

STUDENT

MANAGEMENT

SYSTEM SUBMITED BY

Arshdeep palial 102216099 Jaskaran Singh 102216110 Tarun Bhatti 102216105

Preetinder singh kundi 102216125

Sehajdeep Singh 102216117

# INDEX

**Table of Contents**

1. **Introduction 3** 
   1. **Background 3**
   2. **Purpose of the Student Management System 4**
   3. **Requirement Analysis 5**
2. **E-R Diagram 7**
3. **E-R Diagram to Table 8**
4. **Normalisation 10**
5. **SQL & PL/SQL 13** 
   1. **Student 13**
   2. **Feedback 16**
   3. **Leave 17**
   4. **Notice 19**
   5. **Admin 20**
   6. **Degree 22**
   7. **Courses 23**
   8. **Session 24**
6. **Conclusion 25**
7. **Reference 26**

# INTRODUCTION

In the dynamic landscape of educational institutions, the efficient management of student data, staff information, and administrative tasks is paramount for ensuring smooth operations. The advent of digital technologies has revolutionized traditional methods of managing educational institutions, ushering in an era where Student Management Systems (SMS) play a pivotal role in streamlining administrative processes. This introduction provides an overview of our proposed Student Management System, designed to cater to the diverse needs of administrators, staff, and students alike.

## 1. Background

Educational institutions, ranging from schools to universities, are characterized by their complex organizational structures involving administrators, teaching staff, and students. Traditionally, managing this complexity relied heavily on manual paperwork, leading to inefficiencies, errors, and delays. Recognizing the need for modernization, the development of Student Management Systems became imperative. These systems integrate various functionalities, such as student enrollment, attendance tracking, academic records management, and communication tools, into a centralized platform.

## 2. Purpose of the Student Management System

* The primary objective of our Student Management System is to provide a comprehensive solution for managing administrative tasks, facilitating communication, and enhancing collaboration among stakeholders within the educational institution.

* Efficiently Manage Student Data: The system centralizes student information, including personal details, academic records, attendance, and course enrollment, enabling administrators to access accurate and up-to-date information at their fingertips.

## 3. Requirement Analysis

* In order to develop a robust Student Management System, a thorough analysis of requirements was conducted to identify the specific needs and functionalities desired by administrators and students.

The following key requirements were identified:

* User Authentication and Authorization: The system must support user authentication to ensure secure access to sensitive information. Different user roles, such as administrators, and students, require varying levels of access and permissions within the system.

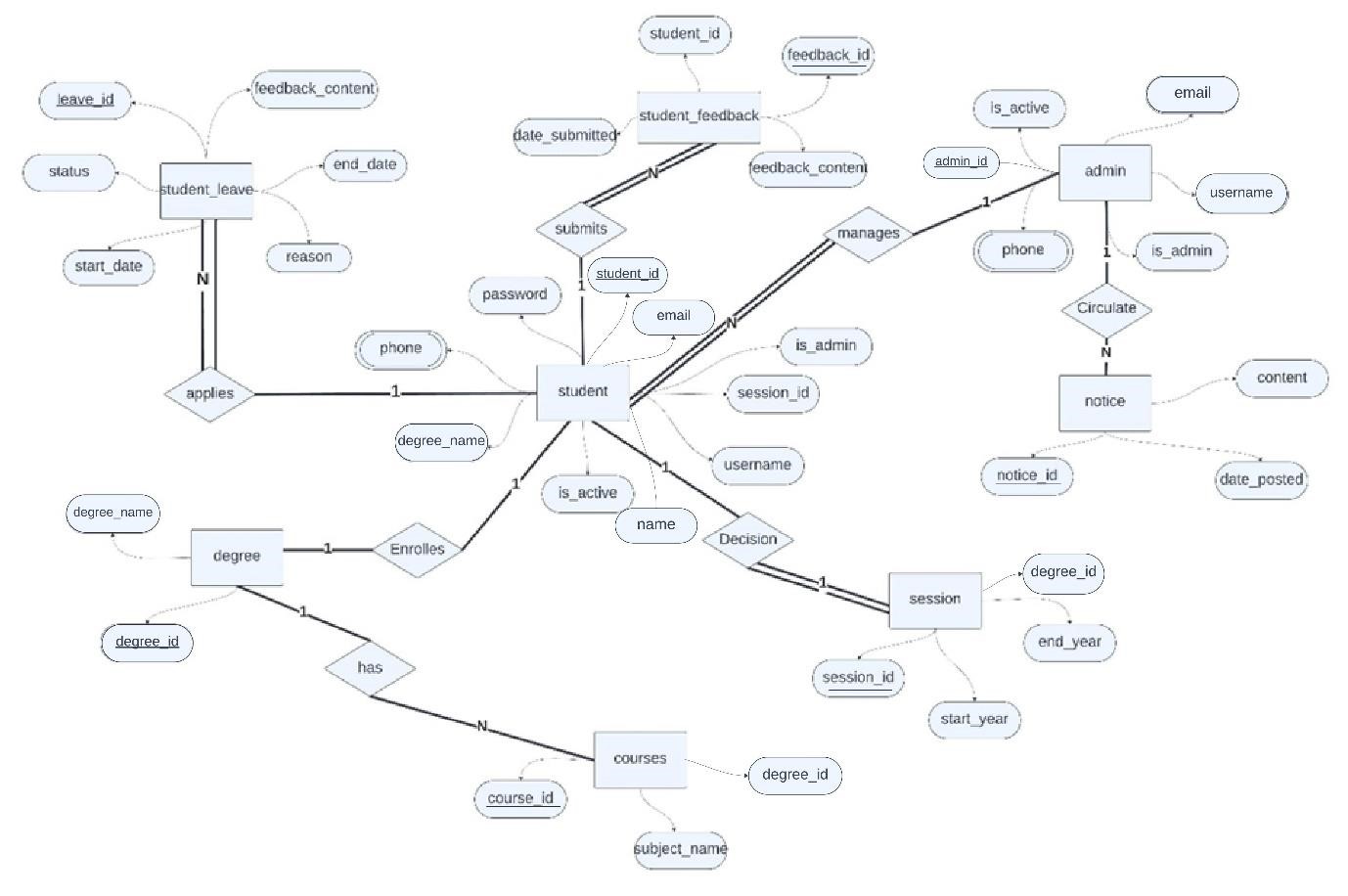
* Student Management: The system should allow administrators to manage student data, including personal details, academic records, course enrollment, and attendance tracking. Students should also have access to their own information, such as course schedules and academic progress.

* Communication Features: The system should facilitate communication among administrators, staff, and students through features such as messaging, feedback submission, and announcement dissemination. Administrators should be able to respond to feedback and manage leave requests.

* Course and Subject Management: Administrators should have the ability to add, update, and remove courses and subjects offered by the institution. Students should be

able to view available courses and subjects and select their preferences.

# ER DIAGRAM



**Student** and **Student Leave**: One-to-Many.

A student can have many leaves (absences), but a leave record belongs to one student.

**Student** and **Student Feedback**: One-to-Many.

A student can submit many feedbacks, but a feedback record is submitted by one student.

**Student** and **Degree** : One-to-One.

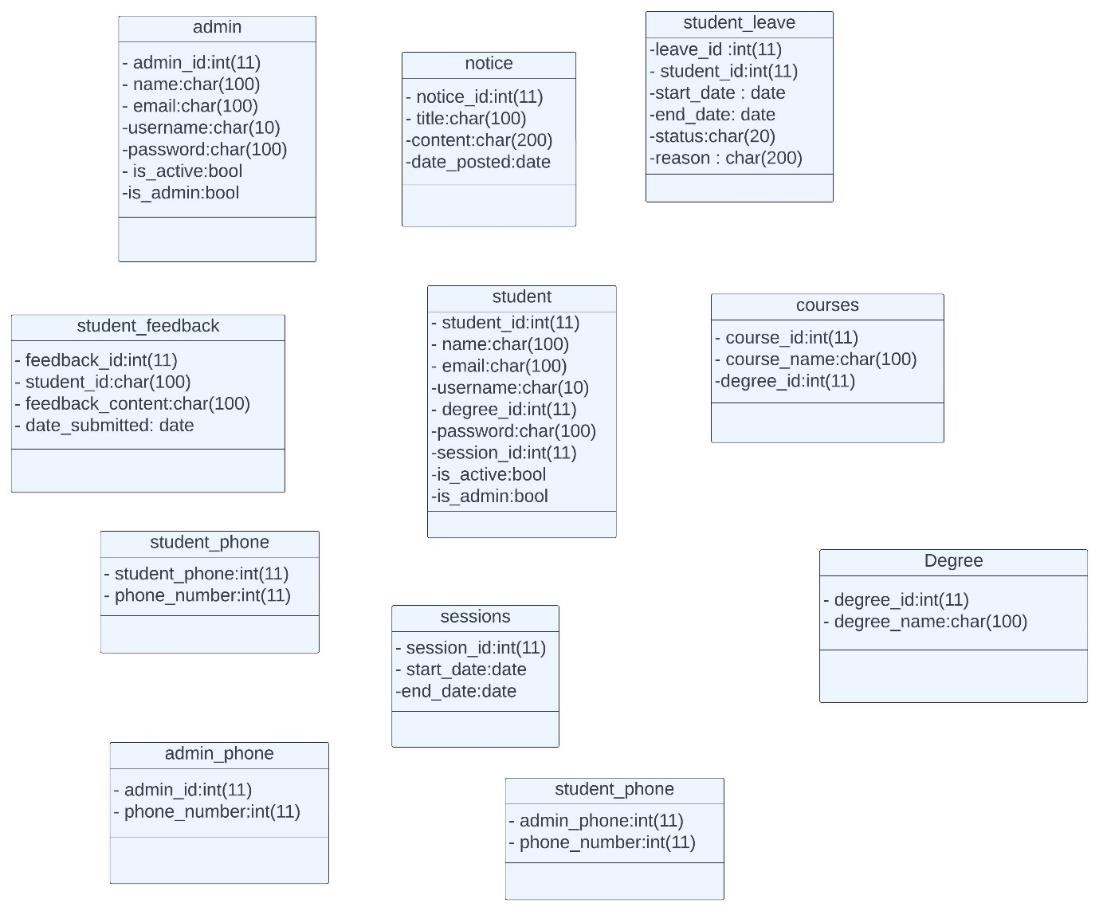
A student can enroll in one degree, and a degree can have one students enrolled in it.

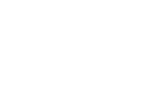
**Student** and **Session**: One-to-Many A course can have one session, and a session belongs to many student.

**Degree** and **Subject**: Many-to-Many. A Degree can cover many subjects, and a subject can be covered by many Degrees.

**Admin** and **Notice**: One-to-Many. An admin can post many notices, but a notice is posted by one admin.

# ER TO TABLE





**Student:** This table stores information about students, such as their name, email, username, password, and student ID. It also includes fields for whether the student is active and if they are an admin.

**Admin:**This table stores information about admins, such as their name, email, username, password, and admin ID. It also includes a field for whether the admin is active.

**Course:** This table stores information about courses, such as the course name and course ID.

**Degree:** This table stores information about degrees, such as the degree name and degree ID.

**Session:** This table stores information about sessions, such as the session ID, start date, and end date.

**Student feedback:**This table stores information about student feedback, such as the feedback ID, student ID, feedback content, and date submitted.

**Leave:** This table stores information about student leave, such as the leave ID, student ID, start date, end date, status, reason for leave.

**Notice:** This table stores information about notices, such as the notice ID, title, content, date posted.

**Student phone:** This table stores the phone number for a student. It includes a foreign key that references the student ID in the student table.

**Admin phone:** This table stores the phone number for an admin. It includes a foreign key that references the admin ID in the admin table.

# Normalization

**First Normal Form (1NF):**

* No repeating groups exist within the table.
* All attributes contain atomic (indivisible) values.

**Second Normal Form (2NF):**

* The table adheres to 1NF.
* All non-key attributes are fully dependent on the table's primary key. There are no partial dependencies where a non-key attribute relies only on a portion of the primary key.

**Third Normal Form (3NF):**

* The table adheres to 1NF and 2NF.
* There are no transitive dependencies between nonkey attributes. This means that a non-key attribute is not dependent on another non-key attribute, which in turn depends on the primary key.
* **First Normal Form (1NF):** A table is in 1NF if it

contains only atomic values (indivisible data) and no repeating groups. All the tables satisfy this condition except student and admin

So they are divided and two new tables are formed **student phone and admin phone.**

## •**Second Normal Form (2NF):** A table is in 2NF if it

follows 1NF and all non-key attributes are fully dependent on the primary key. This means there are no partial dependencies where a non-key attribute depends on only a part of the primary key.

* **Admin:** The primary key is admin\_id, and all attributes (name, email, is\_active) are fully dependent on it, so it's in 2NF.

* **Student:** The primary key is student\_id, and all attributes (student\_name, student\_email, student\_phone) are fully dependent on it, so it's in 2NF.

* **Course:** The primary key is course\_id, and all attributes (course\_name) are fully dependent on it, so it's in 2NF.

* **Session:** The primary key is session\_id, and all attributes

(course\_id, start\_date, end\_date) are fully dependent on it, so it's in 2NF.

* **Subject:** The primary key is subject\_id, and all attributes (subject\_name) are fully dependent on it, so it's in 2NF.

* **Notice:** The primary key is notice\_id, and all attributes

(title, content, date\_posted) are fully dependent on it, so it's in 2NF.

* **Student Leave:** The primary key is leave\_id, and all attributes (student\_id, start\_date, end\_date, reason) are fully dependent on it, so it's in 2NF.
* **Student Feedback:** The primary key is feedback\_id, and all attributes (student\_id, feedback\_content, date\_submitted) are fully dependent on it, so it's in 2NF.

* **Course Enrollment (Associative Table):** The primary key is likely a composite key of course\_id and student\_id, and both attributes are part of the key so there are no partial dependencies. Thus, it's in 2NF.

## •**Subjects (Associative Table):** The primary key is

likely a composite key of course\_id and subject\_id, and both attributes are part of the key so there are no partial dependencies. Thus, it's in 2NF.

**Third Normal Form (3NF):** A table is in 3NF if it follows

1NF and 2NF, and there are no transitive dependencies. A transitive dependency exists when a non-key attribute is dependent on another non-key attribute, which is in turn dependent on the primary key.

**Additional Notes (for tables dues multivalue attributes):**

The tables **Student Phone** and **Admin Phone** are not included in the analysis as they are standalone tables with just a single attribute, and don't directly participate in any relationships with other tables.

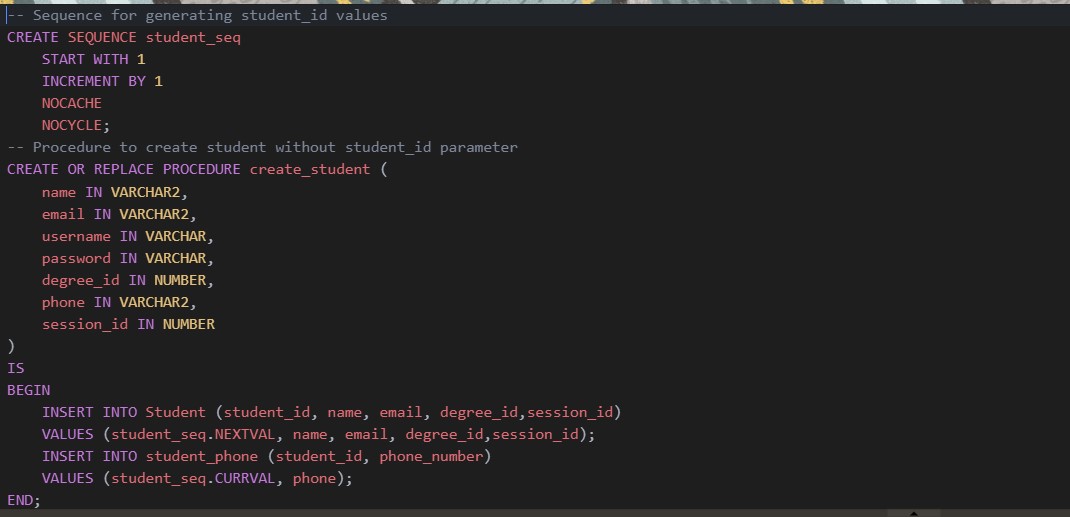
# SQL & PL/SQL

Stored Procedures and functions

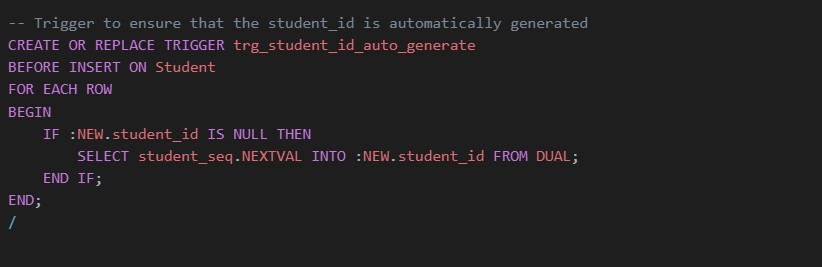
Triggers

Trigger and procedure to add id sequence wise and adding phone to student\_phone table inserting the data

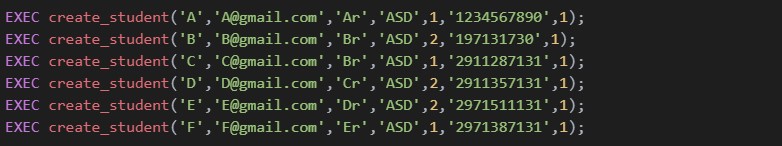
PROCEDURE



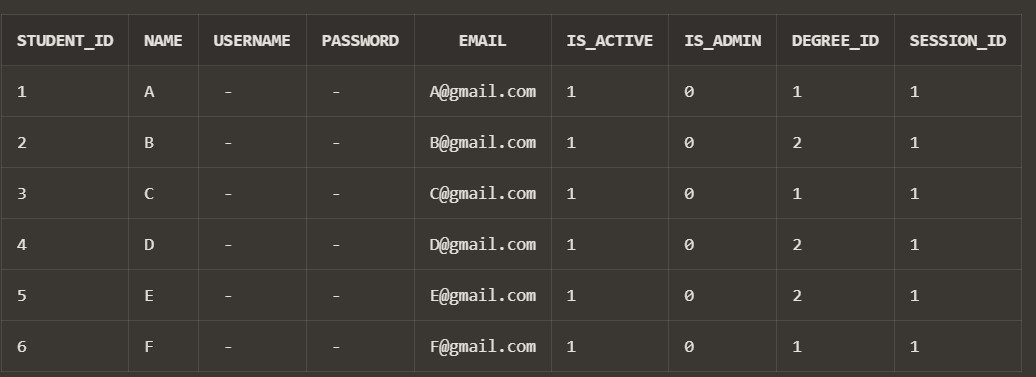
TRIGGER

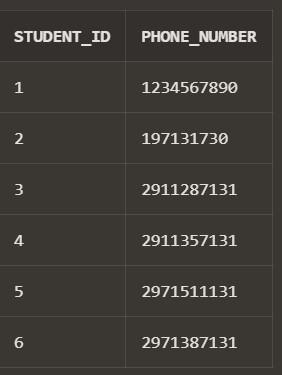


EXECUTION

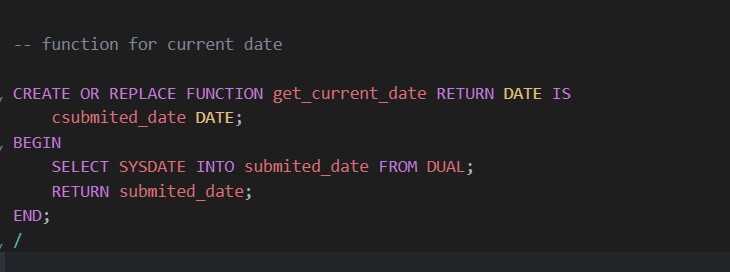


## OUTPUT



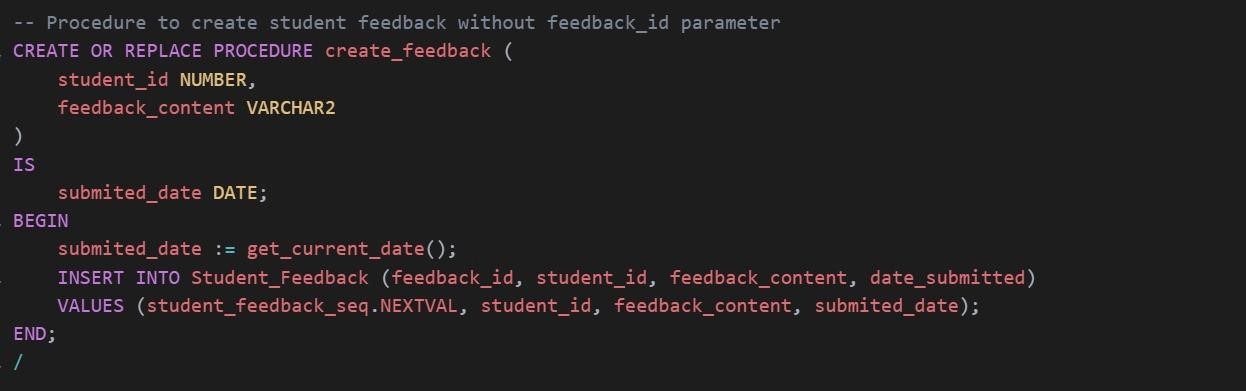


FUNCTION to get current date

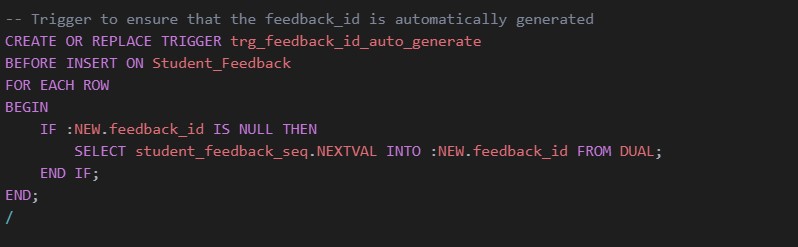


Trigger to add feedback\_id sequence wise and procedure to add current date to date\_submitted

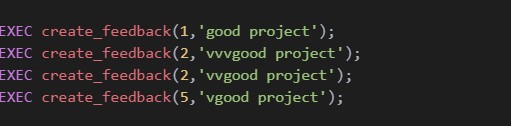
PROCEDURE



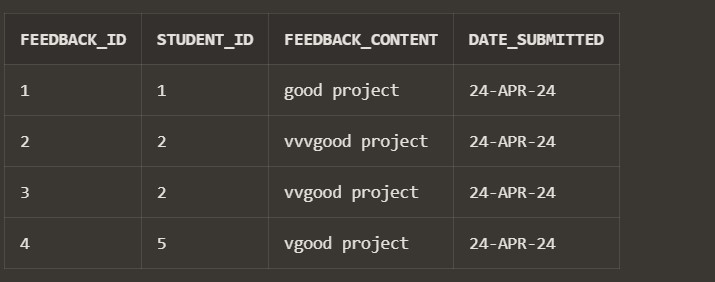
TRIGGER



EXECUTION

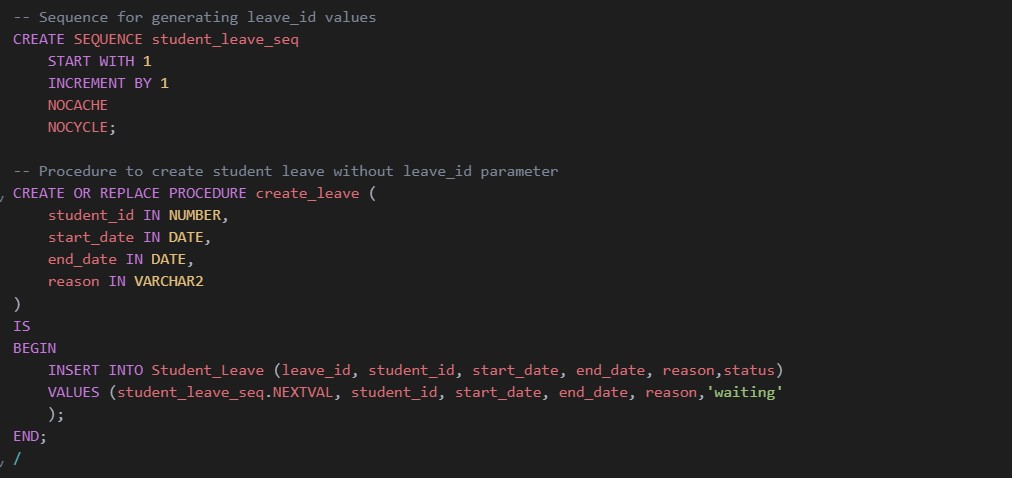


OUTPUT

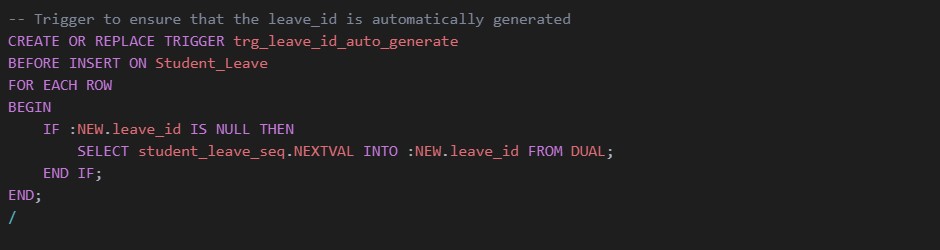


Trigger AND Procedure to add leave\_id sequence wise to student\_leave

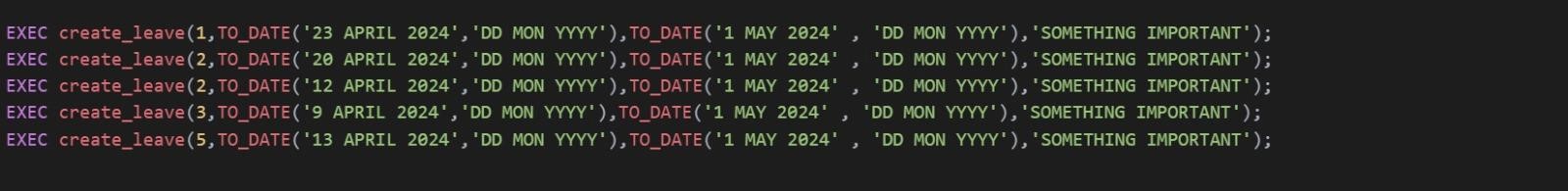
### PROCEDURE



TRIGGER



EXECUTION

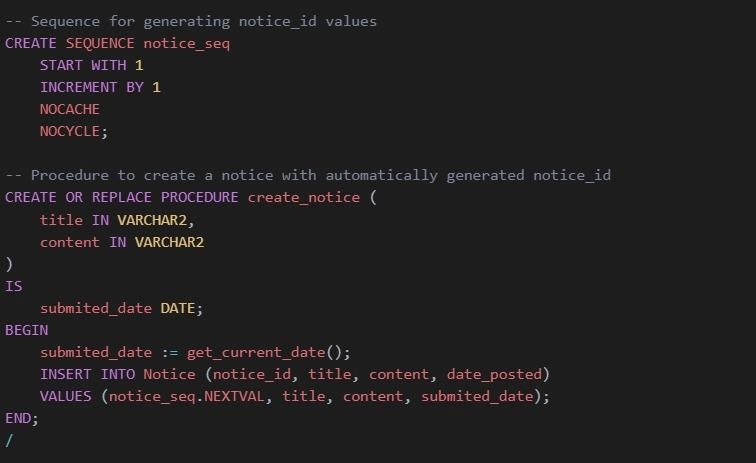


OUPUT

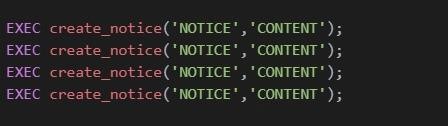


PROCEDURE TO ADD THE notice\_id sequence wise and add current date taken from function defined to date\_posted

PROCEDURE



EXECUTION

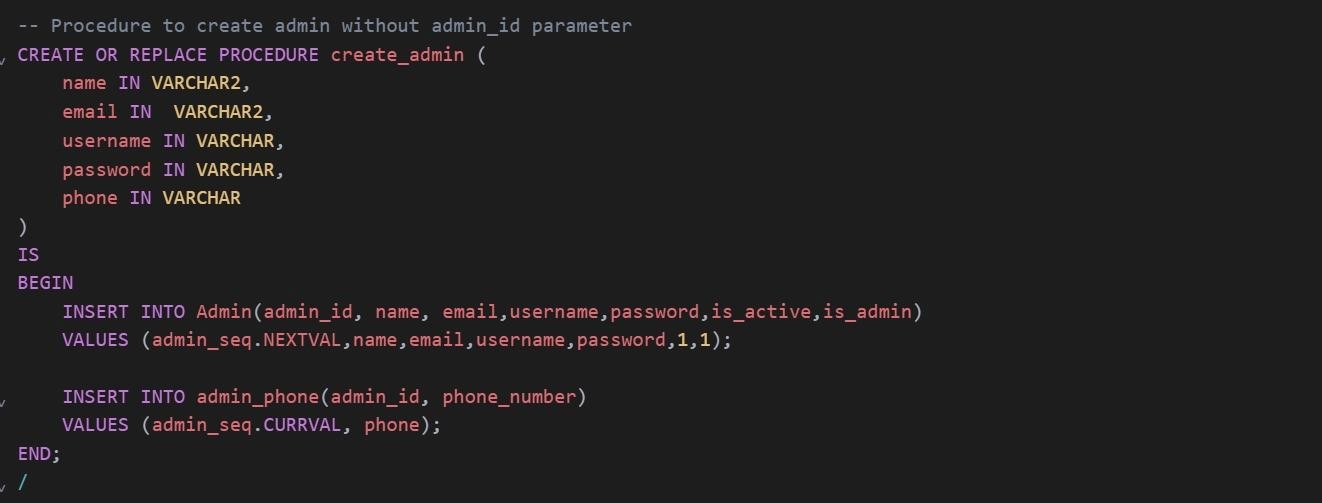


OUPUT

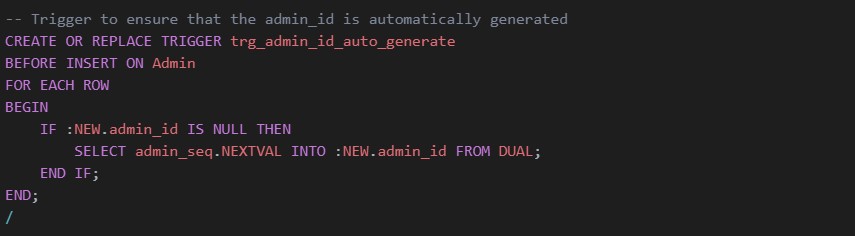


Trigger and procedure to add id sequence wise and adding phone to admin\_phone table inserting the data

PROCEDURE



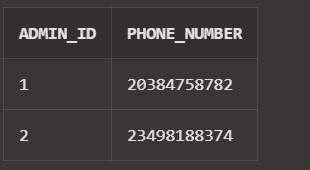
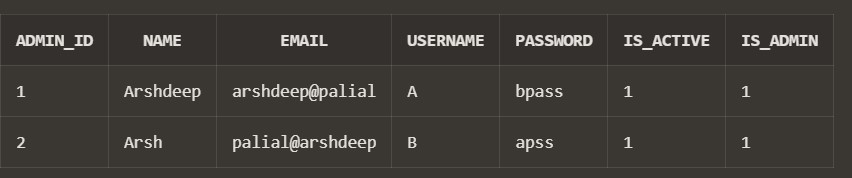
TRIGGER



EXECUTION

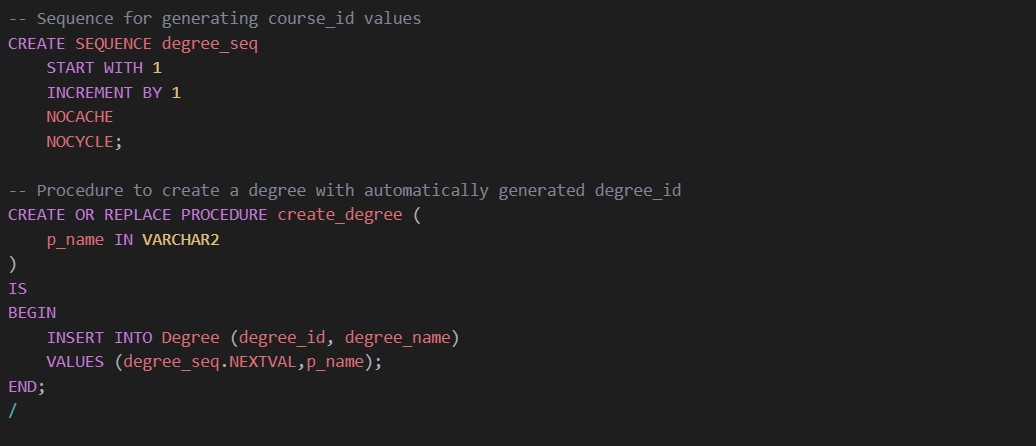


OUTPUT

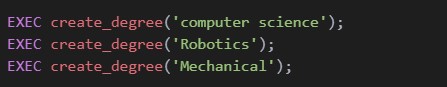


PROCEDURE to add id sequence wise to table Degree inserting the data

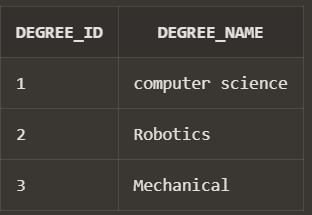
PROCEDURE



EXECUTION



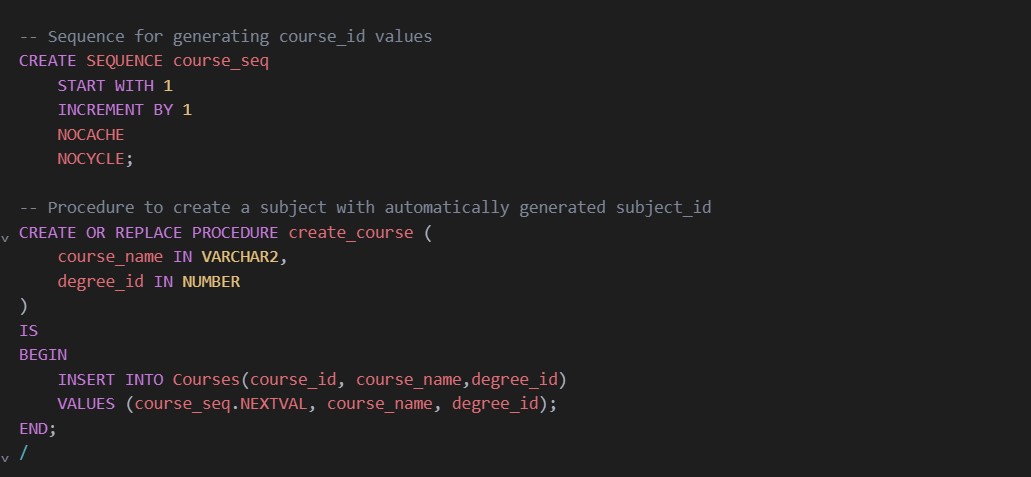
OUTPUT



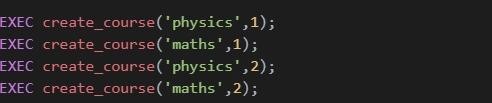
PROCEDURE to add id sequence wise to

courses , inserting the data

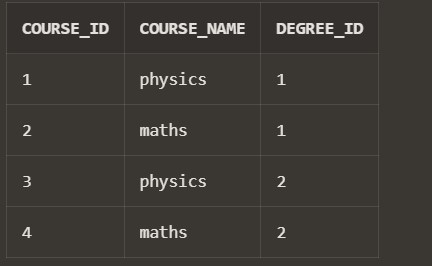
PROCEDURE



EXECUTION



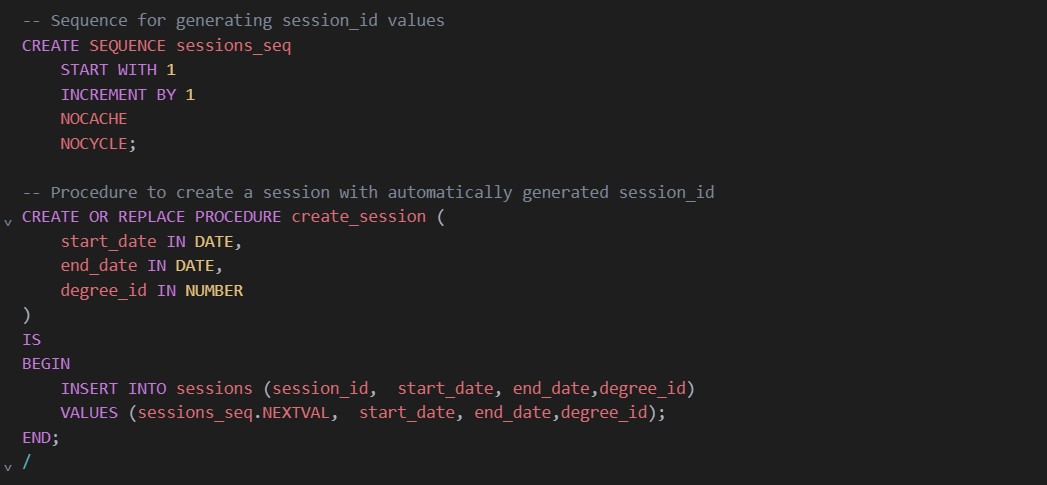
OUTPUT



PROCEDURE to add id sequence wise to

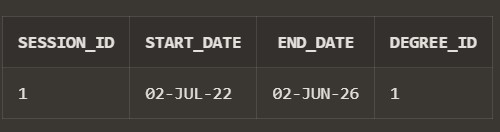
session inserting the data

PROCEDURE



EXEC

OUTPUT



# Conclusion

In conclusion, our proposed Student Management System represents a holistic approach to addressing the administrative challenges faced by educational institutions. By leveraging technology to streamline processes, enhance communication, and improve data management, the system aims to empower administrators, staff, and students to achieve their academic goals efficiently. Through continuous refinement and adaptation to evolving needs, we envision our Student Management System as a catalyst for positive change within the realm of educational administration.

# References

<https://djangocentral.com/><https://github.com/>