Here is your **complete professional documentation** for the Grade Reporting System SQL code, **line-by-line explanation**, formatted as clean, ready-to-print **PDF content**.

Copy this entire text and paste into Microsoft Word → Save as PDF → Done.

Markdown

~~~markdown
# GRADE REPORTING SYSTEM
## Database Design & SQL Implementation
### Wolkite University – College of Computing and Informatics
### DBMS Project Documentation
### December 2025

---

### SQL CODE WITH LINE-BY-LINE EXPLANATION

```sql
DROP DATABASE IF EXISTS GradeReportingSystem;
~~~

- Removes the database if it already exists to avoid conflicts during recreation.

SQL

```sql
CREATE DATABASE GradeReportingSystem;
```

- Creates a new database named GradeReportingSystem.

SQL

```sql
USE GradeReportingSystem;
```

- Sets GradeReportingSystem as the active database for all subsequent commands.

SQL

```sql
CREATE TABLE Departments (
    DepartmentID   INT AUTO_INCREMENT PRIMARY KEY,
    DepartmentName VARCHAR(100) NOT NULL UNIQUE
);
```

- Creates Departments table
- DepartmentID: Auto-incremented unique identifier (Primary Key)
- DepartmentName: Name of department, must be unique and cannot be null

SQL

```
CREATE TABLE Instructors (
    InstructorID   INT AUTO_INCREMENT PRIMARY KEY,
    FirstName      VARCHAR(50) NOT NULL,
    LastName       VARCHAR(50) NOT NULL,
    DepartmentID   INT NOT NULL,
    Email          VARCHAR(100) UNIQUE NOT NULL,
    Phone          VARCHAR(15),
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

- Creates Instructors table
- InstructorID: Unique auto-incremented ID
- FirstName & LastName: Required fields
- DepartmentID: Links instructor to a department
- Email: Must be unique across all instructors
- Phone: Optional contact number
- FOREIGN KEY: Establishes one-to-many relationship (One department → Many instructors)
- ON DELETE RESTRICT: Prevents deleting a department if instructors exist
- ON UPDATE CASCADE: Updates DepartmentID automatically if changed

```
SQL

CREATE TABLE Students (
    StudentID      INT AUTO_INCREMENT PRIMARY KEY,
    FirstName      VARCHAR(50) NOT NULL,
    LastName       VARCHAR(50) NOT NULL,
    DateOfBirth    DATE NOT NULL,
    Email          VARCHAR(100) UNIQUE,
    Phone          VARCHAR(15),
    Program        VARCHAR(100) NOT NULL,
    EnrollmentYear YEAR NOT NULL
);
```

- Creates Students table
- StudentID: Auto-generated unique student identifier
- Email: Optional but must be unique if provided
- Program: Student’s field of study (e.g., Computer Science)
- EnrollmentYear: Year student joined the university

```
SQL

CREATE TABLE Courses (
    CourseCode     VARCHAR(10) PRIMARY KEY,
    CourseName     VARCHAR(100) NOT NULL,
    CreditHours    INT NOT NULL CHECK (CreditHours > 0),
    DepartmentID   INT NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
```

- Creates Courses table
- CourseCode: Unique identifier (e.g., CS301) used as Primary Key
- CreditHours: Must be positive integer
- DepartmentID: Indicates which department offers the course
- FOREIGN KEY: One department can offer many courses

SQL

```sql
CREATE TABLE CourseOfferings (
    OfferingID     INT AUTO_INCREMENT PRIMARY KEY,
    CourseCode     VARCHAR(10) NOT NULL,
    InstructorID   INT NOT NULL,
    AcademicYear   VARCHAR(9) NOT NULL,
    Semester       ENUM('1','2','Summer') NOT NULL,
    FOREIGN KEY (CourseCode) REFERENCES Courses(CourseCode)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (InstructorID) REFERENCES Instructors(InstructorID)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    UNIQUE KEY unique_offering (CourseCode, InstructorID, AcademicYear, Semes
);
```

- Creates CourseOfferings table (represents a specific section of a course in a semester)
- OfferingID: Unique ID for each course section
- Links one course to one instructor in a specific academic year and semester
- UNIQUE constraint prevents duplicate offerings of the same course by same instructor in same semester

SQL

```
CREATE TABLE Enrollments (
    EnrollmentID   INT AUTO_INCREMENT PRIMARY KEY,
    StudentID      INT NOT NULL,
    OfferingID     INT NOT NULL,
    EnrollmentDate DATE DEFAULT (CURRENT_DATE),
    MidExam        DECIMAL(5,2) CHECK (MidExam BETWEEN 0 AND 100),
    FinalExam      DECIMAL(5,2) CHECK (FinalExam BETWEEN 0 AND 100),
    Assignment     DECIMAL(5,2) CHECK (Assignment BETWEEN 0 AND 100),
    TotalGrade     DECIMAL(5,2) AS (COALESCE(MidExam,0)*0.40 + COALESCE(Final
    LetterGrade    VARCHAR(2)  AS (
        CASE
            WHEN TotalGrade >= 90 THEN 'A+'
            WHEN TotalGrade >= 85 THEN 'A '
            WHEN TotalGrade >= 80 THEN 'A-'
            WHEN TotalGrade >= 75 THEN 'B+'
            WHEN TotalGrade >= 70 THEN 'B '
            WHEN TotalGrade >= 65 THEN 'B-'
            WHEN TotalGrade >= 60 THEN 'C+'
            WHEN TotalGrade >= 50 THEN 'C '
            WHEN TotalGrade >= 40 THEN 'D '
            ELSE 'F'
        END
    ) STORED,
    Status         ENUM('Registered','In Progress','Completed','Withdrawn') D
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (OfferingID) REFERENCES CourseOfferings(OfferingID)
        ON DELETE CASCADE ON UPDATE CASCADE,
    UNIQUE KEY unique_enrollment (StudentID, OfferingID)
);
```

- Main table for storing student grades
- TotalGrade: Automatically calculated (40% Mid + 50% Final + 10% Assignment)
- LetterGrade: Automatically assigned based on total score
- UNIQUE constraint prevents a student from enrolling twice in the same course offering
- CASCADE DELETE: If a student or course offering is deleted, enrollment record is removed

```
SQL

INSERT INTO Departments (DepartmentName) VALUES
('Computer Science'), ('Software Engineering'), ('Information Technology');
```

- Inserts three academic departments

```
SQL

INSERT INTO Instructors (FirstName, LastName, DepartmentID, Email) VALUES
('Bereket', 'Bedilu', 1, 'bereket@wku.edu.et'),
('Fenet', 'Girma', 2, 'fenet@wku.edu.et');
```

- Adds two instructors to their respective departments

```
SQL

INSERT INTO Students (FirstName, LastName, DateOfBirth, Email, Program, Enrol
('Anthony', 'Woldetensay', '2002-06-15', 'anthony@wku.edu.et', 'CS', 2024),
('Firaol', 'Desalegn', '2001-11-20', 'firaol@wku.edu.et', 'SE', 2024);
```

- Registers two students

```
SQL

INSERT INTO Courses (CourseCode, CourseName, CreditHours, DepartmentID) VALUE
('CS301', 'Database Systems', 4, 1),
('SE302', 'Software Engineering', 3, 2);
```

- Adds two courses under correct departments

```
SQL

INSERT INTO CourseOfferings (CourseCode, InstructorID, AcademicYear, Semester
('CS301', 1, '2025-2026', '1'),
('SE302', 2, '2025-2026', '1');
```

- Creates course sections taught by specific instructors in 2025–2026 Semester 1

```
SQL

INSERT INTO Enrollments (StudentID, OfferingID, MidExam, FinalExam, Assignmen
(1, 1, 88, 92, 95, 'Completed'),
(2, 2, 78, 85, 80, 'Completed');
```

- Records grades for students
- TotalGrade and LetterGrade are automatically calculated
- Example: Student 1 → Total = 90.7 → LetterGrade = A+

Database Features Implemented:

- Full Normalization (3NF+)
- Referential Integrity via Foreign Keys
- Automatic Grade Calculation
- Prevent Duplicate Enrollments
- Secure and Scalable Design
- Ready for University Use

Prepared by Group 2 – Wolkite University

text

```
**How to get PDF:**
1. Copy all the above text
2. Paste into Microsoft Word or Google Docs
3. Format title as Heading 1
4. Save/Export as PDF

Your documentation is now **100% complete, professional, and ready for submis
```