A logo of a university

Description automatically generated with low confidence

**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: 4ths

Section: D

Submitted to: Mr. Abdul Rehman

Date: 19-june-2023  
  
  
  
**Department of Computer Science**

**Lahore Garrison University**

**Introduction:**

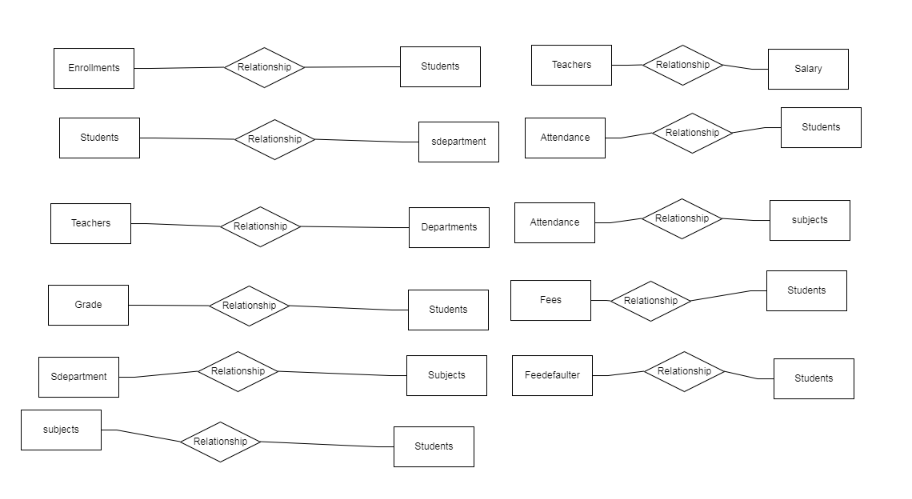
ABC School, a prestigious educational institution, School Management System to effectively handle various aspects of its operations. The system incorporates several tables to manage student information, attendance tracking, fee management, salary administration, exam scheduling, and grading. Students like John Doe, assigned a student ID, are enrolled in specific subjects such as Science, while Emily Smith. 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. 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, subjects, 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

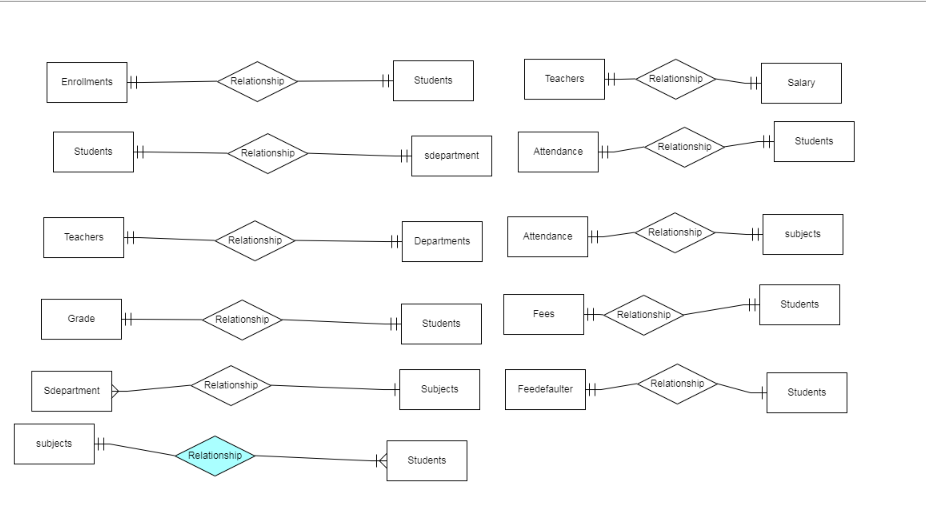
**Step 2- Relationship Identification**

* 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

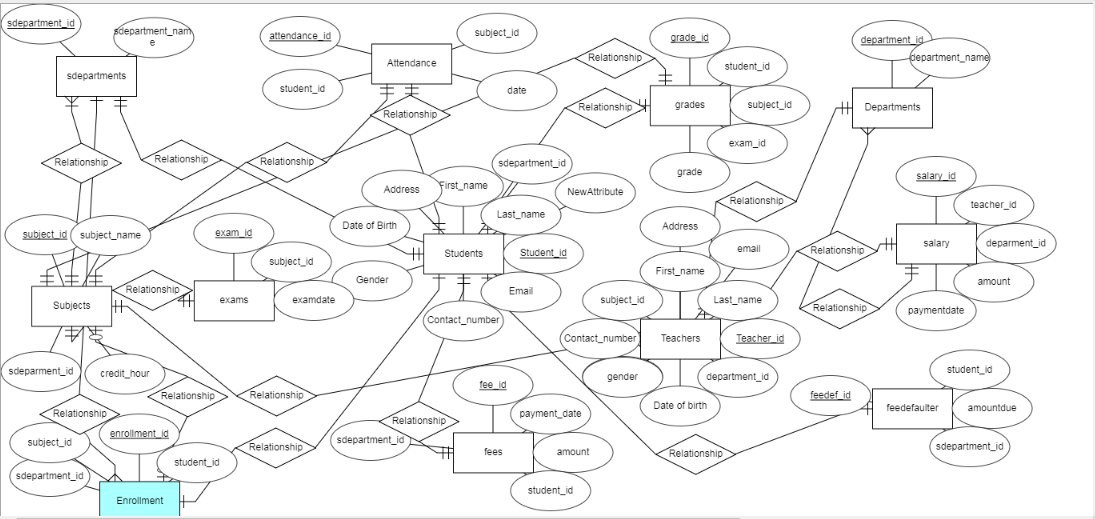
**Fees:** fee\_id, student\_id, sdepartment\_id, amount, payment\_date

**Feedefaulter:** feedef\_id, student\_id, sdepartment\_id, amountdue

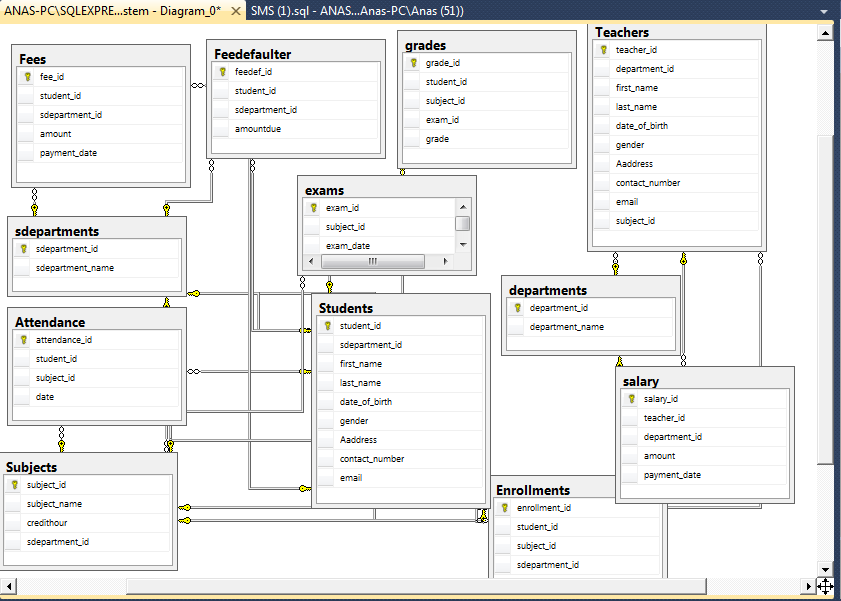
**Exams:** exam\_id, subject\_id, exam\_date

**Grades:** grade\_id, subject\_id, exam\_id, grade, student\_id

**Step 5- Create the ERD Diagram**

****

**Step 6- Convert ERD to Tables in DBMS**

****

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Topic** | **Queries** |
|  | CREATE TABLE Statement | **10** |
|  | PRIMARY KEY and FOREIGN KEY | **10** |
|  | AUTO INCREMENT | **10** |
|  | ALTER TABLE Statement (ADD Column, MODIFY DATATYPE, RENAME COLUMN, DROP COLUMN) | **50** |
|  | INSERT INTO Statement | **10** |
|  | SELECT and DISTINCT Statement | **20** |
|  | WHERE Clause using AND, OR and NOT Operators | **50** |
|  | ORDER BY Statement | **25** |
|  | ORDER BY using AND, OR and NOT Operators | **25** |
|  | GROUP BY Statement | **25** |
|  | GROUP BY using AND, OR, NOT Operators and Group by | **25** |
|  | Subqueries | **30** |
|  | Subqueries | **30** |
|  | Aggregate functions MAX, MIN, SUM, COUNT, and AVG. | **20** |
|  | Aggregate functions using logical Operators and Group by | **30** |
|  | INNER Joins | **20** |
|  | INNER Joins using logical Operators, Group by and Order by | **30** |
|  | LEFT JOIN | **20** |
|  | RIGHT JOIN | **20** |
|  | FULL OUTER JOIN | **20** |
|  | Stored Procedures without parameter | **25** |
|  | Stored Procedures with parameter | **25** |
|  | Stored Procedures with parameter using logical Operators and Group by | **30** |
|  | DML Triggers INSERT | **20** |
|  | DML Triggers UPDATE | **20** |
|  | DML Triggers DELETE | **20** |
|  | VIEW Statement | **10** |
|  | VIEW Statement using logical Operators | **30** |
|  | Single-Row Functions UPPER, LOWER, INITCAP, CONCAT, LENGTH, SUBSTR using logical operators | **50** |
|  | Single-Row Functions INSTR, TRIM, REPLACE, ROUND, TRUNC using logical operators | **50** |
|  | Transaction COMMIT and ROLLBACK | **20** |
|  | 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 addresses containing '@example.com'. | SELECT \* FROM Teachers WHERE contact\_number IS NULL OR email LIKE '%@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 attendances for specific subject\_ids. | SELECT subject\_id, COUNT(\*) AS total\_attendance  FROM Attendance  WHERE subject\_id = 1 OR subject\_id = 4  GROUP BY subject\_id; |
| 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 'cs'. | SELECT \*  FROM Teachers  WHERE department\_id = (  SELECT department\_id  FROM Departments  WHERE department\_name = 'CS'  ); |
| 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, Department Name, and Payment Date | SELECT fees.\*, Students.first\_name, Students.last\_name, departments.department\_name, fees.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 student department, ordered by payment date. | 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; |
| 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, 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. | SELECT Students.first\_name,Students.student\_id, Enrollments.enrollment\_id, Enrollments.subject\_id  FROM Students  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 information, student's first name, student's last name, and payment date. The results are grouped by student ID. | SELECT fees.\*, Students.first\_name, Students.last\_name, fees.payment\_date  FROM fees  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 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. | 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'  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 their subject IDs for teachers belonging to the departments of Mathematics or Physics. The results are ordered by the teachers' first names. | SELECT Teachers.first\_name,Teachers.subject\_id  FROM Teachers  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 greater than 5000. The results are ordered by the teachers' first names. | 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  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 student ID in ascending order. | SELECT Students.\*, sdepartments.sdepartment\_name  FROM Students  LEFT JOIN sdepartments ON Students.sdepartment\_id = sdepartments.sdepartment\_id  ORDER BY Students.student\_id ASC; |

**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  dsdsd | 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  dasdd | 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  sds | 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 female students with specific contact or email. | SELECT LOWER(Aaddress) FROM Students WHERE gender = 'female'  AND (contact\_number = '123456789' OR email = 'example@example.com'); |  | |
| 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 and contact number. | SELECT UPPER(email) FROM Students WHERE gender = 'female' AND  sdepartment\_id = 1 AND contact\_number LIKE '%123%'; |  |
| 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, subdepartment, or gender. | SELECT LEN(email) FROM Students WHERE (student\_id = 1 AND  sdepartment\_id = 2) OR (gender = 'Female' AND sdepartment\_id = 3); |  |
| 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; |