = 3 );`
21	Retrieve students who have obtained a grade higher than a certain value:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE grade > 'A');`
22	Retrieve students who have obtained a grade lower than a certain value:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE grade < 'b');`
23	Retrieve students who have obtained a grade within a specific range:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE grade BETWEEN 'lower_range' AND 'upper_range');`
24	Retrieve students who have obtained a grade in a specific subject and exam:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id = 1 AND exam_id = 5);`
25	Retrieve students who have obtained a grade in a specific subject or exam:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id =1 OR exam_id = 2);`
26	Retrieve students who have obtained a grade in a specific subject and a grade higher than a certain value:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id = 1 AND grade > 4);`
27	Retrieve students who have obtained a grade in a specific subject or a grade lower than a certain value:	`SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id = 5 OR grade < 'D');`

28	Retrieve students who have obtained a grade in a specific subject and belong to a specific department:	SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id = 6) AND sdepartment_id =4;
29	Retrieve students who have obtained a grade in a specific subject or belong to a specific department:	SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id = 3) OR sdepartment_id = 12;
30	Retrieve students who have obtained a grade in a specific subject and exam or belong to a specific department:	SELECT * FROM Students WHERE student_id IN (SELECT student_id FROM Grades WHERE subject_id = 1 AND exam_id = 1) OR sdepartment_id = 2;

14. Aggregate functions MAX, MIN, SUM, COUNT, and AVG. – 20 Queries

1	Subjects: Retrieve the maximum credit hour among all subjects	SELECT MAX(credithour) AS max_credit_hour FROM Subjects;
2	. FeeDefaulter: Retrieve the minimum amount due among all defaulters.	SELECT MIN(amountdue) AS min_amount_due FROM FeeDefaulter;
3	Fees: Retrieve the total amount paid in fees.	SELECT SUM(amount) AS total_amount_paid FROM Fees;
4	Students: Retrieve the total number of students in each sdepartment_id.	SELECT sdepartment_id, COUNT(*) AS total_students FROM Students GROUP BY sdepartment_id;
5	Salary: Retrieve the average salary.	SELECT AVG(amount) AS average_salary FROM Salary;
6	Grades: Retrieve the maximum grade for each student.	SELECT student_id, MAX(grade) AS max_grade FROM Grades GROUP BY student_id;
7	Enrollments: Retrieve the total number of enrollments for each subject..	SELECT subject_id, COUNT(*) AS total_enrollments FROM Enrollments GROUP BY subject_id;);
8	Students: Retrieve the minimum date of birth among all students	SELECT MIN(date_of_birth) AS min_date_of_birth FROM Students;
9	. Fees: Retrieve the total fees paid by male students.	SELECT SUM(amount) AS total_fees_paid FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Male');

10	Subjects: Retrieve the count of distinct subject names	SELECT COUNT(DISTINCT subject_name) AS distinct_subjects FROM Subjects;
11	Students: Retrieve the average age of male students.	SELECT AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students WHERE gender = 'Male';
12	FeeDefaulter: Retrieve the total amount due for each sdepartment_id.	SELECT sdepartment_id, SUM(amountdue) AS total_amount_due FROM FeeDefaulter GROUP BY sdepartment_id;
13	Salary: Retrieve the minimum salary for teachers in the Mathematics department.	SELECT MIN(amount) AS min_salary FROM Salary WHERE teacher_id IN (SELECT teacher_id FROM Teachers WHERE department_id = (SELECT department_id FROM departments WHERE department_name = 'Mathematics')
14	Subjects: Retrieve the average credit hour for subjects in the Science department.	SELECT AVG(credithour) AS average_credit_hour FROM Subjects WHERE subject_id IN (SELECT subject_id FROM Subjects WHERE sdepartment_id = (SELECT department_id FROM departments WHERE department_name = 'Science'));
15	Enrollments: Retrieve the count of students enrolled in subjects with a credit hour greater than 4.	SELECT COUNT(*) AS total_students_enrolled FROM Enrollments WHERE subject_id IN (SELECT subject_id FROM Subjects WHERE credithour > 4);
16	Fees: Retrieve the maximum amount paid by each student.	SELECT student_id, MAX(amount) AS max_amount_paid FROM Fees GROUP BY student_id;
17	Grades: Retrieve the total number of grades	SELECT count(grade) AS total_grades FROM Grades;
18	. Students: Retrieve the total number of students for each sdepartment_id and gender	SELECT sdepartment_id, gender, COUNT(*) AS total_students FROM Students GROUP BY sdepartment_id, gender;
19	. FeeDefaulter: Retrieve the maximum amount due for each sdepartment_id.	SELECT sdepartment_id, MAX(amountdue) AS max_amount_due FROM FeeDefaulter GROUP BY sdepartment_id;
20	Feedefaulter: Retrieve the maximum amount due among all defaulters.	SELECT MAX(amountdue) AS maximum_amount_due FROM Feedefaulter;

-15. Aggregate functions using logical Operators and Group by – 30 Queries

1	Students: Retrieve the average age of male students for each sdepartment_id.	SELECT sdepartment_id, AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students WHERE gender = 'Male' GROUP BY sdepartment_id;
2	Fees: Retrieve the maximum fee amount for each sdepartment_id.	SELECT sdepartment_id, MAX(amount) AS max_fee_amount FROM Fees GROUP BY sdepartment_id;
3	Grades: Retrieve the minimum grade achieved for each subject.	SELECT subject_id, MIN(grade) AS min_grade FROM grades GROUP BY subject_id;
4	Salary: Retrieve the average salary for each department_id.	SELECT department_id, AVG(amount) AS average_salary FROM salary GROUP BY department_id;
5	Students: Retrieve the count of students for each sdepartment_id and gender.	SELECT sdepartment_id, gender, COUNT(student_id) AS student_count FROM Students GROUP BY sdepartment_id, gender;
6	Subjects: Retrieve the maximum credit hours among all subjects.	SELECT MAX(credithour) AS max_credit_hours FROM Subjects;
7	Fees: Retrieve the total fees paid by male students for each sdepartment_id	SELECT sdepartment_id, SUM(amount) AS total_fees_paid FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Male') GROUP BY sdepartment_id;
8	. Fees: Retrieve the total fees paid for each sdepartment_id.	SELECT sdepartment_id, SUM(amount) AS total_fees_paid FROM Fees GROUP BY sdepartment_id;
9	Grades: Retrieve the maximum grade achieved for each subject.	SELECT subject_id, MAX(grade) AS max_grade FROM grades GROUP BY subject_id;
10	Salary: Retrieve the average salary for each department_id.	SELECT department_id ,AVG(amount) AS average_salary FROM salary GROUP BY department_id ;
11	Enrollments: Retrieve the count of male students enrolled in each subject.	SELECT subject_id, COUNT(student_id) AS male_student_count FROM Enrollments WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Male') GROUP BY subject_id;
12	Fees: Retrieve the total fees paid by female students for each sdepartment_id.	SELECT sdepartment_id, SUM(amount) AS total_fees_paid FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Female')

		GROUP BY sdepartment_id;
13	Enrollments: Retrieve the total number of students enrolled in each subject.	SELECT subject_id, COUNT(student_id) AS total_students FROM Enrollments GROUP BY subject_id;
14	Fees: Retrieve the total fees paid for each sdepartment_id..	SELECT sdepartment_id, SUM(amount) AS total_fees_paid FROM Fees GROUP BY sdepartment_id;
15	Grades: Retrieve the maximum grade achieved by female students for each subject	SELECT subject_id, MAX(grade) AS max_grade FROM grades WHERE student_id IN (SELECT student_id FROM Students WHERE gender = 'Female') GROUP BY subject_id;
16	Students: Retrieve the average age of students for each sdepartment_id.	SELECT sdepartment_id, AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students GROUP BY sdepartment_id;
17	Subjects: Retrieve the count of subjects for each sdepartment_id.	SELECT sdepartment_id, COUNT(subject_id) AS subject_count FROM Subjects GROUP BY sdepartment_id;
18	Fees: Retrieve the maximum fee amount paid by each student.	SELECT student_id, MAX(amount) AS max_fee_amount FROM Fees GROUP BY student_id;
19	Subjects: Retrieve the average credit hour for each sdepartment_id	SELECT sdepartment_id, AVG(credithour) AS average_credit_hour FROM Subjects GROUP BY sdepartment_id;
20	. Grades: Retrieve the count of students who scored above 80 in each subject.	SELECT subject_id, COUNT(student_id) AS student_count FROM grades WHERE grade > 80 GROUP BY subject_id;
21	Salary: Retrieve the minimum salary for each department_id	SELECT department_id, MIN(amount) AS min_salary FROM salary GROUP BY department_id;
22	. Fees: Retrieve the total fees paid for each sdepartment_id.	SELECT sdepartment_id, SUM(amount) AS total_fees_paid FROM Fees GROUP BY sdepartment_id;
23	Students: Retrieve the average age of students in the sdepartment_id 1 who are also enrolled	SELECT AVG(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS average_age FROM Students WHERE sdepartment_id = 1 AND student_id IN (SELECT student_id FROM Enrollments);

24	. Fees: Retrieve the total fees paid by students with email addresses ending in 'gmail.com'	SELECT SUM(amount) AS total_fees_paid FROM Fees WHERE student_id IN (SELECT student_id FROM Students WHERE email LIKE '%gmail.com');
25	. Students: Count the total number of male students who are enrolled in subjects with names starting with 'English'.	SELECT COUNT(*) AS total_male_students FROM Students WHERE gender = 'Male' AND student_id IN (SELECT student_id FROM Enrollments WHERE subject_id IN (SELECT subject_id FROM Subjects WHERE subject_name LIKE 'English%'));
26	Salary: Retrieve the average salary for teachers in the 'Arts' department.	SELECT AVG(amount) AS average_salary FROM salary WHERE teacher_id IN (SELECT Teacher_id FROM Teachers WHERE department_id = (SELECT department_id FROM departments WHERE department_name = 'Arts'))
27	Subjects: Retrieve the maximum credit hour among subjects taught by female teachers.	SELECT MAX(credithour) AS max_credit_hour FROM Subjects WHERE subject_id IN (SELECT Teacher_id FROM Teachers WHERE gender = 'Female');
28	Enrollments: Count the number of students enrolled in each sdepartment_id.	SELECT sdepartment_id, COUNT(student_id) AS enrolled_students FROM Enrollments GROUP BY sdepartment_id;
29	Students: Retrieve the minimum age among male students in sdepartment_id 1.	SELECT MIN(DATEDIFF(YEAR, date_of_birth, GETDATE())) AS min_age FROM Students WHERE gender = 'Male' AND sdepartment_id = 1;
30	Fees: Retrieve the total fees paid for each sdepartment_id.	SELECT sdepartment_id, SUM(amount) AS total_fees_paid FROM Fees GROUP BY sdepartment_id;

16. Inner Joins – 20 Queries

1	Retrieve Students with Department Name	`SELECT Students.*, sdepartments.sdepartment_name FROM Students INNER JOIN sdepartments ON Students.sdepartment_id = sdepartments.sdepartment_id;`
2	Retrieve Subjects with Department Name	`SELECT Subjects.*, sdepartments.sdepartment_name FROM Subjects INNER JOIN sdepartments ON Subjects.sdepartment_id = sdepartments.sdepartment_id;`
3	Retrieve Teachers with Department Name	`SELECT Teachers.*, departments.department_name FROM Teachers INNER JOIN departments ON Teachers.department_id = departments.department_id;`
4	Retrieve Enrollments with Student Name and Subject Name	`SELECT Enrollments.*, Students.first_name, Students.last_name, Subjects.subject_name FROM Enrollments INNER JOIN Students ON Enrollments.student_id = Students.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id`
5	Retrieve Attendance with Student Name and Subject Name	`SELECT attendance.*, Students.first_name, Students.last_name, Subjects.subject_name FROM attendance INNER JOIN Students ON attendance.student_id = Students.student_id INNER JOIN Subjects ON attendance.subject_id = Subjects.subject_id;`
6	Retrieve Fees with Student Name and Department Name	`SELECT fees.*, Students.first_name, Students.last_name, departments.department_name FROM fees INNER JOIN Students ON fees.student_id = Students.student_id INNER JOIN departments ON fees.sdepartment_id = departments.department_id;`
7	Retrieve Fee Defaulters with Student Name and Department Name	`SELECT Feedefaulter.*, Students.first_name, Students.last_name, departments.department_name FROM Feedefaulter INNER JOIN Students ON Feedefaulter.student_id = Students.student_id INNER JOIN departments ON Feedefaulter.sdepartment_id = departments.department_id;`
8	Retrieve Grades with Student Name and Subject Name	`SELECT grades.*, Students.first_name, Students.last_name, Subjects.subject_name FROM grades`

		INNER JOIN Students ON grades.student_id = Students.student_id INNER JOIN Subjects ON grades.subject_id = Subjects.subject_id;
9	Retrieve Exams with Subject Name	SELECT exams.*, Subjects.subject_name FROM exams INNER JOIN Subjects ON exams.subject_id = Subjects.subject_id;
10	Retrieve Salary with Teacher Name and Department Name	SELECT salary.*, Teachers.first_name, Teachers.last_name, departments.department_name FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.teacher_id INNER JOIN departments ON salary.department_id = departments.department_id;
11	Retrieve Student Information with Enrollments	SELECT Students.*, Enrollments.enrollment_id, Enrollments.subject_id FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id;
12	Retrieve Subject Information with Department Names	SELECT Subjects.*, sdepartments.sdepartment_name FROM Subjects INNER JOIN sdepartments ON Subjects.sdepartment_id = sdepartments.sdepartment_id;
13	Retrieve Enrollment Information with Student and Subject Details	SELECT Enrollments.*, Students.first_name, Students.last_name, Subjects.subject_name FROM Enrollments INNER JOIN Students ON Enrollments.student_id = Students.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id;
14	Retrieve Teacher Information with Department Name and Email	SELECT Teachers.*, departments.department_name, Teachers.email FROM Teachers INNER JOIN departments ON Teachers.department_id = departments.department_id;
15	Retrieve Attendance Information with Student Name, Subject Name, and Date	SELECT attendance.*, Students.first_name, Students.last_name, Subjects.subject_name, attendance.date FROM attendance INNER JOIN Students ON attendance.student_id = Students.student_id INNER JOIN Subjects ON attendance.subject_id = Subjects.subject_id;
16	Retrieve Fees Information with Student Name,	SELECT fees.*, Students.first_name, Students.last_name, departments.department_name, fees.payment_date

	Department Name, and Payment Date	FROM fees INNER JOIN Students ON fees.student_id = Students.student_id INNER JOIN departments ON fees.sdepartment_id = departments.department_id;
17	Retrieve Fee Defaulters with Student Name, Department Name, and Amount Due	SELECT Feedefaulter.*, Students.first_name, Students.last_name, departments.department_name, Feedefaulter.amountdue FROM Feedefaulter INNER JOIN Students ON Feedefaulter.student_id = Students.student_id INNER JOIN departments ON Feedefaulter.sdepartment_id = departments.department_id;
18	Retrieve Exam Grades with Student Name, Subject Name, and Exam ID	SELECT grades.*, Students.first_name, Students.last_name, Subjects.subject_name, grades.exam_id FROM grades INNER JOIN Students ON grades.student_id = Students.student_id INNER JOIN Subjects ON grades.subject_id = Subjects.subject_id;
19	Retrieve Exam Details with Subject Name and Exam Date	SELECT exams.*, Subjects.subject_name, exams.exam_date FROM exams INNER JOIN Subjects ON exams.subject_id = Subjects.subject_id;
20	Retrieve Salary Details with Teacher Name, Department Name, and Payment Date	SELECT salary.*, Teachers.first_name, Teachers.last_name, departments.department_name, salary.payment_date FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.teacher_id INNER JOIN departments ON salary.department_id = departments.department_id;

17. Inner Joins using logical Operators, Group by and Order by– 30 Queries

1	Retrieve Students with Department Name where the Date of Birth is '2002-10-05' or the Department Name is 'CS', ordered by the last name.	SELECT Students.*, sdepartments.sdepartment_name FROM Students INNER JOIN sdepartments ON Students.sdepartment_id = sdepartments.sdepartment_id where Students.date_of_birth='2002-10-05' or sdepartments.sdepartment_name='CS' ORDER BY Students.last_name;

2	Retrieve Subject details with Department Name where the Credit Hours are greater than 2 and less than 5, grouped by Subject ID, Subject Name, Credit Hours, and Department Name.	SELECT Subjects.subject_id, Subjects.subject_name, Subjects.credithour, sdepartments.sdepartment_name FROM Subjects INNER JOIN sdepartments ON Subjects.sdepartment_id = sdepartments.sdepartment_id where Subjects.credithour>2 and Subjects.credithour<5 GROUP BY Subjects.subject_id, Subjects.subject_name, Subjects.credithour, sdepartments.sdepartment_name;
3	Retrieve Teacher details with Department Name where the gender is 'male' or the Department Name is 'SE', ordered by the Teacher's first name.	SELECT Teachers.*, departments.department_name FROM Teachers INNER JOIN departments ON Teachers.department_id = departments.department_id where Teachers.gender='male'or departments.department_name='SE' ORDER BY Teachers.first_name;
4	Retrieve Enrollment details with Student's first name, last name, and Subject's subject name where the Student ID is 1 or the Subject Name is 'Math', ordered by the Enrollment ID.	SELECT Enrollments.*, Students.first_name, Students.last_name, Subjects.subject_name FROM Enrollments INNER JOIN Students ON Enrollments.student_id = Students.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id where Students.student_id=1 or Subjects.subject_name='Math' ORDER BY Enrollments.enrollment_id;
5	Function: Retrieve attendance details for a specific date, including attendance ID, date, student's first name, student's last name, and subject name, grouped by attendance ID, date, student's first name, student's last name, and subject name.	SELECT Attendance.attendance_id,Attendance.date, Students.first_name, Students.last_name, Subjects.subject_name FROM attendance INNER JOIN Students ON attendance.student_id = Students.student_id INNER JOIN Subjects ON attendance.subject_id = Subjects.subject_id where Attendance.date='2023-10-05' GROUP BY Attendance.attendance_id,Attendance.date, Students.first_name, Students.last_name, Subjects.subject_name
6	Function: Retrieve fee details for male students with an amount of 12000, including all fee information, student's first name, student's last name, and the name of the	SELECT fees.*, Students.first_name, Students.last_name, sdepartments.sdepartment_name FROM fees INNER JOIN Students ON fees.student_id = Students.student_id INNER JOIN sdepartments ON fees.sdepartment_id = sdepartments.sdepartment_id where Students.gender='male' and Fees.amount=12000 ORDER BY fees.payment_date;

	student department, ordered by payment date.	
7	Function: Retrieve details of fee defaulters with an amount due between 12000 and 20000, including fee defaulter ID, amount due, student's first name, student's last name, and the name of the department they belong to, grouped by fee defaulter ID, amount due, student's first name, student's last name, and department name.	SELECT Feedefaulter.feedef_id,Feedefaulter.amountdue, Students.first_name, Students.last_name, departments.department_name FROM Feedefaulter INNER JOIN Students ON Feedefaulter.student_id = Students.student_id INNER JOIN departments ON Feedefaulter.sdepartment_id = departments.department_id where Feedefaulter.amountdue between 12000 and 20000 GROUP BY Feedefaulter.feedef_id,Feedefaulter.amountdue, Students.first_name, Students.last_name, departments.department_name
8	Function: Retrieve exam details for exams taking place on June 7, 2023, or exams related to the subject with the name "PF", including all exam information and the name of the subject, ordered by exam date.	SELECT exams.*, Subjects.subject_name FROM exams INNER JOIN Subjects ON exams.subject_id = Subjects.subject_id where exams.exam_date='2023-06-07' or Subjects.subject_name='PF' ORDER BY exams.exam_date;
9	Function: Retrieve salary details for female teachers or teachers belonging to the English department, including salary ID, payment date, teacher's first name, teacher's last name, and the name of the department they belong to, grouped by salary ID, payment date, teacher's first name, teacher's last name, and department name.	SELECT salary.salary_id,salary.payment_date, Teachers.first_name, Teachers.last_name, departments.department_name FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.teacher_id INNER JOIN departments ON salary.department_id = departments.department_id where Teachers.gender='female' or departments.department_name='English' GROUP BY salary.salary_id,salary.payment_date, Teachers.first_name, Teachers.last_name, departments.department_name;
10	Retrieve student details for female students who have an enrollment ID of 1,	SELECT Students.first_name,Students.student_id, Enrollments.enrollment_id, Enrollments.subject_id FROM Students

	including student's first name, student ID, enrollment ID, and subject ID, grouped by student's first name, student ID, enrollment ID, and subject ID, ordered by student ID.	INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id where Enrollments.enrollment_id=1 and Students.gender='FEMALE' group by Students.first_name,Students.student_id, Enrollments.enrollment_id, Enrollments.subject_id ORDER BY Students.student_id;
11	Function: Retrieve subject details with credit hours between 2 and 5, including subject ID, credit hours, and the name of the department it belongs to. The results are grouped by subject ID, credit hours, and department name, and ordered by credit hours.	SELECT Subjects.subject_id,Subjects.credithour, sdepartments.sdepartment_name FROM Subjects INNER JOIN sdepartments ON Subjects.sdepartment_id = sdepartments.sdepartment_id where Subjects.credithour between 2 and 5 group by Subjects.subject_id,Subjects.credithour, sdepartments.sdepartment_name ORDER BY Subjects.credithour;
12	Function: Retrieve teacher details from the departments of CS or SE, including the teacher's first name, department name, and department ID. The results are grouped by the teacher's first name, department name, and department ID, and ordered by the department ID.	SELECT Teachers.first_name, departments.department_name, departments.department_id FROM Teachers INNER JOIN departments ON Teachers.department_id = departments.department_id where departments.department_name='CS 'OR departments.department_name='SE' GROUP BY Teachers.first_name, departments.department_name, departments.department_id order by departments.department_id
13	Function: Retrieve student details along with attendance information for students who have attended on October 5, 2022, or are male. Includes all student attributes and attendance ID, and date. The results are ordered by the students' last names.	SELECT Students.*, attendance.attendance_id, attendance.date FROM Students INNER JOIN attendance ON Students.student_id = attendance.student_id where Attendance.date='2022-10-5' or Students.gender='male' ORDER BY Students.last_name;
14	Function: Retrieve fee details for students, including all fee	SELECT fees.*, Students.first_name, Students.last_name, fees.payment_date FROM fees

	information, student's first name, student's last name, and payment date. The results are grouped by student ID.	INNER JOIN Students ON fees.student_id = Students.student_id GROUP BY Students.student_id
15	Function: Retrieve fee defaulter details, including all fee defaulter information, student's first name, student's last name, and the amount due. The results include fee defaulters from students belonging to department ID 1 or are male. The results are ordered by the amount due.	SELECT Feedefaulter.*, Students.first_name, Students.last_name, Feedefaulter.amountdue FROM Feedefaulter INNER JOIN Students ON Feedefaulter.student_id = Students.student_id where Students.sdepartment_id=1 or Students.gender='male' ORDER BY Feedefaulter.amountdue;
16	Function: Retrieve grade details for male students who achieved an 'A' grade. Includes grade ID, student's first name, student's last name, and exam ID. The results are grouped by grade ID, student's first name, student's last name, and exam ID.	SELECT grades.grade_id, Students.first_name, Students.last_name, grades.exam_id FROM grades INNER JOIN Students ON grades.student_id = Students.student_id WHERE Students.gender = 'Male' AND grades.grade = 'A' GROUP BY grades.grade_id, Students.first_name, Students.last_name, grades.exam_id;
17	Function: Retrieve exam details for exams taking place on or after January 1, 2023, and related to subjects with "Math" in their name. Includes all exam information, subject name, and exam date. The results are ordered by exam date.	SELECT exams.*, Subjects.subject_name, exams.exam_date FROM exams INNER JOIN Subjects ON exams.subject_id = Subjects.subject_id WHERE exams.exam_date >= '2023-01-01' AND Subjects.subject_name LIKE '%Math%' ORDER BY exams.exam_date;
18	Function: Retrieve salary details for male teachers whose salary amount is greater than 5000. Includes salary ID, teacher's first	SELECT salary.salary_id, Teachers.first_name, Teachers.last_name, salary.payment_date FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.teacher_id WHERE salary.amount > 5000 AND Teachers.gender = 'Male'

	name, teacher's last name, and payment date. The results are grouped by salary ID, teacher's first name, teacher's last name, and payment date.	`GROUP BY salary.salary_id, Teachers.first_name, Teachers.last_name, salary.payment_date;`
19	Function: Retrieve student details for female students enrolled in the subject with ID 1. Includes all student attributes along with enrollment ID and subject ID. The results are ordered by the students' last names.	`SELECT Students.*, Enrollments.enrollment_id, Enrollments.subject_id` `FROM Students` `INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id` `WHERE Students.gender = 'Female' AND Enrollments.subject_id = 1` `ORDER BY Students.last_name;`
20	Function: Retrieve first names of teachers and their department IDs from the "Science" department, who have a salary amount greater than 5000. The results are ordered by the department ID.	`SELECT Teachers.first_name,Teachers.department_id` `FROM Teachers` `INNER JOIN departments ON Teachers.department_id = departments.department_id` `INNER JOIN salary ON Teachers.teacher_id = salary.teacher_id` `WHERE departments.department_name = 'Science' AND salary.amount > 5000` `order by Teachers.department_id;`
21	Function: Retrieve student IDs for students who are enrolled in subjects with a credit hour greater than 3 or the subject name is "CS". The results are grouped by student ID.	`SELECT Students.student_id` `FROM Students` `INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id` `INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id` `WHERE Subjects.credithour > 3 or Subjects.subject_name='CS'` `group by Students.student_id`
22	Function: Retrieve subject IDs and subject names for subjects that have exams scheduled on either January 1, 2023, or January 3, 2023. The results are grouped by subject ID and subject name.	`SELECT Subjects.subject_id,Subjects.subject_name` `FROM Subjects` `INNER JOIN exams ON Subjects.subject_id = exams.subject_id` `WHERE exams.exam_date = '2023-01-01' or exams.exam_date='2023-01-03'` `group by Subjects.subject_id,Subjects.subject_name`
23	Function: Retrieve first names of teachers and	`SELECT Teachers.first_name,Teachers.subject_id` `FROM Teachers`

	their subject IDs for teachers belonging to the departments of Mathematics or Physics. The results are ordered by the teachers' first names.	INNER JOIN departments ON Teachers.department_id = departments.department_id WHERE departments.department_name IN ('Mathematics', 'Physics') order by Teachers.first_name
24	Function: Retrieve student IDs for students who have an enrollment ID of 1 or are male. The results are grouped by student ID.	SELECT Students.student_id FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id where Enrollments.enrollment_id=1 or Students.gender='male' GROUP BY Students.student_id;
25	Function: Retrieve subject IDs and subject names for subjects that have assigned teachers. The results are ordered by the subject ID.	SELECT Subjects.subject_id,Subjects.subject_name FROM Subjects inner JOIN Teachers ON Subjects.subject_id = Teachers.subject_id WHERE Teachers.teacher_id IS NULL order by Subjects.subject_id;
26	Function: Retrieve student IDs and first names for students born on January 1, 2000, and belonging to the 'Engineering' department. The results are grouped by student ID and first name	SELECT Students.student_id,Students.first_name FROM Students INNER JOIN departments ON Students.sdepartment_id = departments.department_id WHERE Students.date_of_birth = '2000-01-01' AND departments.department_name = 'Engineering' group by Students.student_id,Students.first_name;
27	Function: Retrieve student IDs and department IDs for students who are enrolled in subjects with a credit hour greater than 3 and belong to the 'Science' department. The results are grouped by student ID and department ID, and ordered by student ID.	SELECT Students.student_id,Students.sdepartment_id FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id INNER JOIN departments ON Students.sdepartment_id = departments.department_id WHERE Subjects.credithour > 3 AND departments.department_name = 'Science' group by Students.student_id,Students.sdepartment_id order by Students.student_id;
28	Function: Retrieve first names of teachers and their department IDs for teachers belonging to the 'Mathematics' department or having a salary amount	SELECT Teachers.first_name,Teachers.department_id FROM Teachers INNER JOIN departments ON Teachers.department_id = departments.department_id INNER JOIN salary ON Teachers.teacher_id = salary.teacher_id WHERE departments.department_name = 'Mathematics' OR salary.amount > 5000

	greater than 5000. The results are ordered by the teachers' first names.	`order by Teachers.first_name;`
29	Function: Retrieve grade details for female students or grades related to subjects belonging to department ID 3. Includes all grade information, student's first name, student's last name, and subject name. The results are ordered by the grade ID.	`SELECT grades.*, Students.first_name, Students.last_name, Subjects.subject_name` `FROM grades` `INNER JOIN Students ON grades.student_id = Students.student_id` `INNER JOIN Subjects ON grades.subject_id = Subjects.subject_id` `where Students.gender='female' or Subjects.sdepartment_id=3` `ORDER BY grades.grade_id;`
30	Function: Retrieve subject ID and subject name for the subject with ID 2, where the teacher with ID 1 is assigned to the subject. The results are ordered by the subject ID.	`SELECT Subjects.subject_id,Subjects.subject_name` `FROM Subjects` `inner JOIN Teachers ON Subjects.subject_id = Teachers.subject_id` `WHERE Teachers.teacher_id = 1 and Subjects.subject_id=2` `order by Subjects.subject_id;`

18. Left Joins– 20 Queries

1	Function: Retrieve all student details along with the name of the department they belong to. The results include all student attributes and the department name. The query performs a left join between the "Students" table and the "sdepartments" table based on the department ID.	`SELECT Students.*, sdepartments.sdepartment_name` `FROM Students` `LEFT JOIN sdepartments ON Students.sdepartment_id = sdepartments.sdepartment_id;`

2	Function: Retrieve all subject details along with the first name and last name of the assigned teacher (if any). The results include all subject attributes and the first name and last name of the assigned teacher. The query performs a left join between the "Subjects" table and the "Teachers" table based on the subject ID.	`SELECT Subjects.*, Teachers.first_name, Teachers.last_name FROM Subjects LEFT JOIN Teachers ON Subjects.subject_id = Teachers.subject_id`
3	Function: Retrieve all student details along with their enrollment ID (if any). The results include all student attributes and the enrollment ID. The query performs a left join between the "Students" table and the "Enrollments" table based on the student ID.	`SELECT Students.*, Enrollments.enrollment_id FROM Students LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id;`
4	Function: Retrieve all subject details along with the exam dates (if any). The results include all subject attributes and the exam date. The query performs a left join between the "Subjects" table and the "exams" table based on the subject ID.	`SELECT Subjects.*, exams.exam_date FROM Subjects LEFT JOIN exams ON Subjects.subject_id = exams.subject_id;`

5	Retrieve all student details along with their attendance dates (if any)	SELECT Students.*, attendance.date FROM Students LEFT JOIN attendance ON Students.student_id = attendance.student_id;
6	Student fees details available.	SELECT Students.*, fees.amount, fees.payment_date FROM Students LEFT JOIN fees ON Students.student_id = fees.student_id;
7	Student amount due details included.	SELECT Students.*, Feedefaulter.amountdue FROM Students LEFT JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id;
8	Teacher salary details provided.	SELECT Teachers.*, salary.amount, salary.payment_date FROM Teachers LEFT JOIN salary ON Teachers.teacher_id = salary.teacher_id;
9	Subject grades included.	SELECT Subjects.*, grades.grade FROM Subjects LEFT JOIN grades ON Subjects.subject_id = grades.subject_id;
10	Department with assigned teachers.	SELECT departments.*, Teachers.first_name, Teachers.last_name FROM departments LEFT JOIN Teachers ON departments.department_id = Teachers.department_id;
11	Students without enrollments.	SELECT Students.* FROM Students LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id WHERE Enrollments.enrollment_id IS NULL;
12	Subjects without assigned teachers.	SELECT Subjects.* FROM Subjects LEFT JOIN Teachers ON Subjects.subject_id = Teachers.subject_id WHERE Teachers.teacher_id IS NULL;

13	Students with attendance on June 1, 2023..	SELECT Students.*, attendance.date FROM Students LEFT JOIN attendance ON Students.student_id = attendance.student_id WHERE attendance.date = '2023-06-01';
14	Students with fees paid on June 15, 2023.	SELECT Students.*, fees.amount, fees.payment_date FROM Students LEFT JOIN fees ON Students.student_id = fees.student_id WHERE fees.payment_date = '2023-06-15';
15	Students with fees paid on June 15, 2023.	SELECT Students.*, fees.amount, fees.payment_date FROM Students LEFT JOIN fees ON Students.student_id = fees.student_id WHERE fees.payment_date = '2023-06-15';
16	Students with amount due over 1000.	SELECT Students.*, Feedefaulter.amountdue FROM Students LEFT JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id WHERE Feedefaulter.amountdue > 1000;
17	Teachers with salary payment on June 30, 2023.	SELECT Teachers.*, salary.amount, salary.payment_date FROM Teachers LEFT JOIN salary ON Teachers.teacher_id = salary.teacher_id WHERE salary.payment_date = '2023-06-30';
18	Subjects with grades for exam 1.	SELECT Subjects.*, grades.grade FROM Subjects LEFT JOIN grades ON Subjects.subject_id = grades.subject_id WHERE grades.exam_id = 1;
19	Teachers in the Science department.	SELECT departments.*, Teachers.first_name, Teachers.last_name FROM departments LEFT JOIN Teachers ON departments.department_id = Teachers.department_id WHERE departments.department_name = 'Science'
20	Students with their respective department names, ordered by	SELECT Students.*, sdepartments.sdepartment_name FROM Students LEFT JOIN sdepartments ON Students.sdepartment_id = sdepartments.sdepartment_id ORDER BY Students.student_id ASC;

	student ID in ascending order.	

19. Right Joins– 20 Queries

1	Departments with their corresponding teachers.	`SELECT departments.*, Teachers.first_name, Teachers.last_name FROM departments RIGHT JOIN Teachers ON departments.department_id = Teachers.department_id;`
2	Subjects with their corresponding exam dates.	`SELECT Subjects.*, exams.exam_date FROM Subjects RIGHT JOIN exams ON Subjects.subject_id = exams.subject_id;`
3	Teachers with their corresponding subject names.	`SELECT Teachers.*, Subjects.subject_name FROM Teachers RIGHT JOIN Subjects ON Teachers.subject_id = Subjects.subject_id;`
4	Students with their corresponding enrollment IDs.	`SELECT Students.*, Enrollments.enrollment_id FROM Students RIGHT JOIN Enrollments ON Students.student_id = Enrollments.student_id;`
5	Student attendance with date information.	`SELECT Students.*, attendance.date FROM Students RIGHT JOIN attendance ON Students.student_id = attendance.student_id;`
6	Student fees with amount and payment date.	`SELECT Students.*, fees.amount, fees.payment_date FROM Students RIGHT JOIN fees ON Students.student_id = fees.student_id;`
7	Student fee defaulters with amount due.	`SELECT Students.*, Feedefaulter.amountdue FROM Students RIGHT JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id;`
8	Teachers' salary details with payment information.	`SELECT Teachers.*, salary.amount, salary.payment_date FROM Teachers RIGHT JOIN salary ON Teachers.teacher_id = salary.teacher_id;`

9	Subject grades	SELECT Subjects.*, grades.grade FROM Subjects RIGHT JOIN grades ON Subjects.subject_id = grades.subject_id
10	Subject of student departments	SELECT Subjects.*, sdepartments.sdepartment_name FROM Subjects RIGHT JOIN sdepartments ON Subjects.sdepartment_id = sdepartments.sdepartment_id;
11	Teacher's without subjects	SELECT Teachers.* FROM Teachers RIGHT JOIN Subjects ON Teachers.subject_id = Subjects.subject_id WHERE Subjects.subject_id IS NULL;
12	Subjects without exams	SELECT exams.* FROM exams RIGHT JOIN Subjects ON exams.subject_id = Subjects.subject_id WHERE Subjects.subject_id IS NULL;
13	Student attendance on specific date	SELECT Students.*, attendance.date FROM Students RIGHT JOIN attendance ON Students.student_id = attendance.student_id WHERE attendance.date = '2023-06-01';
14	Students info, fee amount, payment date on a 2023-06-15	SELECT Students.*, fees.amount, fees.payment_date FROM Students RIGHT JOIN fees ON Students.student_id = fees.student_id WHERE fees.payment_date = '2023-06-15';
15	Fee defaulter students where amount > 1000	SELECT Students.*, Feedefaulter.amountdue FROM Students RIGHT JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id WHERE Feedefaulter.amountdue > 1000;
16	Teacher info, salary amount, payment date On 2023-06-30	SELECT Teachers.*, salary.amount, salary.payment_date FROM Teachers RIGHT JOIN salary ON Teachers.teacher_id = salary.teacher_id WHERE salary.payment_date = '2023-06-30';
17	Grades of subject where exam id =1	SELECT Subjects.*, grades.grade FROM Subjects RIGHT JOIN grades ON Subjects.subject_id = grades.subject_id WHERE grades.exam_id = 1;

18	Teachers info , department info where department name is science	SELECT departments.*, Teachers.first_name, Teachers.last_name FROM departments RIGHT JOIN Teachers ON departments.department_id = Teachers.department_id WHERE departments.department_name = 'Science';
19	Student info , sdepartment info in asc order	SELECT Students.*, sdepartments.sdepartment_name FROM Students RIGHT JOIN sdepartments ON Students.sdepartment_id = sdepartments.sdepartment_id ORDER BY Students.student_id ASC
20	Teachers with subject in desc order	SELECT Subjects.*, Teachers.first_name, Teachers.last_name FROM Subjects RIGHT JOIN Teachers ON Subjects.subject_id = Teachers.subject_id ORDER BY Subjects.subject_name DESC

20. Full Outer Joins– 20 Queries

1	Department of teachers	SELECT departments.*, Teachers.first_name, Teachers.last_name FROM departments FULL OUTER JOIN Teachers ON departments.department_id = Teachers.department_id;
2	Exam of subjects	SELECT Subjects.*, exams.exam_date FROM Subjects FULL OUTER JOIN exams ON Subjects.subject_id = exams.subject_id
3	Subject of teachers	SELECT Teachers.*, Subjects.subject_name FROM Teachers FULL OUTER JOIN Subjects ON Teachers.subject_id = Subjects.subject_id;
4	Enrolled students	SELECT Students.*, Enrollments.enrollment_id FROM Students

		FULL OUTER JOIN Enrollments ON Students.student_id = Enrollments.student_id
5	Attendance of students	SELECT Students.*, attendance.date FROM Students FULL OUTER JOIN attendance ON Students.student_id = attendance.student_id
6	Fees of students	SELECT Students.*, fees.amount, fees.payment_date FROM Students FULL OUTER JOIN fees ON Students.student_id = fees.student_id;
7	Fee defaulter students	SELECT Students.*, Feedefaulter.amountdue FROM Students FULL OUTER JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id;
8	Teacher salary info	SELECT Teachers.*, salary.amount, salary.payment_date FROM Teachers FULL OUTER JOIN salary ON Teachers.teacher_id = salary.teacher_id;
9	Grades of subjects	SELECT Subjects.*, grades.grade FROM Subjects FULL OUTER JOIN grades ON Subjects.subject_id = grades.subject_id;
10	Subjects of student department	SELECT Subjects.*, sdepartments.sdepartment_name FROM Subjects FULL OUTER JOIN sdepartments ON Subjects.sdepartment_id = sdepartments.sdepartment_id
11	Teacher without subjects	SELECT Teachers.* FROM Teachers FULL OUTER JOIN Subjects ON Teachers.subject_id = Subjects.subject_id WHERE Subjects.subject_id IS NULL;

12	Subjects without exam	SELECT exams.* FROM exams FULL OUTER JOIN Subjects ON exams.subject_id = Subjects.subject_id WHERE Subjects.subject_id IS NULL;
13	Attendance of student on 2023-06-01	SELECT Students.*, attendance.date FROM Students FULL OUTER JOIN attendance ON Students.student_id = attendance.student_id WHERE attendance.date = '2023-06-01'
14	Student info , fee payment date and amount on 2023-06-15	SELECT Students.*, fees.amount, fees.payment_date FROM Students FULL OUTER JOIN fees ON Students.student_id = fees.student_id WHERE fees.payment_date = '2023-06-15'
15	Fee defaulter students info where amount is greater than 1000	SELECT Students.*, feedefaulter.amountdue FROM Students FULL OUTER JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id WHERE Feedefaulter.amountdue > 1000;
16	Teacher salary amount and date on specific date	SELECT Teachers.*, salary.amount, salary.payment_date FROM Teachers FULL OUTER JOIN salary ON Teachers.teacher_id = salary.teacher_id WHERE salary.payment_date = '2023-06-30';
17	Grade of student where subject id =1	SELECT Students.*, grades.grade FROM Students FULL OUTER JOIN grades ON Students.student_id = grades.student_id WHERE grades.subject_id = 1;
18	Enrollment of student where subject I d=1	SELECT Students.*, Enrollments.enrollment_id

		FROM Students FULL OUTER JOIN Enrollments ON Students.student_id = Enrollments.student_id WHERE Enrollments.subject_id = 1;
19	Teacher info and department info where teacher name is john and last name is doe	SELECT departments.*, Teachers.first_name, Teachers.last_name FROM departments FULL OUTER JOIN Teachers ON departments.department_id = Teachers.department_id WHERE Teachers.first_name = 'John' AND Teachers.last_name = 'Doe';
20	Subject and exam info where exam is on 2023-07-15	SELECT Subjects.*, exams.exam_date FROM Subjects FULL OUTER JOIN exams ON Subjects.subject_id = exams.subject_id WHERE exams.exam_date = '2023-07-15';

21. Stored Procedures without parameters– 25 Queries

1	GetAllStudents	CREATE PROCEDURE GetAllStudents AS BEGIN SELECT * FROM Students; END;
2	GetAllStudents	CREATE PROCEDURE GetAllDepartments AS BEGIN SELECT * FROM departments; END;
3	GetAllSubjects	CREATE PROCEDURE GetAllSubjects AS BEGIN SELECT * FROM Subjects; END;
4	GetAllTeachers	CREATE PROCEDURE GetAllTeachers AS BEGIN SELECT * FROM Teachers;

		END;
5	GetAllEnrollments	CREATE PROCEDURE GetAllEnrollments AS BEGIN SELECT * FROM Enrollments; END;
6	GetAllAttendance	CREATE PROCEDURE GetAllAttendance AS BEGIN SELECT * FROM attendance; END;
7	GetAllFees	CREATE PROCEDURE GetAllFees AS BEGIN SELECT * FROM fees; END;
8	GetAllFeeDefaulters	CREATE PROCEDURE GetAllFeeDefaulters AS BEGIN SELECT * FROM Feedefaulter; END;
9	GetAllSalaries	CREATE PROCEDURE GetAllSalaries AS BEGIN SELECT * FROM salary; END;
10	GetAllExams	CREATE PROCEDURE GetAllExams AS BEGIN SELECT * FROM exams; END;
11	GetAllGrades	CREATE PROCEDURE GetAllGrades AS BEGIN SELECT * FROM grades; END;
12	GetStudentsWithDepartments	CREATE PROCEDURE GetStudentsWithDepartments AS BEGIN SELECT Students.*, departments.department_name FROM Students

		INNER JOIN departments ON Students.sdepartment_id = departments.department_id; END;
13	GetTeachersWithDepartments	CREATE PROCEDURE GetTeachersWithDepartments AS BEGIN SELECT Teachers.*, departments.department_name FROM Teachers INNER JOIN departments ON Teachers.department_id = departments.department_id; END;
14	GetSubjectsWithDepartments	CREATE PROCEDURE GetSubjectsWithDepartments AS BEGIN SELECT Subjects.*, departments.department_name FROM Subjects INNER JOIN departments ON Subjects.department_id = departments.department_id; END;
15	GetEnrollmentsWithStudents	CREATE PROCEDURE GetEnrollmentsWithStudents AS BEGIN SELECT Enrollments.*, Students.first_name, Students.last_name FROM Enrollments INNER JOIN Students ON Enrollments.student_id = Students.student_id; END;
16	GetAttendanceWithStudents	CREATE PROCEDURE GetAttendanceWithStudents AS BEGIN SELECT attendance.*, Students.first_name, Students.last_name FROM attendance INNER JOIN Students ON attendance.student_id = Students.student_id; END;
17	GetFeesWithStudentsAndDepartments	CREATE PROCEDURE GetFeesWithStudentsAndDepartments

		AS BEGIN SELECT fees.*, Students.first_name, Students.last_name, departments.department_name FROM fees INNER JOIN Students ON fees.student_id = Students.student_id INNER JOIN departments ON fees.sdepartment_id = departments.department_id; END;
18	GetFeeDefaultersWithStudentsAndDepartments	CREATE PROCEDURE GetFeeDefaultersWithStudentsAndDepartments AS BEGIN SELECT Feedefaulter.*, Students.first_name, Students.last_name, departments.department_name FROM Feedefaulter INNER JOIN Students ON Feedefaulter.student_id = Students.student_id INNER JOIN departments ON Feedefaulter.sdepartment_id = departments.department_id; END;
19	GetSalariesWithTeachersAndDepartments	CREATE PROCEDURE GetSalariesWithTeachersAndDepartments AS BEGIN SELECT salary.*, Teachers.first_name, Teachers.last_name, departments.department_name FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.teacher_id INNER JOIN departments ON salary.department_id = departments.department_id; END;
20	GetTeachersWithAssignedSubjects	CREATE PROCEDURE GetTeachersWithAssignedSubjects AS BEGIN SELECT Teachers.*, Subjects.subject_name FROM Teachers

		INNER JOIN Subjects ON Teachers.subject_id = Subjects.subject_id; END;
21	GetGradesWithStudentsSubjectsAndExams	CREATE PROCEDURE GetGradesWithStudentsSubjectsAndExams AS BEGIN SELECT grades.*, Students.first_name, Students.last_name, Subjects.subject_name, exams.exam_date FROM grades INNER JOIN Students ON grades.student_id = Students.student_id INNER JOIN Subjects ON grades.subject_id = Subjects.subject_id INNER JOIN exams ON grades.exam_id = exams.exam_id; END;
22	GetStudentsWithEnrolledSubjects	CREATE PROCEDURE GetStudentsWithEnrolledSubjects AS BEGIN SELECT Students.*, Subjects.subject_name FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id; END;
23	GetStudentsWithAttendance	CREATE PROCEDURE GetStudentsWithAttendance AS BEGIN SELECT Students.*, attendance.date FROM Students INNER JOIN attendance ON Students.student_id = attendance.student_id; END;
24	GetStudentsWithGradesAndExams	CREATE PROCEDURE GetStudentsWithGradesAndExams AS BEGIN SELECT Students.*, grades.grade, exams.exam_date FROM Students INNER JOIN grades ON Students.student_id = grades.student_id

		INNER JOIN exams ON grades.exam_id = exams.exam_id; END;
25	GetStudentsWithFeesAndDepartments	CREATE PROCEDURE GetStudentsWithFeesAndDepartments AS BEGIN SELECT Students.*, fees.amount, departments.department_name FROM Students INNER JOIN fees ON Students.student_id = fees.student_id INNER JOIN departments ON Students.sdepartment_id = departments.department_id; END; ---

22. Stored Procedures with parameters– 25 Queries

1	GetStudentWithEnrolledSubjects	CREATE PROCEDURE GetStudentWithEnrolledSubjects @student_id INT AS BEGIN SELECT Students.*, Subjects.subject_name FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id WHERE Students.student_id = @student_id; END;
2	GetTeacherWithAssignedSubjects	CREATE PROCEDURE GetTeacherWithAssignedSubjects @teacher_id INT AS BEGIN SELECT Teachers.*, Subjects.subject_name FROM Teachers INNER JOIN Subjects ON Teachers.subject_id = Subjects.subject_id WHERE Teachers.teacher_id = @teacher_id; END;

3	GetStudentWithAttendance	CREATE PROCEDURE GetStudentWithAttendance @student_id INT AS BEGIN SELECT Students.*, attendance.date FROM Students INNER JOIN attendance ON Students.student_id = attendance.student_id WHERE Students.student_id = @student_id; END;
4	GetStudentWithFeesAndDepartments	CREATE PROCEDURE GetStudentWithFeesAndDepartments @student_id INT AS BEGIN SELECT Students.*, fees.amount, departments.department_name FROM Students INNER JOIN fees ON Students.student_id = fees.student_id INNER JOIN departments ON Students.sdepartment_id = departments.department_id WHERE Students.student_id = @student_id; END;
5	GetStudentWithGradesAndExams	CREATE PROCEDURE GetStudentWithGradesAndExams @student_id INT AS BEGIN SELECT Students.*, grades.grade, exams.exam_date FROM Students INNER JOIN grades ON Students.student_id = grades.student_id INNER JOIN exams ON grades.exam_id = exams.exam_id WHERE Students.student_id = @student_id; END;
6	GetStudentsFromDepartmentWithEnrolledSubjects	CREATE PROCEDURE GetStudentsFromDepartmentWithEnrolledSubjects @department_id INT AS BEGIN

		SELECT Students.*, Subjects.subject_name FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id INNER JOIN Subjects ON Enrollments.subject_id = Subjects.subject_id WHERE Students.sdepartment_id = @department_id; END;
7	GetTeachersFromDepartmentWithAssignedSubjects	CREATE PROCEDURE GetTeachersFromDepartmentWithAssignedSubjects @department_id INT AS BEGIN SELECT Teachers.*, Subjects.subject_name FROM Teachers INNER JOIN Subjects ON Teachers.subject_id = Subjects.subject_id WHERE Teachers.department_id = @department_id; END;
8	GetStudentsFromDepartmentWithAttendance	CREATE PROCEDURE GetStudentsFromDepartmentWithAttendance @department_id INT AS BEGIN SELECT Students.*, attendance.date FROM Students INNER JOIN attendance ON Students.student_id = attendance.student_id WHERE Students.sdepartment_id = @department_id; END;
9	GetStudentsFromDepartmentWithFeesAndDepartments	CREATE PROCEDURE GetStudentsFromDepartmentWithFeesAndDepartments @department_id INT AS BEGIN SELECT Students.*, fees.amount, departments.department_name FROM Students INNER JOIN fees ON Students.student_id = fees.student_id

		INNER JOIN departments ON Students.sdepartment_id = departments.department_id WHERE Students.sdepartment_id = @department_id; END;
10	GetStudentsFromDepartmentWithGradesAndExams	CREATE PROCEDURE GetStudentsFromDepartmentWithGradesAndExams @department_id INT AS BEGIN SELECT Students.*, grades.grade, exams.exam_date FROM Students INNER JOIN grades ON Students.student_id = grades.student_id INNER JOIN exams ON grades.exam_id = exams.exam_id WHERE Students.sdepartment_id = @department_id; END;
11	GetStudentsEnrolledInSubject	CREATE PROCEDURE GetStudentsEnrolledInSubject @subject_id INT AS BEGIN SELECT Students.* FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id WHERE Enrollments.subject_id = @subject_id; END;
12	GetTeachersAssignedToSubject	CREATE PROCEDURE GetTeachersAssignedToSubject @subject_id INT AS BEGIN SELECT Teachers.* FROM Teachers WHERE Teachers.subject_id = @subject_id; END;
13	GetStudentsWithExamAttendance	CREATE PROCEDURE GetStudentsWithExamAttendance @exam_date DATE AS BEGIN SELECT Students.*

		FROM Students INNER JOIN attendance ON Students.student_id = attendance.student_id INNER JOIN exams ON attendance.subject_id = exams.subject_id WHERE exams.exam_date = @exam_date; END;
14	GetStudentsWithFeesAmount	CREATE PROCEDURE GetStudentsWithFeesAmount @fees_amount INT AS BEGIN SELECT Students.* FROM Students INNER JOIN fees ON Students.student_id = fees.student_id WHERE fees.amount = @fees_amount; END;
15	GetStudentsFromDepartmentWithFeeDue	CREATE PROCEDURE GetStudentsFromDepartmentWithFeeDue @department_id INT AS BEGIN SELECT Students.* FROM Students INNER JOIN Feedefaulter ON Students.student_id = Feedefaulter.student_id WHERE Students.sdepartment_id = @department_id AND Feedefaulter.amountdue> 0; END;
16	GetStudentsWithSpecificGrade	CREATE PROCEDURE GetStudentsWithSpecificGrade @subject_id INT, @grade VARCHAR(20) AS BEGIN SELECT Students.* FROM Students INNER JOIN grades ON Students.student_id = grades.student_id WHERE grades.subject_id = @subject_id AND grades.grade = @grade; END;
17	GetTeachersWithSpecificSalary	CREATE PROCEDURE GetTeachersWithSpecificSalary @salary_amount INT

		AS BEGIN SELECT Teachers.* FROM Teachers INNER JOIN salary ON Teachers.teacher_id = salary.teacher_id WHERE salary.amount = @salary_amount; END;
18	GetTeachersFromDepartmentWithSalaryPayment	CREATE PROCEDURE GetTeachersFromDepartmentWithSalaryPayment @department_id INT, @payment_date DATE AS BEGIN SELECT Teachers.* FROM Teachers INNER JOIN salary ON Teachers.teacher_id = salary.teacher_id WHERE Teachers.department_id = @department_id AND salary.payment_date = @payment_date end
19	GetStudentsWithExamAndGrade	CREATE PROCEDURE GetStudentsWithExamAndGrade @exam_date DATE, @grade VARCHAR(20) AS BEGIN SELECT Students.* FROM Students INNER JOIN grades ON Students.student_id = grades.student_id INNER JOIN exams ON grades.exam_id = exams.exam_id WHERE exams.exam_date = @exam_date AND grades.grade = @grade; END
20	GetStudentsByDepartment	CREATE PROCEDURE GetStudentsByDepartment @department_id INT AS BEGIN SELECT Students.* FROM Students INNER JOIN Enrollments ON Students.student_id = Enrollments.student_id INNER JOIN departments ON Enrollments.sdepartment_id = departments.department_id

		WHERE departments.department_id = @department_id; END;
21	getfeesofstud	create procedure getfeesofstud @stuid int as begin select payment_date from Fees where student_id=@stuid end
22	getgrade	create procedure getgrade @stuid int as begin select grade from grades where student_id=@stuid end
23	GetStudentsByDepartment	CREATE PROCEDURE GetStudentsByDepartment @departmentName VARCHAR(50) AS BEGIN SELECT * FROM Students s INNER JOIN sdepartments sd ON s.sdepartment_id = sd.sdepartment_id WHERE sd.sdepartment_name LIKE '%' + @departmentName + '%' END;
24	GetEnrollmentsBySubject	CREATE PROCEDURE GetEnrollmentsBySubject @subjectName VARCHAR(50) AS BEGIN SELECT * FROM Enrollments e INNER JOIN Subjects s ON e.subject_id = s.subject_id WHERE s.subject_name LIKE '%' + @subjectName + '%' END;
25	GetTeachersByGender	CREATE PROCEDURE GetTeachersByGender @gender VARCHAR(10) AS BEGIN SELECT * FROM Teachers WHERE gender = @gender END;

23. Stored Procedures with parameters using logical operators and group by– 30 Queries

1	GetStudentCountByDepartmentAndCreditHour	CREATE PROCEDURE GetStudentCountByDepartmentAndCreditHour @department_id INT, @credit_hour_threshold INT AS BEGIN SELECT s.sdepartment_id, COUNT(*) AS student_count FROM Students s INNER JOIN Enrollments e ON s.student_id = e.student_id INNER JOIN Subjects su ON e.subject_id = su.subject_id WHERE s.sdepartment_id = @department_id AND su.credithour > @credit_hour_threshold GROUP BY s.sdepartment_id order by s.sdepartment_id END;
2	GetStudentCountByDepartmentAndGender	CREATE PROCEDURE GetStudentCountByDepartmentAndgeender @department_id INT, @gender VARCHAR(10) AS BEGIN SELECT sdepartment_id AS department_id, gender, COUNT(*) AS student_count FROM Students WHERE sdepartment_id = @department_id AND gender = @gender GROUP BY sdepartment_id, gender order by sdepartment_id END;
3	GetStudentCountByDepartmentAndGrade	CREATE PROCEDURE GetStudentCountByDepartmentAndGrades @department_id INT, @grade VARCHAR(10) AS BEGIN SELECT s.sdepartment_id AS department_id, g.grade, COUNT(*) AS student_count FROM Students s INNER JOIN grades g ON s.student_id = g.student_id

		WHERE s.sdepartment_id = @department_id AND g.grade = @grade GROUP BY s.sdepartment_id, g.grade order by s.sdepartment_id END;
4	GetTeacherCountByDepartmentAndGender	CREATE PROCEDURE GetTeacherCountByDepartmentAndGender @department_id INT, @gender VARCHAR(10) AS BEGIN SELECT department_id, gender, COUNT(*) AS teacher_count FROM Teachers WHERE department_id = @department_id AND gender = @gender GROUP BY department_id, gender order by department_id END;
5	GetEnrollmentCountBySubjectAndDepartment	CREATE PROCEDURE GetEnrollmentCountBySubjectAndDepartment @subject_id INT, @department_id INT AS BEGIN SELECT e.subject_id, s.sdepartment_id, COUNT(*) AS enrollment_count FROM Enrollments e INNER JOIN Students s ON e.student_id = s.student_id INNER JOIN sdepartments sd ON s.sdepartment_id = sd.sdepartment_id WHERE e.subject_id = @subject_id AND sd.sdepartment_id = @department_id GROUP BY e.subject_id, s.sdepartment_id order by s.sdepartment_id END;
6	GetAttendanceCountByDate	CREATE PROCEDURE GetAttendanceCountByDate @date DATE, @st_id INT AS BEGIN SELECT date, student_id, COUNT(*) AS attendance_count FROM Attendance WHERE date = @date AND student_id = @st_id GROUP BY date, student_id order by student_id END;

7	GetStudentsByDepartmentAndContactNumber	CREATE PROCEDURE GetStudentsByDepartmentAndContactNumber @departmentName VARCHAR(50), @contactNumber VARCHAR(20) AS BEGIN SELECT s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email FROM Students s INNER JOIN sdepartments sd ON s.sdepartment_id = sd.sdepartment_id WHERE sd.sdepartment_name LIKE '%' + @departmentName + '%' AND s.contact_number LIKE '%' + @contactNumber + '%' GROUP BY s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email END;
8	GetAverageSalaryByDepartmentAndGender	CREATE PROCEDURE GetAverageSalaryByDepartmentAndGender @department_id INT, @gender VARCHAR(10) AS BEGIN SELECT Teachers.department_id, Teachers.gender, AVG(salary.amount) AS average_salary FROM salary INNER JOIN Teachers ON salary.teacher_id = Teachers.teacher_id WHERE Teachers.department_id = @department_id AND Teachers.gender = @gender GROUP BY Teachers.department_id, Teachers.gender order by Teachers.department_id END;
9	GetTotalAttendanceCountByDepartmentAndSubjects	CREATE PROCEDURE GetTotalAttendanceCountByDepartmentAndSubjects @s_id INT, @subject_id INT AS BEGIN SELECT Students.student_id, Attendance.subject_id, COUNT(*) AS attendance_count FROM attendance

		INNER JOIN Students ON attendance.student_id = Students.student_id WHERE Students.sdepartment_id = @s_id AND attendance.subject_id = @subject_id GROUP BY Students.student_id, Attendance.subject_id order by Students.student_id END;
10	GetTeachersBySubjectAndGender	CREATE PROCEDURE GetTeachersBySubjectAndGender @subjectName VARCHAR(50), @gender VARCHAR(10) AS BEGIN SELECT t.* FROM Teachers t INNER JOIN Subjects s ON t.subject_id = s.subject_id WHERE s.subject_name LIKE '%' + @subjectName + '%' AND t.gender = @gender GROUP BY t.teacher_id, t.department_id, t.first_name, t.last_name, t.date_of_birth, t.gender, t.Aaddress, t.contact_number, t.email, t.subject_id; END; ----
11	GetStudentsByDepartmentAndEmailDomain	CREATE PROCEDURE GetStudentsByDepartmentAndEmailDomain @departmentName VARCHAR(50), @emailDomain VARCHAR(50) AS BEGIN SELECT * FROM Students WHERE sdepartment_id IN (SELECT sdepartment_id FROM sdepartments WHERE sdepartment_name LIKE '%' + @departmentName + '%') AND email LIKE '%' + @emailDomain GROUP BY student_id, sdepartment_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email; END;
12	GetAverageAmountDueByDepartmentAndGender	CREATE PROCEDURE GetAverageAmountDueByDepartmentAndGender @department_id INT, @gender VARCHAR(10)

		AS BEGIN SELECT Students.sdepartment_id AS department_id, Students.gender, AVG(Feedefaulter.amountdue) AS average_amount_due FROM Feedefaulter INNER JOIN Students ON Feedefaulter.student_id = Students.student_id WHERE Students.sdepartment_id = @department_id AND Students.gender = @gender GROUP BY Students.sdepartment_id, Students.gender; END;
13	GetGenderCountByDepartment	CREATE PROCEDURE GetGenderCountByDepartment @department_id INT, @name VARCHAR(20) AS BEGIN SELECT gender, COUNT(*) AS gender_count FROM Students WHERE sdepartment_id = @department_id AND first_name LIKE '%' + @name + '%' GROUP BY gender; END;
14	GetEnrollmentsByStudentAndSubject	CREATE PROCEDURE GetEnrollmentsByStudentAndSubject @studentID INT, @subjectID INT AS BEGIN SELECT student_id, subject_id, COUNT(*) AS enrollment_count FROM Enrollments WHERE student_id = @studentID AND subject_id = @subjectID GROUP BY student_id, subject_id END;
15	GetStudentsByDepartmentAndGender	CREATE PROCEDURE GetStudentsByDepartmentAndGender @departmentName VARCHAR(50), @gender VARCHAR(10) AS BEGIN SELECT s.student_id, s.sdepartment_id, s.first_name,

		s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email FROM Students s INNER JOIN sdepartments sd ON s.sdepartment_id = sd.sdepartment_id WHERE sd.sdepartment_name LIKE '%' + @departmentName + '%' AND s.gender = @gender GROUP BY s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email END;
16	GetTeachersBySubjectAndEmail	CREATE PROCEDURE GetTeachersBySubjectAndEmail @subjectName VARCHAR(50), @email VARCHAR(50) AS BEGIN SELECT t.teacher_id, t.department_id, t.first_name, t.last_name, t.date_of_birth, t.gender, t.Aaddress, t.contact_number, t.email, t.subject_id FROM Teachers t INNER JOIN Subjects s ON t.subject_id = s.subject_id WHERE s.subject_name LIKE '%' + @subjectName + '%' AND t.email LIKE '%' + @email + '%' GROUP BY t.teacher_id, t.department_id, t.first_name, t.last_name, t.date_of_birth, t.gender, t.Aaddress, t.contact_number, t.email, t.subject_id END;
17	GetStudentsByDepartmentAndAddress	CREATE PROCEDURE GetStudentsByDepartmentAndAddress @departmentName VARCHAR(50), @address VARCHAR(100) AS BEGIN SELECT s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email FROM Students s INNER JOIN sdepartments sd ON s.sdepartment_id = sd.sdepartment_id WHERE sd.sdepartment_name LIKE '%' + @departmentName + '%' AND s.Aaddress LIKE '%' + @address + '%'

		GROUP BY s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email END;
18	GetStudentCountByDepartmentAndGender	CREATE PROCEDURE GetStudentCountByDepartmentAndGender @department_id INT, @gender VARCHAR(10) AS BEGIN SELECT COUNT(*) AS student_count FROM Students WHERE sdepartment_id = @department_id AND gender = @gender; END;
19	GetAverageFeesByDepartmentAndGender	CREATE PROCEDURE GetAverageFeesByDepartmentAndGender @department_id INT, @gender VARCHAR(10) AS BEGIN SELECT Students.sdepartment_id, Students.gender, AVG(fees.amount) AS average_fees FROM Students INNER JOIN fees ON Students.student_id = fees.student_id WHERE Students.sdepartment_id = @department_id AND Students.gender = @gender GROUP BY Students.sdepartment_id, Students.gender; END;
20	GetAverageCreditHourByDepartmentExcludingSubject	CREATE PROCEDURE GetAverageCreditHourByDepartmentExcludingSubject @department_id INT, @subject_id INT AS BEGIN SELECT s.sdepartment_id, AVG(s.credithour) AS average_credit_hour FROM Subjects s WHERE s.sdepartment_id = @department_id AND s.subject_id != @subject_id GROUP BY s.sdepartment_id; END;
21	GetTeachersByDepartmentAndLastName	CREATE PROCEDURE GetTeachersByDepartmentAndLastName

		@departmentName VARCHAR(50), @lastName VARCHAR(50) AS BEGIN SELECT t.teacher_id, t.first_name, t.last_name, t.date_of_birth, t.gender, t.Aaddress, t.contact_number, t.email, t.subject_id FROM Teachers t INNER JOIN departments d ON t.department_id = d.department_id WHERE d.department_name LIKE '%' + @departmentName + '%' AND t.last_name = @lastName GROUP BY t.teacher_id, t.first_name, t.last_name, t.date_of_birth, t.gender, t.Aaddress, t.contact_number, t.email, t.subject_id; END;
22	GetStudentsByEnrollmentAndGender	CREATE PROCEDURE GetStudentsByEnrollmentAndGender @enrollmentID INT, @gender VARCHAR(10) AS BEGIN SELECT s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email FROM Students s INNER JOIN Enrollments e ON s.student_id = e.student_id WHERE e.enrollment_id = @enrollmentID AND s.gender = @gender GROUP BY s.student_id, s.sdepartment_id, s.first_name, s.last_name, s.date_of_birth, s.gender, s.Aaddress, s.contact_number, s.email; END;
23	GetMaxAgeByDepartment	CREATE PROCEDURE GetMaxAgeByDepartment @department_id INT, @gender varchar(20) AS BEGIN SELECT Teachers.department_id, MAX(DATEDIFF(YEAR, Teachers.date_of_birth, GETDATE())) AS max_age FROM Teachers WHERE Teachers.department_id = @department_id and gender= @gender GROUP BY Teachers.department_id; END;

24	GetTotalFeesPaidByGender	CREATE PROCEDURE GetTotalFeesPaidByGender @gender VARCHAR(10), @id INT AS BEGIN SELECT Students.gender, SUM(fees.amount) AS total_fees_paid FROM Students INNER JOIN fees ON Students.student_id = fees.student_id WHERE Students.gender = @gender AND Students.sdepartment_id = @id GROUP BY Students.gender; END;
25	GetMinCreditHourByDepartment	CREATE PROCEDURE GetMinCreditHourByDepartment @department_id INT AS BEGIN SELECT MIN(credithour) AS min_credit_hour FROM Subjects WHERE sdepartment_id = @department_id GROUP BY sdepartment_id; END;
26	GetAverageSalaryByGender	CREATE PROCEDURE GetAverageSalaryByGender @gender VARCHAR(10), @department_id INT AS BEGIN SELECT t.gender, AVG(s.amount) AS average_salary FROM salary s INNER JOIN Teachers t ON s.teacher_id = t.teacher_id WHERE t.gender = @gender AND t.department_id = @department_id GROUP BY t.gender; END;
27	GetTeachersBySubjectAndGender	CREATE PROCEDURE GetTeachersBySubjectAndGender @subjectName VARCHAR(50), @gender VARCHAR(10) AS BEGIN SELECT t.* FROM Teachers t

		INNER JOIN Subjects s ON t.subject_id = s.subject_id WHERE s.subject_name LIKE '%' + @subjectName + '%' AND t.gender = @gender END
28	GetEnrollmentsByStudentAndSubject	CREATE PROCEDURE GetEnrollmentsByStudentAndSubject @studentID INT, @subjectID INT AS BEGIN SELECT enrollment_id, subject_id FROM Enrollments WHERE student_id = @studentID AND subject_id = @subjectID END
29	GetTeachersBySubjectAndDateOfBirthRange	CREATE PROCEDURE GetTeachersBySubjectAndDateOfBirthRange @subjectName VARCHAR(50), @startDate DATE, @endDate DATE AS BEGIN SELECT t.first_name, t.department_id FROM Teachers t INNER JOIN Subjects s ON t.subject_id = s.subject_id WHERE s.subject_name LIKE '%' + @subjectName + '%' AND t.date_of_birth BETWEEN @startDate AND @endDate GROUP BY t.first_name, t.department_id END;
30	GetStudentsByDepartmentAndDOBRange	CREATE PROCEDURE GetStudentsByDepartmentAndDOBRange @departmentName VARCHAR(50), @startDate DATE, @endDate DATE AS BEGIN SELECT s.first_name, s.last_name FROM Students s INNER JOIN sdepartments sd ON s.sdepartment_id = sd.sdepartment_id WHERE sd.sdepartment_name LIKE '%' + @departmentName + '%' AND s.date_of_birth BETWEEN @startDate AND @endDate GROUP BY s.first_name, s.last_name

		END;

24. DML Triggers INSERT – 20 Queries

1	TR_Students_FORINSERTED: Audit new student insertions.	CREATE TRIGGER TR_Students_FORINSERTED ON Students AFTER INSERT AS BEGIN DECLARE @student_id INT; DECLARE @first_name VARCHAR(100); SELECT @student_id = student_id, @first_name = first_name FROM inserted; INSERT INTO StudentAudit VALUES ('New student with id=' + CAST(@student_id AS VARCHAR(10)) + '&name=' + @first_name + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
2	TR_Subjects_FORINSERTED: Audit new subject insertions.	CREATE TRIGGER TR_Subjects_FORINSERTED ON Subjects AFTER INSERT AS BEGIN DECLARE @subject_id INT; DECLARE @subject_name VARCHAR(100); SELECT @subject_id = subject_id, @subject_name = subject_name FROM inserted; INSERT INTO SubjectAudit VALUES ('New subject with id=' + CAST(@subject_id AS VARCHAR(10)) + '&name=' + @subject_name + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
3	TR_Departments_FORINSERTED: Audit new department insertions.	INSERT INTO Subjects (subject_id,subject_name) VALUES (1,'ENG'); CREATE TRIGGER TR_Departments_FORINSERTED ON Departments AFTER INSERT AS

		BEGIN DECLARE @department_id INT; DECLARE @department_name VARCHAR(100); SELECT @department_id = department_id, @department_name = department_name FROM inserted; INSERT INTO DepartmentAudit VALUES ('New department with id=' + CAST(@department_id AS VARCHAR(10)) + '&name=' + @department_name + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
4	TR_StudentDepartments_FORINSERTED: Audit new student department insertions.	INSERT INTO Departments (department_id,department_name) VALUES (1,'ENG'); CREATE TRIGGER TR_StudentDepartments_FORINSERTED ON SDepartments AFTER INSERT AS BEGIN DECLARE @sdepartment_id INT; DECLARE @sdepartment_name VARCHAR(100); SELECT @sdepartment_id = sdepartment_id, @sdepartment_name = sdepartment_name FROM inserted; INSERT INTO StudentDepartmentAudit VALUES ('New student department with id=' + CAST(@sdepartment_id AS VARCHAR(10)) + '&name=' + @sdepartment_name + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
5	TR_Teachers_FORINSERTED: Audit new teacher insertions.	CREATE TRIGGER TR_Teachers_FORINSERTED ON Teachers AFTER INSERT AS BEGIN DECLARE @teacher_id INT; DECLARE @first_name VARCHAR(100); SELECT @teacher_id = teacher_id, @first_name = first_name FROM inserted; INSERT INTO TeacherAudit VALUES (

		'New teacher with id=' + CAST(@teacher_id AS VARCHAR(10)) + '&name=' + @first_name + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
6	TR_Enrollments_FORINSERTED: Audit new enrollment insertions..	CREATE TRIGGER TR_Enrollments_FORINSERTED ON Enrollments AFTER INSERT AS BEGIN DECLARE @enrollment_id INT; DECLARE @student_id INT; SELECT @enrollment_id = enrollment_id, @student_id = student_id FROM inserted; INSERT INTO EnrollmentAudit VALUES ('New enrollment with id=' + CAST(@enrollment_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
7	TR_Attendance_FORINSERTED: Audit new attendance record insertions.	CREATE TRIGGER TR_Attendance_FORINSERTED ON Attendance AFTER INSERT AS BEGIN DECLARE @attendance_id INT; DECLARE @student_id INT; DECLARE @subject_id INT; SELECT @attendance_id = attendance_id, @student_id = student_id, @subject_id = subject_id FROM inserted; INSERT INTO AttendanceAudit VALUES ('New attendance record with id=' + CAST(@attendance_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
8	TR_Fees_FORINSERTED: Audit new fee record insertions.	CREATE TRIGGER TR_Fees_FORINSERTED ON Fees AFTER INSERT

		AS BEGIN DECLARE @fee_id INT; DECLARE @student_id INT; DECLARE @sdepartment_id INT; SELECT @fee_id = fee_id, @student_id = student_id, @sdepartment_id = sdepartment_id FROM inserted; INSERT INTO FeesAudit VALUES ('New fee record with id=' + CAST(@fee_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&sdepartment_id=' + CAST(@sdepartment_id AS VARCHAR(10)) + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
9	TR_FeeDefaulter_FORINSERTED: Audit new fee defaulter record insertions.	CREATE TRIGGER TR_FeeDefaulter_FORINSERTED ON FeeDefaulter AFTER INSERT AS BEGIN DECLARE @feedef_id INT; DECLARE @student_id INT; DECLARE @sdepartment_id INT; SELECT @feedef_id = feedef_id, @student_id = student_id, @sdepartment_id = sdepartment_id FROM inserted; INSERT INTO FeeDefaulterAudit VALUES ('New fee defaulter record with id=' + CAST(@feedef_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&sdepartment_id=' + CAST(@sdepartment_id AS VARCHAR(10)) + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
10	TR_Salary_FORINSERTED: Audit new salary record insertions	CREATE TRIGGER TR_Salary_FORINSERTED ON Salary AFTER INSERT AS BEGIN DECLARE @salary_id INT; DECLARE @teacher_id INT; DECLARE @department_id INT;

		SELECT @salary_id = salary_id, @teacher_id = teacher_id, @department_id = department_id FROM inserted; INSERT INTO SalaryAudit VALUES ('New salary record with id=' + CAST(@salary_id AS VARCHAR(10)) + '&teacher_id=' + CAST(@teacher_id AS VARCHAR(10)) + '&department_id=' + CAST(@department_id AS VARCHAR(10)) + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
11	TR_Exams_FORINSERTED: Audit new exam record insertions.	CREATE TRIGGER TR_Exams_FORINSERTED ON Exams AFTER INSERT AS BEGIN DECLARE @exam_id INT; DECLARE @subject_id INT; DECLARE @exam_date DATE; SELECT @exam_id = exam_id, @subject_id = subject_id, @exam_date = exam_date FROM inserted; INSERT INTO ExamsAudit VALUES ('New exam record with id=' + CAST(@exam_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + '&exam_date=' + CAST(@exam_date AS VARCHAR(20)) + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
12	TR_Grades_FORINSERTED: Audit new grade record insertions.	CREATE TRIGGER TR_Grades_FORINSERTED ON Grades AFTER INSERT AS BEGIN DECLARE @grade_id INT; DECLARE @student_id INT; DECLARE @subject_id INT; DECLARE @exam_id INT; DECLARE @grade CHAR(1); SELECT @grade_id = grade_id, @student_id = student_id, @subject_id = subject_id, @exam_id = exam_id, @grade = grade FROM inserted;

		INSERT INTO GradesAudit VALUES ('New grade record with id=' + CAST(@grade_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + '&exam_id=' + CAST(@exam_id AS VARCHAR(10)) + '&grade=' + @grade + ' is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
13	Trigger_for_inserting_into_the_AttendanceAudit_table: Audit new attendance record insertions (specifically for Attendance table).	CREATE TRIGGER Trigger_for_inserting_into_the_AttendanceAudit_table ON Attendance AFTER INSERT AS BEGIN INSERT INTO AttendanceAudit VALUES ('New attendance record is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
14	Fees_FORINSERTED: Audit new fee record insertions.	CREATE TRIGGER Fees_FORINSERTED ON Fees AFTER INSERT AS BEGIN INSERT INTO FeesAudit VALUES ('New fee record is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
15	FeeDefaulter_FORINSERTED: Audit new fee defaulter record insertions.	CREATE TRIGGER FeeDefaulter_FORINSERTED ON FeeDefaulter AFTER INSERT AS BEGIN INSERT INTO FeeDefaulterAudit VALUES ('New fee defaulter record is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
16	Salary_FORINSERTED: Audit new salary record insertions.	CREATE TRIGGER Salary_FORINSERTED ON Salary AFTER INSERT AS BEGIN INSERT INTO SalaryAudit

		VALUES ('New salary record is added at ' + CAST(GETDATE() AS VARCHAR(20))); END;
17	Exams_FORINSERTED: Audit new exam record insertions.	CREATE TRIGGER Exams_FORINSERTED ON Exams AFTER INSERT AS BEGIN INSERT INTO ExamsAudit VALUES ('New exam scheduled at ' + CAST(GETDATE() AS VARCHAR(20))); END;
18	Grades_FORINSERTED: Audit new grade record insertions.	CREATE TRIGGER Grades_FORINSERTED ON Grades AFTER INSERT AS BEGIN INSERT INTO GradesAudit VALUES ('New grade recorded at ' + CAST(GETDATE() AS VARCHAR(20))); END;
19	Departments_FORINSERTED: Audit new department insertions (generic).	CREATE TRIGGER Departments_FORINSERTED ON Departments AFTER INSERT AS BEGIN INSERT INTO DepartmentAudit VALUES ('New department created at ' + CAST(GETDATE() AS VARCHAR(20))); END;
20	Teachers_FORINSERTED: Audit new teacher insertions.	CREATE TRIGGER Teachers_FORINSERTED ON Teachers AFTER INSERT AS BEGIN INSERT INTO TeacherAudit VALUES ('New teacher added at ' + CAST(GETDATE() AS VARCHAR(20))); END;

25. DML Triggers update – 20 Queries

1	TR_Students_FORupdate: Audit updates to student records.		CREATE TRIGGER TR_Students_FORupdate ON Students AFTER UPDATE AS BEGIN DECLARE @student_id INT; DECLARE @first_name VARCHAR(100);

			SELECT @student_id = student_id, @first_name = first_name FROM inserted; INSERT INTO StudentAudit VALUES ('Updated student with id=' + CAST(@student_id AS VARCHAR(10)) + '&name=' + @first_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
2	TR_Subjects_FORupdate: Audit updates to subject records.		CREATE TRIGGER TR_Subjects_FORupdate ON Subjects AFTER UPDATE AS BEGIN DECLARE @subject_id INT; DECLARE @subject_name VARCHAR(100); SELECT @subject_id = subject_id, @subject_name = subject_name FROM inserted; INSERT INTO SubjectAudit VALUES ('Updated subject with id=' + CAST(@subject_id AS VARCHAR(10)) + '&name=' + @subject_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
3	TR_Departments_FORupdate: Audit updates to department records.		CREATE TRIGGER TR_Departments_FORupdate ON Departments AFTER UPDATE AS BEGIN DECLARE @department_id INT; DECLARE @department_name VARCHAR(100); SELECT @department_id = department_id, @department_name = department_name FROM inserted; INSERT INTO DepartmentAudit VALUES ('Updated department with id=' + CAST(@department_id AS VARCHAR(10)) + '&name=' + @department_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
4	TR_StudentDepartments_FORupdate: Audit updates to student department records..		CREATE TRIGGER TR_StudentDepartments_FORupdate ON SDepartments AFTER UPDATE AS BEGIN

			DECLARE @sdepartment_id INT; DECLARE @sdepartment_name VARCHAR(100); SELECT @sdepartment_id = sdepartment_id, @sdepartment_name = sdepartment_name FROM inserted; INSERT INTO StudentDepartmentAudit VALUES ('Updated student department with id=' + CAST(@sdepartment_id AS VARCHAR(10)) + '&name=' + @sdepartment_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
5	TR_Teachers_FORupdate: Audit updates to teacher records		CREATE TRIGGER TR_Teachers_FORupdate ON Teachers AFTER UPDATE AS BEGIN DECLARE @teacher_id INT; DECLARE @first_name VARCHAR(100); SELECT @teacher_id = teacher_id, @first_name = first_name FROM inserted; INSERT INTO TeacherAudit VALUES ('Updated teacher with id=' + CAST(@teacher_id AS VARCHAR(10)) + '&name=' + @first_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
6	TR_Enrollments_FORupdate: Audit updates to enrollment records.		CREATE TRIGGER TR_Enrollments_FORupdate ON Enrollments AFTER UPDATE AS BEGIN DECLARE @enrollment_id INT; DECLARE @student_id INT; SELECT @enrollment_id = enrollment_id, @student_id = student_id FROM inserted; INSERT INTO EnrollmentAudit VALUES ('Updated enrollment with id=' + CAST(@enrollment_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;

7	TR_Attendance_FORupdate: Audit updates to attendance records..		CREATE TRIGGER TR_Attendance_FORupdate ON Attendance AFTER UPDATE AS BEGIN DECLARE @attendance_id INT; DECLARE @student_id INT; DECLARE @subject_id INT; SELECT @attendance_id = attendance_id, @student_id = student_id, @subject_id = subject_id FROM inserted; INSERT INTO AttendanceAudit VALUES ('Updated attendance record with id=' + CAST(@attendance_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
8	TR_Fees_FORupdate: Audit updates to fee records		CREATE TRIGGER TR_Fees_FORupdate ON Fees AFTER UPDATE AS BEGIN DECLARE @fee_id INT; DECLARE @student_id INT; DECLARE @sdepartment_id INT; SELECT @fee_id = fee_id, @student_id = student_id, @sdepartment_id = sdepartment_id FROM inserted; INSERT INTO FeesAudit VALUES ('Updated fee record with id=' + CAST(@fee_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&sdepartment_id=' + CAST(@sdepartment_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
9	TR_FeeDefaulter_FORupdate: Audit updates to fee defaulter records.		CREATE TRIGGER TR_FeeDefaulter_FORupdate ON FeeDefaulter AFTER UPDATE AS BEGIN DECLARE @feedef_id INT; DECLARE @student_id INT; DECLARE @sdepartment_id INT;

			SELECT @feedef_id = feedef_id, @student_id = student_id, @sdepartment_id = sdepartment_id FROM inserted; INSERT INTO FeeDefaulterAudit VALUES ('Updated fee defaulter record with id=' + CAST(@feedef_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&sdepartment_id=' + CAST(@sdepartment_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
10	TR_Salary_FORupdate: Audit updates to salary records.		CREATE TRIGGER TR_Salary_FORupdate ON Salary AFTER UPDATE AS BEGIN DECLARE @salary_id INT; DECLARE @teacher_id INT; DECLARE @department_id INT; SELECT @salary_id = salary_id, @teacher_id = teacher_id, @department_id = department_id FROM inserted; INSERT INTO SalaryAudit VALUES ('Updated salary record with id=' + CAST(@salary_id AS VARCHAR(10)) + '&teacher_id=' + CAST(@teacher_id AS VARCHAR(10)) + '&department_id=' + CAST(@department_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
11	TR_Exams_FORupdate: Audit updates to exam records.		CREATE TRIGGER TR_Exams_FORupdate ON Exams AFTER UPDATE AS BEGIN DECLARE @exam_id INT; DECLARE @subject_id INT; DECLARE @exam_date DATE; SELECT @exam_id = exam_id, @subject_id = subject_id, @exam_date = exam_date FROM inserted; INSERT INTO ExamsAudit

			VALUES ('Updated exam record with id=' + CAST(@exam_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + '&exam_date=' + CAST(@exam_date AS VARCHAR(20)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
12	TR_Grades_FORupdate: Audit updates to grade records.		CREATE TRIGGER TR_Grades_FORupdate ON Grades AFTER UPDATE AS BEGIN DECLARE @grade_id INT; DECLARE @student_id INT; DECLARE @subject_id INT; DECLARE @exam_id INT; DECLARE @grade CHAR(1); SELECT @grade_id = grade_id, @student_id = student_id, @subject_id = subject_id, @exam_id = exam_id, @grade = grade FROM inserted; INSERT INTO GradesAudit VALUES ('Updated grade record with id=' + CAST(@grade_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + '&exam_id=' + CAST(@exam_id AS VARCHAR(10)) + '&grade=' + @grade + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
13	Trigger_for_update: Audit updates to attendance records.		CREATE TRIGGER Trigger_for_update ON Attendance AFTER update AS BEGIN INSERT INTO AttendanceAudit VALUES ('Attendance record updated: ' + CAST(GETDATE() AS VARCHAR(20))); END;
14	Fees_FORupdate: Audit updates to fee records.		CREATE TRIGGER Fees_FORupdate ON Fees AFTER update

			AS BEGIN INSERT INTO FeesAudit VALUES ('Fee record updated: ' + CAST(GETDATE() AS VARCHAR(20))); END;
15	FeeDefaulter_FORupdate: Audit updates to fee defaulter records.		CREATE TRIGGER FeeDefaulter_FORupdate ON FeeDefaulter AFTER update AS BEGIN INSERT INTO FeeDefaulterAudit VALUES ('fee defaulter record updated ' + CAST(GETDATE() AS VARCHAR(20))); END;
16	Salary_FORupdate: Audit updates to salary records.		CREATE TRIGGER Salary_FORupdate ON Salary AFTER update AS BEGIN INSERT INTO SalaryAudit VALUES (' salary record updated ' + CAST(GETDATE() AS VARCHAR(20))); END;
17	Exams_FORupdate: Audit updates to exam records.		CREATE TRIGGER Exams_FORupdate ON Exams AFTER update AS BEGIN INSERT INTO ExamsAudit VALUES ('updated exam scheduled at ' + CAST(GETDATE() AS VARCHAR(20))); END;
18	Grades_FORupdate: Audit updates to grade records.		CREATE TRIGGER Grades_FORupdate ON Grades AFTER update AS BEGIN INSERT INTO GradesAudit VALUES ('updated grade recorded at ' + CAST(GETDATE() AS VARCHAR(20))); END;
19	Departments_FORupdated: Audit updates to department records.		CREATE TRIGGER Departments_FORupdated ON Departments AFTER update AS BEGIN INSERT INTO DepartmentAudit

			VALUES ('Department record updated: ' + CAST(GETDATE() AS VARCHAR(20))); END;
20	Teachers_FORupdate: Audit updates to teacher records.		CREATE TRIGGER Teachers_FORupdate ON Teachers AFTER update AS BEGIN INSERT INTO TeacherAudit VALUES (' teacher record updated ' + CAST(GETDATE() AS VARCHAR(20))); END;

26. delete trigger– 20 Queries

1	TR_Students_FORDELETE: Audit student deletions.	CREATE TRIGGER Students_fORdelete ON Students AFTER DELETE AS BEGIN DECLARE @student_id INT; DECLARE @first_name VARCHAR(100); SELECT @student_id = student_id, @first_name = first_name FROM deleted; INSERT INTO StudentAudit VALUES ('Deleted student with id=' + CAST(@student_id AS VARCHAR(10)) + '&name=' + @first_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
2	TR_Subjects_FORDELETE: Audit subject deletions.	-- TR_Subjects_FORDELETE CREATE TRIGGER TR_Subjects_FORDELETE ON Subjects AFTER DELETE AS BEGIN DECLARE @subject_id INT; DECLARE @subject_name VARCHAR(100); SELECT @subject_id = subject_id, @subject_name = subject_name FROM deleted; INSERT INTO SubjectAudit VALUES (

		'Deleted subject with id=' + CAST(@subject_id AS VARCHAR(10)) + '&name=' + @subject_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
3	TR_Departments_FORDELETE: Audit department deletions..	-- TR_Departments_FORDELETE CREATE TRIGGER TR_Departments_FORDELETE ON Departments AFTER DELETE AS BEGIN DECLARE @department_id INT; DECLARE @department_name VARCHAR(100); SELECT @department_id = department_id, @department_name = department_name FROM deleted; INSERT INTO DepartmentAudit VALUES ('Deleted department with id=' + CAST(@department_id AS VARCHAR(10)) + '&name=' + @department_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
4	TR_StudentDepartments_FORDELETE: Audit student department deletions	-- TR_StudentDepartments_FORDELETE CREATE TRIGGER TR_StudentDepartments_FORDELETE ON SDepartments AFTER DELETE AS BEGIN DECLARE @sdepartment_id INT; DECLARE @sdepartment_name VARCHAR(100); SELECT @sdepartment_id = sdepartment_id, @sdepartment_name = sdepartment_name FROM deleted; INSERT INTO StudentDepartmentAudit VALUES ('Deleted student department with id=' + CAST(@sdepartment_id AS VARCHAR(10)) + '&name=' + @sdepartment_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
5	TR_Teachers_FORDELETE: Audit teacher deletions.	-- TR_Teachers_FORDELETE CREATE TRIGGER TR_Teachers_FORDELETE ON Teachers AFTER DELETE

		AS BEGIN DECLARE @teacher_id INT; DECLARE @first_name VARCHAR(100); SELECT @teacher_id = teacher_id, @first_name = first_name FROM deleted; INSERT INTO TeacherAudit VALUES ('Deleted teacher with id=' + CAST(@teacher_id AS VARCHAR(10)) + '&name=' + @first_name + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
6	TR_Enrollments_FORDELETE: Audit enrollment deletions.	-- TR_Enrollments_FORDELETE CREATE TRIGGER TR_Enrollments_FORDELETE ON Enrollments AFTER DELETE AS BEGIN DECLARE @enrollment_id INT; DECLARE @student_id INT; SELECT @enrollment_id = enrollment_id, @student_id = student_id FROM deleted; INSERT INTO EnrollmentAudit VALUES ('Deleted enrollment with id=' + CAST(@enrollment_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
7	TR_Attendance_FORDELETE: Audit attendance record deletions.	-- TR_Attendance_FORDELETE CREATE TRIGGER TR_Attendance_FORDELETE ON Attendance AFTER DELETE AS BEGIN DECLARE @attendance_id INT; DECLARE @student_id INT; DECLARE @subject_id INT; SELECT @attendance_id = attendance_id, @student_id = student_id, @subject_id = subject_id FROM deleted;

		INSERT INTO AttendanceAudit VALUES ('Deleted attendance record with id=' + CAST(@attendance_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
8	TR_Fees_FORDELETE: Audit fee record deletions.	-- TR_Fees_FORDELETE CREATE TRIGGER TR_Fees_FORDELETE ON Fees AFTER DELETE AS BEGIN DECLARE @fee_id INT; DECLARE @student_id INT; DECLARE @sdepartment_id INT; SELECT @fee_id = fee_id, @student_id = student_id, @sdepartment_id = sdepartment_id FROM deleted; INSERT INTO FeesAudit VALUES ('Deleted fee record with id=' + CAST(@fee_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&sdepartment_id=' + CAST(@sdepartment_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
9	TR_FeeDefaulter_FORDELETE: Audit fee defaulter record deletions.	-- TR_FeeDefaulter_FORDELETE CREATE TRIGGER TR_FeeDefaulter_FORDELETE ON FeeDefaulter AFTER DELETE AS BEGIN DECLARE @feedef_id INT; DECLARE @student_id INT; DECLARE @sdepartment_id INT; SELECT @feedef_id = feedef_id, @student_id = student_id, @sdepartment_id = sdepartment_id FROM deleted; INSERT INTO FeeDefaulterAudit VALUES (

		'Deleted fee defaulter record with id=' + CAST(@feedef_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&sdepartment_id=' + CAST(@sdepartment_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
10	TR_Salary_FORDELETE: Audit salary record deletions.	-- TR_Salary_FORDELETE CREATE TRIGGER TR_Salary_FORDELETE ON Salary AFTER DELETE AS BEGIN DECLARE @salary_id INT; DECLARE @teacher_id INT; DECLARE @department_id INT; SELECT @salary_id = salary_id, @teacher_id = teacher_id, @department_id = department_id FROM deleted; INSERT INTO SalaryAudit VALUES ('Deleted salary record with id=' + CAST(@salary_id AS VARCHAR(10)) + '&teacher_id=' + CAST(@teacher_id AS VARCHAR(10)) + '&department_id=' + CAST(@department_id AS VARCHAR(10)) + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;
11	TR_Exams_FORDELETE: Audit exam record deletions.	-- TR_Exams_FORDELETE CREATE TRIGGER TR_Exams_FORDELETE ON Exams AFTER DELETE AS BEGIN DECLARE @exam_id INT; DECLARE @subject_id INT; DECLARE @exam_date DATE; SELECT @exam_id = exam_id, @subject_id = subject_id, @exam_date = exam_date FROM deleted; INSERT INTO ExamsAudit VALUES ('Deleted exam record with id=' + CAST(@exam_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + '&exam_date=' + CAST(@exam_date AS VARCHAR(20)) +

		``' at ' + CAST(GETDATE() AS VARCHAR(20))); END;```
12	TR_Grades_FORDELETE: Audit grade record deletions.	```-- TR_Grades_FORDELETE CREATE TRIGGER TR_Grades_FORDELETE ON Grades AFTER DELETE AS BEGIN DECLARE @grade_id INT; DECLARE @student_id INT; DECLARE @subject_id INT; DECLARE @exam_id INT; DECLARE @grade CHAR(1); SELECT @grade_id = grade_id, @student_id = student_id, @subject_id = subject_id, @exam_id = exam_id, @grade = grade FROM deleted; INSERT INTO GradesAudit VALUES ('Deleted grade record with id=' + CAST(@grade_id AS VARCHAR(10)) + '&student_id=' + CAST(@student_id AS VARCHAR(10)) + '&subject_id=' + CAST(@subject_id AS VARCHAR(10)) + '&exam_id=' + CAST(@exam_id AS VARCHAR(10)) + '&grade=' + @grade + ' at ' + CAST(GETDATE() AS VARCHAR(20))); END;```
13	Trigger_for_del: Audit attendance record deletions.	```CREATE TRIGGER Trigger_for_del ON Attendance AFTER delete AS BEGIN INSERT INTO AttendanceAudit VALUES ('Deleted attendance record at ' + CAST(GETDATE() AS VARCHAR(20))); END;```
14	FeesFORdelete: Audit fee record deletions.	```CREATE TRIGGER FeesFORdelete ON Fees AFTER delete AS BEGIN INSERT INTO FeesAudit VALUES ('deleteed fee record is ' + CAST(GETDATE() AS VARCHAR(20))); END;```

15	FeeDefaulter_FORdelete: Audit fee defaulter record deletions.	CREATE TRIGGER FeeDefaulter_FORdelete ON FeeDefaulter AFTER delete AS BEGIN INSERT INTO FeeDefaulterAudit VALUES ('deleted fee defaulter record is ' + CAST(GETDATE() AS VARCHAR(20))); END;
16	Salary_FORdelete: Audit salary record deletions.	CREATE TRIGGER Salary_FORdelete ON Salary AFTER delete AS BEGIN INSERT INTO SalaryAudit VALUES ('deleated salary is ' + CAST(GETDATE() AS VARCHAR(20))); END;
17	Exams_FORdelete: Audit exam record deletions.	CREATE TRIGGER Exams_FORdelete ON Exams AFTER delete AS BEGIN INSERT INTO ExamsAudit VALUES ('deleated exam scheduled is ' + CAST(GETDATE() AS VARCHAR(20))); END;
18	Grades_FORdelete: Audit grade record deletions.	CREATE TRIGGER Grades_FORdeleate ON Grades AFTER delete AS BEGIN INSERT INTO GradesAudit VALUES ('delete grade recorded is ' + CAST(GETDATE() AS VARCHAR(20))); END;
19	Departments_FORdelete: Audit department deletions.	CREATE TRIGGER Departments_FORdelete ON Departments AFTER delete AS BEGIN INSERT INTO DepartmentAudit VALUES ('delete department is ' + CAST(GETDATE() AS VARCHAR(20))); END;
20	Teachers_FORdelete: Audit teacher deletions.	CREATE TRIGGER Teachers_FORdelete ON Teachers AFTER delete AS BEGIN INSERT INTO TeacherAudit

		VALUES (' delete teacher is' + CAST(GETDATE() AS VARCHAR(20))); END;

29. Single-Row Functions UPPER, LOWER, LENGTH, SUBSTR using logical operators– 50 Queries

1	Retrieve lowercased first names of specific student.	select LOWER(first_name) from Students where student_id=3 and sdepartment_id=1
2	Retrieve lowercased first names of male student.	SELECT LOWER(first_name) FROM Students WHERE gender = 'male'and student_id=2;
3	Retrieve lowercased last names of female students	SELECT LOWER(last_name) FROM Students WHERE gender = 'female' AND sdepartment_id = 1;
4	Retrieve lowercased department names related to science and technology.	SELECT LOWER(department_name) FROM departments WHERE department_name LIKE '%science%' OR department_name LIKE '%technology%';
5	Retrieve lowercased subject names related to math with credit hours > 3	SELECT LOWER(subject_name) FROM Subjects WHERE subject_name LIKE '%math%' AND credithour > 3;
6	Retrieve lowercased first names of male teachers in specific department or subject.	SELECT LOWER(first_name) FROM Teachers WHERE gender = 'male' AND (department_id = 1 OR subject_id = 2);
7	Retrieve lowercased addresses of	SELECT LOWER(Aaddress) FROM Students WHERE gender = 'female' AND (contact_number = '123456789' OR email = 'example@example.com');

	female students with specific contact or email.	
8	Retrieve lowercased emails of male teachers in specific department or subject.	`SELECT LOWER(email) FROM Teachers WHERE gender = 'male' AND (department_id = 1 OR subject_id = 2);`
9	Retrieve lowercased subdepartment names related to engineering.	`SELECT LOWER(sdepartment_name) FROM sdepartments WHERE sdepartment_name LIKE '%engineering%' AND sdepartment_id = 1;`
10	Retrieve lowercased contact numbers of male students in specific department or email.	`SELECT LOWER(contact_number) FROM Students WHERE gender = 'male' AND (sdepartment_id = 1 OR email = 'example@example.com');`
11	Retrieve lowercased emails of female students with specific department and contact number.	`SELECT LOWER(email) FROM Students WHERE gender = 'female' AND sdepartment_id = 1 AND contact_number LIKE '%123%';`
12	. Retrieve uppercased first name of male student.	`SELECT UPPER(first_name) FROM Students WHERE gender = 'male' and student_id=2;`
13	Retrieve uppercased last names of female students.	`SELECT UPPER(last_name) FROM Students WHERE gender = 'female' AND sdepartment_id = 1;`
14	Retrieve uppercased department names related to science and technology.	`SELECT UPPER(department_name) FROM departments WHERE department_name LIKE '%science%' OR department_name LIKE '%technology%';`

15	Retrieve uppercased subject names related to math with credit hours > 3.	`SELECT UPPER(subject_name) FROM Subjects WHERE subject_name LIKE '%math%' AND credithour > 3;`
16	Retrieve uppercased first names of male teachers in specific department or subject.	`SELECT UPPER(first_name) FROM Teachers WHERE gender = 'male' AND (department_id = 1 OR subject_id = 2);`
17	Retrieve uppercased addresses of female students with specific contact or email	`SELECT UPPER(Aaddress) FROM Students WHERE gender = 'female' AND (contact_number = '123456789' OR email = 'example@example.com');`
18	Retrieve uppercased emails of male teachers in specific department or subject.	`SELECT UPPER(email) FROM Teachers WHERE gender = 'male' AND (department_id = 1 OR subject_id = 2);`
19	Retrieve uppercased subdepartment names related to engineering.	`SELECT UPPER(sdepartment_name) FROM sdepartments WHERE sdepartment_name LIKE '%engineering%' AND sdepartment_id = 1;`
20	Retrieve uppercased contact numbers of male students in specific department or email.	`SELECT UPPER(contact_number) FROM Students WHERE gender = 'male' AND (sdepartment_id = 1 OR email = 'example@example.com');`
21	. Retrieve uppercased emails of female students with specific department	`SELECT UPPER(email) FROM Students WHERE gender = 'female' AND sdepartment_id = 1 AND contact_number LIKE '%123%';`

	and contact number.	
22	Retrieve length of first name for specific student or department.	`select len(first_name) from Students where student_id=1 or sdepartment_id=1`
23	Retrieve length of department name for specific department IDs.	`SELECT LEN(department_name) FROM departments WHERE department_id = 1 OR department_id = 2;`
24	Retrieve length of subdepartment name for specific subdepartment ID.	`SELECT LEN(sdepartment_name) FROM sdepartments WHERE sdepartment_id = 1 AND sdepartment_id = 2;`
25	. Retrieve length of subject name for specific subject IDs.	`SELECT LEN(subject_name) FROM Subjects WHERE subject_id = 1 OR subject_id = 2;`
26	Retrieve length of first name for specific teacher and department.	`SELECT LEN(first_name) FROM Teachers WHERE teacher_id = 1 AND department_id = 1;`
27	Retrieve length of last name for specific department IDs	`SELECT LEN(last_name) FROM Teachers WHERE department_id = 1 OR department_id = 2;`
28	Retrieve length of address for specific student and subdepartment.	`SELECT LEN(Aaddress) FROM Students WHERE student_id = 1 AND sdepartment_id = 1;`
29	Retrieve length of contact number for specific students.	`SELECT LEN(contact_number) FROM Students WHERE student_id = 1 OR student_id = 2;`

30	Retrieve length of email for specific student and subdepartment.	`SELECT LEN(email) FROM Students WHERE student_id = 1 AND sdepartment_id = 1;`
31	Retrieve length of department name for specific department IDs.	`SELECT LEN(department_name) FROM departments WHERE department_id = 1 AND department_id = 2;`
32	Retrieve length of subdepartment name for specific subdepartment IDs.	`SELECT LEN(sdepartment_name) FROM sdepartments WHERE sdepartment_id = 1 OR sdepartment_id = 2;`
33	Retrieve length of subdepartment name for specific subdepartment or "Math".	`SELECT LEN(sdepartment_name) FROM sdepartments WHERE sdepartment_id = 1 OR sdepartment_name ='Math';`
34	Retrieve length of subject name for specific subject and credit hour.	`SELECT LEN(subject_name) FROM Subjects WHERE subject_id = 1 AND credithour < 4;`
35	Retrieve length of last name for specific department and gender.	`SELECT LEN(last_name) FROM Teachers WHERE department_id = 1 AND gender = 'Female';`
36	Retrieve length of department name for specific department IDs or department ID.	`SELECT LEN(department_name) FROM departments WHERE (department_id = 1 AND department_name = 'Mathematics') OR department_id = 3;`
37	Retrieve length of email for specific student,	`SELECT LEN(email) FROM Students WHERE (student_id = 1 AND sdepartment_id = 2) OR (gender = 'Female' AND sdepartment_id = 3);`

	subdepartment, or gender.	
38	Retrieve length of contact number for specific student and gender.	`SELECT LEN(contact_number) FROM Students WHERE student_id = 1 AND (gender = 'Male' OR sdepartment_id = 2);`
39	Retrieve length of address for specific student, subdepartment, or gender.	`SELECT LEN(Aaddress) FROM Students WHERE (student_id = 1 AND sdepartment_id = 2) OR gender = 'Female';`
40	Retrieve length of last name for specific department IDs and gender.	`SELECT LEN(last_name) FROM Teachers WHERE (department_id = 1 OR department_id = 3) AND gender = 'Female';`
41	Retrieve length of first name for specific student, subdepartment, or gender.	`SELECT LEN(first_name) FROM Students WHERE student_id = 1 AND (sdepartment_id = 2 OR gender = 'Male');`
42	Retrieve substring of first name for specific student or department.	`SELECT SUBSTRING(first_name, 1, 2) AS ExtractString FROM Students WHERE student_id = 1 OR sdepartment_id = 1;`
43	Retrieve substring of first name for specific student and subdepartment.	`SELECT SUBSTRING(first_name, 1, 3) AS ExtractString FROM Students WHERE student_id = 2 AND sdepartment_id = 2;`
44	Retrieve substring of first name for specific student or subdepartment.	`SELECT SUBSTRING(first_name, 1, 4) AS ExtractString FROM Students WHERE student_id = 3 OR sdepartment_id = 3;`
45	Retrieve substring of first name for specific student and subdepartment.	`SELECT SUBSTRING(first_name, 2, 4) AS ExtractString FROM Students WHERE student_id = 4 AND sdepartment_id = 4;`

46	Retrieve substring of first name for specific student or subdepartment.	`SELECT SUBSTRING(first_name, LEN(first_name) - 2, LEN(first_name)) AS ExtractString FROM Students WHERE student_id = 5 OR sdepartment_id = 5;`
47	Retrieve substring of first name for specific student and subdepartment.	`SELECT SUBSTRING(first_name, 3, 2) AS ExtractString FROM Students WHERE student_id = 6 AND sdepartment_id = 6;`
48	Retrieve substring of first name for specific student and subdepartment.	`SELECT SUBSTRING(first_name, 2, 3) AS ExtractString FROM Students WHERE student_id = 10 AND sdepartment_id = 10;`
49	Retrieve substring of contact number for specific student and subdepartment.	`SELECT SUBSTRING(contact_number, 1, 3) AS ExtractString FROM Students WHERE student_id = 3 AND sdepartment_id = 1;`
50	Retrieve substring of email for specific subdepartment.	`SELECT SUBSTRING(email, 1, 1) AS ExtractString FROM Students WHERE sdepartment_id = 3;`

Transaction COMMIT and ROLLBACK

30. Single-Row Functions TRIM, REPLACE, ROUND, TRUNC using logical operators **50 Queries**

1	Students: Retrieve the trimmed version of the Aaddress field.	`SELECT TRIM(Aaddress) AS trimmed_Aaddress FROM Students;`
2	Fees: Convert the amount field to decimal with two decimal places.	`SELECT CAST(amount AS decimal(10, 2)) AS truncated_amount FROM Fees;`
3	Students: Retrieve the trimmed versions of the first_name and last_name fields.	`SELECT TRIM(first_name) AS trimmed_first_name, TRIM(last_name) AS trimmed_last_name FROM Students;`
4	Students: Replace 'Mr.' with 'Ms.' in the first_name field.	`SELECT REPLACE(first_name, 'Mr.', 'Ms.') AS replaced_first_name FROM Students;`

5	Students: Retrieve the first_name, last_name, and truncated version of the date_of_birth field.	`SELECT first_name, last_name, CAST(date_of_birth AS DATE) AS truncated_date_of_birth FROM Students;`
6	Departments: Update the department_name field, replacing 'Engineering' with 'Computer Science'.	`UPDATE departments SET department_name = REPLACE(department_name, 'Engineering', 'Computer Science') WHERE department_name = 'Engineering';`
7	Fees: Retrieve the student_id and round the amount field to two decimal places.	`SELECT student_id, ROUND(amount,2) AS rounded_amount FROM Fees;`
8	Fees: Retrieve the student_id and truncated version of the payment_date field.	`SELECT student_id, CAST(payment_date AS DATE) AS truncated_payment_date FROM Fees;`
9	Students: Update the Aaddress field, replacing 'USA' with 'United States'.	`UPDATE Students SET Aaddress = REPLACE(Aaddress, 'USA', 'United States');`
10	Subjects: Retrieve the rounded average credithour.	`SELECT ROUND(AVG(credithour), 2) AS rounded_average_credit_hour FROM Subjects;`
11	Departments: Update the department_name field from 'Management' to 'Business Administration'.	`UPDATE departments SET department_name = 'Business Administration' WHERE department_name = 'Management';`
12	FeeDefaulter: Retrieve the student_id and round the amountdue field to the nearest tenth.	`SELECT student_id, ROUND(amountdue, -1) AS rounded_amountdue FROM FeeDefaulter;`
13	Exams: Retrieve the subject_id and truncated version of the year from the exam_date field.	`SELECT subject_id, CAST(DATEPART(YEAR, exam_date) AS varchar(max)) AS truncated_exam_year FROM exams;`
14	Teachers: Update the last_name field to 'Doe' for teachers with first names containing 'A'.	`UPDATE Teachers SET last_name = 'Doe' WHERE first_name LIKE '%A%';`
15	Subjects: Update the subject_name field, replacing 'Physics' with 'Chemistry' for subjects named 'Physics'.	`UPDATE Subjects SET subject_name = REPLACE(subject_name, 'Physics', 'Chemistry') WHERE subject_name = 'Physics';`
16	Subjects: Round the credithour field to two decimal places.	`SELECT ROUND(credithour, 2) AS rounded_credit FROM Subjects;`
17	Students: Retrieve the rounded average age of students.	`SELECT ROUND(AVG(DATEDIFF(year, date_of_birth, GETDATE())), 0) AS rounded_average_age FROM Students;`

		;
18	Students: Update the gender field, swapping 'Male' with 'Female' and vice versa.	UPDATE Students SET gender = CASE WHEN gender = 'Male' THEN 'Female' WHEN gender = 'Female' THEN 'Male' ELSE gender END;
19	Departments: Retrieve the department_id and convert it to decimal with two decimal places.	SELECT department_id, CAST(department_id AS decimal(10, 2)) AS truncated_department_id FROM departments
20	Subjects: Update the subject_name field, changing 'Chemistry' to 'Biology'.	UPDATE Subjects SET subject_name = 'Biology' WHERE subject_name = 'Chemistry';
21	Fees: Retrieve the rounded average amount of fees	SELECT ROUND(AVG(amount), 2) AS rounded_average_amount FROM Fees;
22	. Fees: Retrieve the truncated payment month from the payment_date.	SELECT student_id, DATEFROMPARTS(YEAR(payment_date), MONTH(payment_date), 1) AS truncated_payment_month FROM Fees;
23	Teachers: Update the last_name field, changing 'D%' and '%s' to 'Johnson'	UPDATE Teachers SET last_name = 'Johnson' WHERE first_name LIKE 'D%' OR first_name LIKE '%s';
24	. Students: Update the Aaddress field, replacing '123' with '456'.	UPDATE Students SET Aaddress = REPLACE(Aaddress, '123', '456'); SELECT first_name, last_name,
25	Students: Retrieve the first_name, last_name, and truncated contact_number.	CAST(LEFT(contact_number, 3) AS VARCHAR) AS truncated_contact_number FROM Students;
26	Departments: Update the department_name field, changing 'Finance' to 'Accounting'	UPDATE departments SET department_name = 'Accounting' WHERE department_name = 'Finance';
27	. FeeDefaulter: Retrieve the student_id and rounded amountdue to the nearest hundred.	SELECT student_id, ROUND(amountdue, -2) AS rounded_amountdue FROM FeeDefaulter;
28	Exams: Retrieve the subject_id and truncated exam_month.	SELECT subject_id, CAST(CONCAT(YEAR(exam_date), '-', MONTH(exam_date), '-01') AS DATE) AS truncated_exam_month FROM exams;

29	Teachers: Update the first_name field, changing last_name containing 'E' to 'Emily'.	`UPDATE Teachers` `SET first_name = 'Emily'` `WHERE last_name LIKE '%E%';`
30	Subjects: Update the subject_name field, changing 'Chemistry' to 'Physics'.	`UPDATE Subjects` `SET subject_name = REPLACE(subject_name, 'Chemistry', 'Physics')` `WHERE subject_name = 'Chemistry';`
31	Students: Retrieve the rounded average age in multiples of 5.	`SELECT ROUND(AVG(DATEDIFF(year, date_of_birth, GETDATE())) / 5,5) * 5 AS rounded_average_age` `FROM Students;`
32	Subjects: Retrieve the rounded average credit hour in multiples of 5.	`SELECT ROUND(AVG(credithour) /2, 5) * 5 AS rounded_average_credit_hour` `FROM Subjects;`
33	Teachers: Update the gender field, swapping 'Male' and 'Female'.	`UPDATE Teachers` `SET gender = CASE` `WHEN gender = 'Male' THEN 'Female'` `WHEN gender = 'Female' THEN 'Male'` `ELSE gender` `END;`
34	Subjects: Retrieve the trimmed subject_name.	`SELECT TRIM(subject_name) AS trimmed_subject_name` `FROM Subjects;`
35	Students: Retrieve the student_id and updated_address by replacing 'Department' with 'Department of'.	`SELECT s.student_id, REPLACE(s.Aaddress, 'Department', 'Department of') AS updated_address` `FROM Students s` `JOIN departments d ON s.sdepartment_id = d.department_id;`
36	Subjects: Retrieve the trimmed and lowercased subject_name.	`SELECT TRIM(LOWER(subject_name)) AS trimmed_lower_subject_name` `FROM Subjects;`
37	Teachers: Retrieve the Teacher_id and cleaned_contact_number by removing dashes and spaces.	`SELECT Teacher_id,` `REPLACE(REPLACE(contact_number, '-', ''), ' ', '') AS cleaned_contact_number` `FROM Teachers;`
38	Students: Retrieve the student_id, first_name, last_name, and rounded_age.	`SELECT student_id, first_name, last_name,` `ROUND(DATEDIFF(YEAR, date_of_birth, GETDATE()), 0) AS rounded_age` `FROM Students;`
39	Subjects: Retrieve the subject_id, subject_name, and truncated_credit_hour.	`SELECT subject_id, subject_name,` `CAST(ROUND(credithour, 2) AS decimal(10, 2)) AS truncated_credit_hour` `FROM Subjects;`

40	Students: Retrieve the student_id and updated_email by replacing the first_name with '1'.	SELECT student_id, REPLACE(email,first_name ,1) AS updated_email FROM Students;
41	Subjects: Retrieve the subject_id and truncated_subject_name by taking the first 3 characters of subject_name	SELECT subject_id, CAST(LEFT(subject_name, 3) AS VARCHAR(3)) AS truncated_subject_name FROM Subjects;
42	. Students: Retrieve the student_id, first_name, last_name, and truncated_birth_month by extracting the month from date_of_birth	SELECT student_id, first_name, last_name, CAST(MONTH(date_of_birth) AS INT) AS truncated_birth_month FROM Students;
43	. Students: Retrieve the student_id and trimmed_uppercase_first_name by trimming and converting the first_name to uppercase.	SELECT student_id, UPPER(TRIM(first_name)) AS trimmed_uppercase_first_name FROM Students;
44	eachers and Salary: Retrieve the first_name, last_name, and rounded_salary_amount by rounding the amount to the nearest thousand.	SELECT first_name, last_name, ROUND(amount, -3) AS rounded_salary_amount FROM Teachers,salary;
45	Teachers: Retrieve all columns for teachers whose last_name contains 's' and the contact_number is not empty after trimming.	SELECT * FROM Teachers WHERE CHARINDEX('s', last_name) > 0 AND TRIM(contact_number) = contact_number;
46	Students: Retrieve the updated_last_name by replacing spaces with underscores and truncated_birth_date by flooring the birth year to the nearest decade.	SELECT REPLACE(last_name, ' ', '_') AS updated_last_name, CAST(FLOOR(YEAR(date_of_birth) / 10.0) * 10 AS INT) AS truncated_birth_date FROM Students;
47	Teachers: Retrieve the teacher_id and updated_first_name by replacing spaces with the concatenated department_id.	SELECT teacher_id, REPLACE(TRIM(first_name), ' ', CONCAT(' - ', department_id, ' - ')) AS updated_first_name FROM Teachers;
48	Students: Retrieve the updated_info by concatenating last_name, ' - ', and trimmed contact_number.	SELECT CONCAT(REPLACE(last_name, ' ', ' - '), ' - ', TRIM(contact_number)) AS updated_info FROM Students;

49	Subjects: Retrieve the updated_subject_name by replacing spaces with the concatenated Subject_id and rounded_credit_hour.	`SELECT REPLACE(subject_name, ' ', CONCAT(' - ', Subject_id, ' - ')) AS updated_subject_name, ROUND(credithour, 2) AS rounded_credit_hour FROM Subjects;`
50	Teachers and Salary: Retrieve the trimmed_first_name from Teachers and rounded_salary_amount from Salary.	`SELECT TRIM(teachers.first_name) AS trimmed_first_name, ROUND(salary.amount, -3) AS rounded_salary_amount FROM Teachers teachers, salary;`

31. Transaction COMMIT and ROLLBACK– 20 Queries

1	Insert a new student record	`Begin tran` `INSERT INTO Students (student_id, sdepartment_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email)` `VALUES (1, 1, 'John', 'Doe', '2000-01-01', 'Male', '123 Main St', '1234567890', 'john.doe@example.com');` `rollback` `commit`
2	Insert a new department record	`Begin tran` `INSERT INTO departments (department_id, department_name)` `VALUES (1, 'Science');` `rollback` `commit`
3	Update department	`Begin tran` `INSERT INTO departments (department_id, department_name)` `VALUES (1, 'Science');` `rollback` `commit`
4	Update a student's department:	`Begin tran` `UPDATE Students` `SET sdepartment_id = 2` `WHERE student_id = 1;` `rollback` `commit`
5	Delete a student record:	`Begin tran` `DELETE FROM Students` `WHERE student_id = 1;` `rollback` `commit`

6	Insert a new subject record:	Begin tran INSERT INTO Subjects (subject_id, subject_name, credithour) VALUES (1, 'Mathematics', 3); rollback commit
7	Insert a new teacher record:	Begin tran INSERT INTO Teachers (teacher_id, department_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email, subject_id) VALUES (1, 1, 'Jane', 'Smith', '1980-01-01', 'Female', '456 Elm St', '9876543210', 'jane.smith@example.com', 1); rollback commit
8	Update a teacher's department:	Begin tran UPDATE Teachers SET department_id = 2 WHERE teacher_id = 1; rollback commit
9	Delete a teacher record:	Begin tran DELETE FROM Teachers WHERE teacher_id = 1; rollback commit
10	Insert a new enrollment record	Begin tran INSERT INTO Enrollments (enrollment_id, student_id) VALUES (1, 1); rollback commit
11	Update a student's enrollment:	Begin tran UPDATE Enrollments SET student_id = 2 WHERE enrollment_id = 1; rollback commit
12	Delete an enrollment record	Begin tran DELETE FROM Enrollments WHERE enrollment_id = 1; rollback commit
13	Insert a new attendance record:	Begin tran

		INSERT INTO attendance (attendance_id, student_id, subject_id, date) VALUES (1, 1, 1, '2023-01-01'); rollback commit
14	Update a student's attendance	Begin tran UPDATE attendance SET student_id = 2 WHERE attendance_id = 1; rollback commit
15	Delete an attendance record	Begin tran DELETE FROM attendance WHERE attendance_id = 1; rollback commit
16	Insert a new fee record	Begin tran INSERT INTO fees (fee_id, student_id, sdepartment_id, amount, payment_date) VALUES (1, 1, 1, 1000, '2023-01-01'); rollback commit
17	Update a fee record:	Begin tran UPDATE fees SET amount = 1500 WHERE fee_id = 1; rollback commit
18	Insert a new grade record	Begin tran INSERT INTO grades (grade_id, student_id, subject_id, exam_id, grade) VALUES (1, 1, 1, 1, 'A'); rollback commit
19	Update a grade record	Begin tran UPDATE grades SET grade = 'B' WHERE grade_id = 1; rollback commit
20	Delete a fee record	Begin tran DELETE FROM fees WHERE fee_id = 1; rollback commit

32. Exception handling- Try Catch– 20 Queries

1	Try to insert a new student and handle any exceptions that may occur.	BEGIN TRY INSERT INTO Students (student_id, sdepartment_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email) VALUES (1, 1, 'John', 'Doe', '2000-01-01', 'Male', '123 Main St', '1234567890', 'john.doe@example.com'); END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to insert student.'; PRINT ERROR_MESSAGE(); END CATCH;
2	Try to update a student's department and handle any exceptions that may occur.	BEGIN TRY UPDATE Students SET sdepartment_id = 2 WHERE student_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update student department.'; PRINT ERROR_MESSAGE(); END CATCH;
3	Try to delete a student and handle any exceptions that may occur.	BEGIN TRY DELETE FROM Students WHERE student_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to delete student.'; PRINT ERROR_MESSAGE(); END CATCH;
4	Try to insert a new department and handle any exceptions that may occur.	BEGIN TRY INSERT INTO departments (department_id, department_name) VALUES (1, 'Science'); END TRY BEGIN CATCH -- Handle exception

		PRINT 'Error: Failed to insert department.'; PRINT ERROR_MESSAGE(); END CATCH;
5	Try to update a department's name and handle any exceptions that may occur.	BEGIN TRY UPDATE departments SET department_name = 'Mathematics' WHERE department_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update department name.'; PRINT ERROR_MESSAGE(); END CATCH;
6	Try to delete a department and handle any exceptions that may occur.	BEGIN TRY DELETE FROM departments WHERE department_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to delete department.'; PRINT ERROR_MESSAGE(); END CATCH;
7	Try to insert a new subject and handle any exceptions that may occur.	BEGIN TRY INSERT INTO Subjects (subject_id, subject_name, credithour) VALUES (1, 'Mathematics', 3); END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to insert subject.'; PRINT ERROR_MESSAGE(); END CATCH;
8	Try to update a subject's name and handle any exceptions that may occur.	BEGIN TRY UPDATE Subjects SET subject_name = 'Physics' WHERE subject_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update subject name.';

		PRINT ERROR_MESSAGE(); END CATCH;
9	Try to delete a subject and handle any exceptions that may occur.	BEGIN TRY DELETE FROM Subjects WHERE subject_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to delete subject.'; PRINT ERROR_MESSAGE(); END CATCH;
10	Try to insert a new teacher and handle any exceptions that may occur.	BEGIN TRY INSERT INTO Teachers (teacher_id, department_id, first_name, last_name, date_of_birth, gender, Aaddress, contact_number, email, subject_id) VALUES (1, 1, 'Jane', 'Smith', '1980-01-01', 'Female', '456 Elm St', '9876543210', 'jane.smith@example.com', 1); END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to insert teacher.'; PRINT ERROR_MESSAGE(); END CATCH;
11	Try to update a teacher's department and handle any exceptions that may occur.	BEGIN TRY UPDATE Teachers SET department_id = 2 WHERE teacher_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update teacher department.'; PRINT ERROR_MESSAGE(); END CATCH;
12	Try to delete a teacher and handle any exceptions that may occur.	BEGIN TRY DELETE FROM Teachers WHERE teacher_id = 1; END TRY BEGIN CATCH -- Handle exception

		PRINT 'Error: Failed to delete teacher.'; PRINT ERROR_MESSAGE(); END CATCH;
13	Try to enroll a student in a subject and handle any exceptions that may occur.	BEGIN TRY INSERT INTO Enrollments (enrollment_id, student_id) VALUES (1, 1); END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to enroll student in subject.'; PRINT ERROR_MESSAGE(); END CATCH;
14	Try to update a student's enrollment and handle any exceptions that may occur.	BEGIN TRY UPDATE Enrollments SET student_id = 2 WHERE enrollment_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update enrollment.'; PRINT ERROR_MESSAGE(); END CATCH;
15	Try to delete a student's enrollment and handle any exceptions that may occur.	BEGIN TRY DELETE FROM Enrollments WHERE enrollment_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to delete enrollment.'; PRINT ERROR_MESSAGE(); END CATCH;
16	Try to mark attendance for a student and handle any exceptions that may occur.	BEGIN TRY INSERT INTO attendance (attendance_id, student_id, subject_id, date) VALUES (1, 1, 1, '2023-01-01'); END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to mark attendance.'; PRINT ERROR_MESSAGE();

		END CATCH;
17	Try to update a student's attendance and handle any exceptions that may occur.	BEGIN TRY UPDATE attendance SET student_id = 2 WHERE attendance_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update attendance.'; PRINT ERROR_MESSAGE(); END CATCH;
18	Try to delete a student's attendance and handle any exceptions that may occur.	BEGIN TRY DELETE FROM attendance WHERE attendance_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to delete attendance.'; PRINT ERROR_MESSAGE(); END CATCH;
19	Try to insert a new fee record and handle any exceptions that may occur.	BEGIN TRY INSERT INTO fees (fee_id, student_id, sdepartment_id, amount, payment_date) VALUES (1, 1, 1, 1000, '2023-01-01'); END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to insert fee record.'; PRINT ERROR_MESSAGE(); END CATCH;
20	Try to update a fee record and handle any exceptions that may occur.	BEGIN TRY UPDATE fees SET amount = 1500 WHERE fee_id = 1; END TRY BEGIN CATCH -- Handle exception PRINT 'Error: Failed to update fee record.'; PRINT ERROR_MESSAGE(); END CATCH;