

**Bahir Dar University**

Bahir Dar Institute of Technology (Bit)

**Faculty of Computing**

**Department of Software Engineering**

**Advanced Database System**

**(Group Project)**

**Project title: Database Design and Implementation**

**of School management system for**

**Shumabo Secondary and Preparatory School**

**Group Members**

**Name ID**

1. **Abiy Shiferaw.............1306078**
2. **Abrham Desalegn...........1307050**
3. **Jemal Workie..............1307712**
4. **Smachew Gedefaw...........1308736**
5. **Solomon Muhye.............1309375**
6. **Yordanos Ayenew...........1311662**

***Submitted to: Instructor* Tsegawu K.**

***Submit date: February 25, 2022***

**Acknowledgment**

**The development of this school management system is the result of collaborative efforts and we would like to express our gratitude to all who have contributed to its success. We are very grateful and extend our sincere thanks to the principals and staff members of Shum Abo Secondary School for their cooperation, most importantly to the school director Mr. Mesafnt Tefera, Store manager Mrs. Zemenay Abneh and Record officer Mrs. Alemgezi Azmera were kind enough to give us sufficient information about the current system available in the school.**

**Finally, special thanks goes to our classmate Netsanet Alemu who support us and facilitate our communication to the school director.**

**Contents**

[**SECTION 1: Basic Information about the Project 3**](#_Toc128292430)

[**1.1 background information to the existing system 3**](#_Toc128292431)

[**1.2 List of problems in the existing system 4**](#_Toc128292432)

[**1.3 OBJECTIVES OF THE PROJECT 4**](#_Toc128292433)

[**General objectives 4**](#_Toc128292434)

[**Specific objectives of the project. 5**](#_Toc128292435)

[**Scope 5**](#_Toc128292436)

[**SECTION 2: Database Design and implementation 5**](#_Toc128292437)

[**2.1 Design 6**](#_Toc128292438)

[**ER diagrams with their proper descriptions 6**](#_Toc128292439)

[**Mapping of ER diagrams to relational database and Normalized table relationships (logical model) 7**](#_Toc128292440)

[**Database schema diagram (generated by MS SQL Server management studio) 8**](#_Toc128292441)

[**Implementation ( full implementation part ) 8**](#_Toc128292442)

# SECTION 1: Basic Information about the Project

## 1.1 background information to the existing system

The system that we are going to develop is the Student Database Management System for Shum Abo secondary and preparatory school, which will basically come with some sort of features.

Shum Abo Secondary School currently holds and do operations the information of all students and staff members including directors and teachers in a ***manual-based system***. According to the interview with the director, Resource Officer and Registrar of the school, we understand that they store data in two phases. First, when a new data, which is to be stored or manipulated, came into their school, they record the appropriate and required list of attributes (data) in to the manual system. Then this information is stored into a little bit computerized system like structure made of Microsoft products like MS-Word, Excel, … accordingly. This makes the existing system to have sort of computer aided Part, but still the existing system can be considered us entirely manual-based system (operating on hard copy).

**Some Details**

when a new student came into their school, they record the name, sex, age, residence address and the name and address of their parents/guardians. Then this information is stored according to alphabet and section. Second the information of teachers like Employment period, Education period, age, retirement time, and their experience is recorded on hardcopy and softcopy, but this is not only for teacher rather it is also for all staff members. In hardcopy means each information title-by-title record in file and then placed on the cabinets. When the information is required, it is retrieved in the way that first searching the right cabinets, then the right file finally the required information. In softcopy, means the information is stored by folder and then retrieved when needed.

In addition to the students and staff member information, all the activities of the school are recorded at the end of the academic year. Reports are generated to show the practical works of the school in four times in the year. According to the plan, each member of the school including teachers and other staffs report their accomplishment in quarter of the year. Then all of the reports are analyzed and then submitted to the city administration.

Class Representatives checks the attendance of teachers weekly. Then the vice director of the school views the teacher’s attendance form and investigate the reason why the class teachers miss that class and give solution to compensate the missed classes.

Currently administration offices do not use computers for performing their daily tasks. There are a limited number of tasks that are done using Microsoft Office products, such as MS-Word and MS-Excel for performing their daily administrative tasks. All the tasks and calculations are being done manually and paper based.

## 1.2 List of problems in the existing system

The school uses paper based management system for performing various tasks and the school administrators apply their knowledge of hit and miss approach in scheduling classes and courses (preparing the timetable) which wastes labor and much time unnecessarily.

The record officer and teachers prepare transcripts of students manually. The homeroom teachers produce report cards. The homeroom teachers record attendance of students. In order to control absentees and know the number of days that a student has been absent from the school during the school days the attendance officer has to collect the attendance slips from the corresponding homeroom teachers and compile it which is also a time taking process. In addition to that, retrieving records of students who have graduated couple of years ago has been a difficult task and the manual system has difficulty of producing different reports, which are required by the stakeholders.

Teachers may want to associate a student with his parent or emergency persons for disciplinary measures, which need searching of the students, record in the record office. It has been difficult to search a record from thousands of such records and observed that students can take any person claiming that he/she is their parent or emergency person, which creates problem in control of students.

In general, we have tried to mention some problems of the manually system by using the primary and secondary methodologies like observation, interview and data analysis.

List of problems on existing system

* As it is stated in existing system, reports are generated according to the ‘bigar’ and this restrict the staff members because they can’t report other accomplishment beyond specified in the ‘bigar’.
* Shortage of the place to store student and teacher’s data manually using paper or card system
* Lack of security on the staff information
* High work burden for the employee
* Time consuming: - most of the activities done take more time to be done.
* Lack of immediate retrievals and integration of both students and other staff member information: The information is very difficult to retrieve and to find particular information like students record that stored before, the user has to go through various registers and the workers are asked to find the history, these results in difficulty for workers.
* Lack of prompt updating**:** Various changes to information like student or teacher details are difficult to make as paper work is involved.
* Difficulty to generate report.
* The system they are using now is limited to small size of information.
* In this system, computing the grades of students is prone to error and results many problems by falsify the grade of students.
* In this system, the assets of the school may not known exactly because of counting problem.
* In this system, the information is always in change and this causes problems in finance, teacher information.

## 1.3 OBJECTIVES OF THE PROJECT

### General objectives

The rationale of this system is to replace the existing manual school management system of Shum Abo Secondary School that had produced inconvenience not only to the students but also to the school administration. Our goal is to develop a secure, fast, accurate, and user-friendly system.

The main objective of the system is to simplify the school administration and other activities of different staffs. It is used to manage all the information of Shum Abo Secondary School including storing students’ registration, class progress, grade, transfer and parents’ information and staffs’ information and display this information in a tabular form when need by an authorized person.

It allows full and easy control for all the parties of a learning process and work as a bridge between all of them. It allows complete control of the different activities that include class, administration, and more.

### Specific objectives of the project.

* Build a responsive system to manage the different school activities
* Show the information and description of the school
* Student/school performance monitoring: - Trace students/school progress
* Store data in a s**ingle storage/**Centralized data and easy access to all
* Facilitate grades entry process by teachers
* Make a virtual communication between the members of educational process
* Coordinate scheduling and communications between staffs
* F**etch educational reports**
* Automation of registration and enrolment
* Manage student, parents and staffs information
* Consistently update information of all the students
* Manage resources that were managed and handled by manpower previously.

### Scope

The scope of the project involves in:

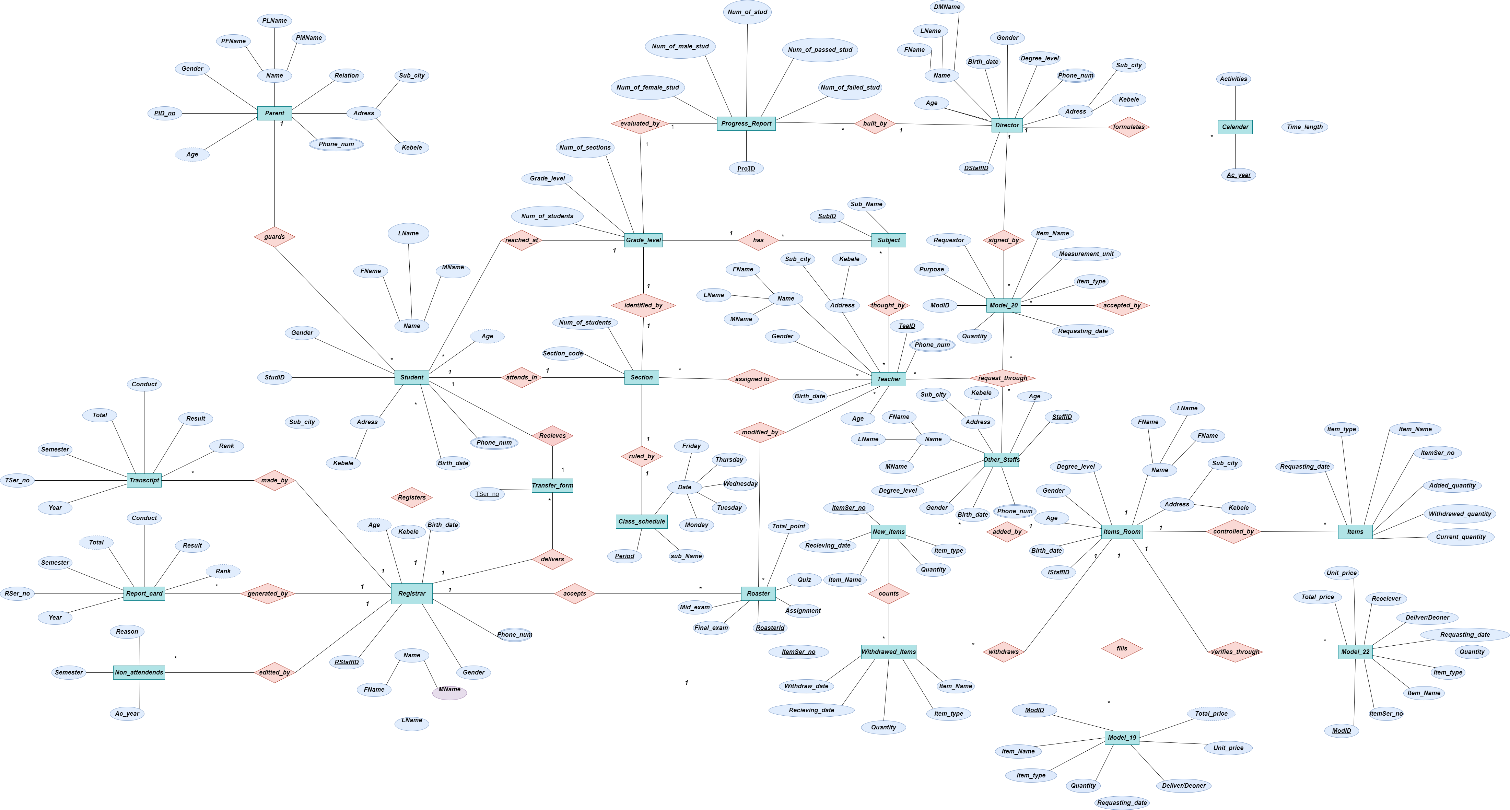
* focuses on recording new students’ information
* And it should allow the staffs in the school to do some tasks based on their privileges
* our system should cover some auditing on the items available.
* our project should include digital grading
* it should help parents to coordinate with the school
* the project covers from student level to director of the school

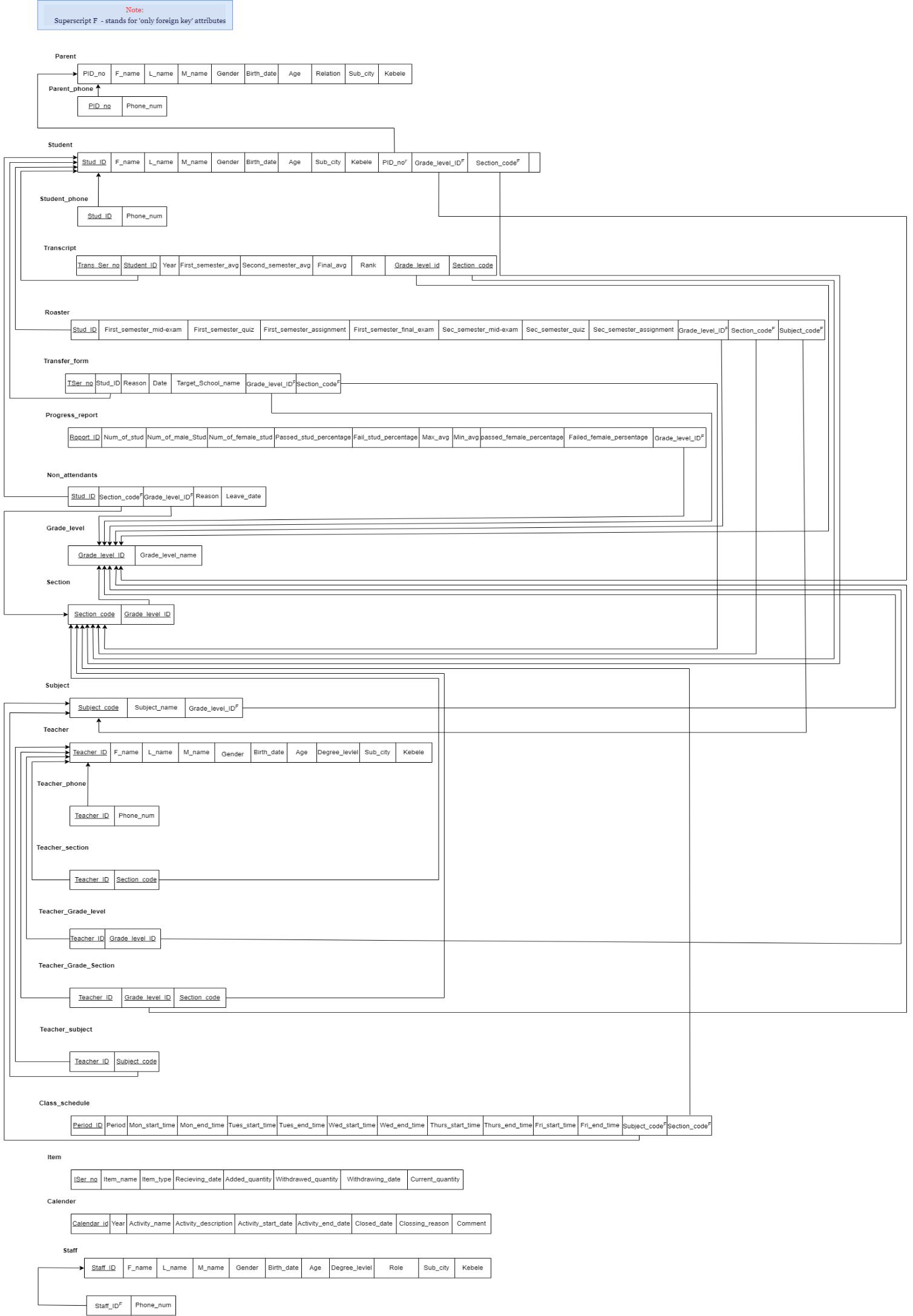
our school management database system would involve managing student records, financial information, tracing attendance, creating reports for teachers, or administrators to review, and providing overall control over the data entry processes. Additionally it may include some of the store and other related staff activities in the school. And as it is for a small school community. it mainly focuses on managing the basic activities that are involved around the school. The main aim of the project is to help the schools to solve the problems of the database management system.

The basic scope of our School Management System (SMS) includes managing student data, tracking student attendance, handling admissions and registrations, storing class schedules, administering grades and assessments, managing invoices and fees, tracking achievement records, providing communication tools for parents or guardians, streamlining administrative processes like payroll and HR management.

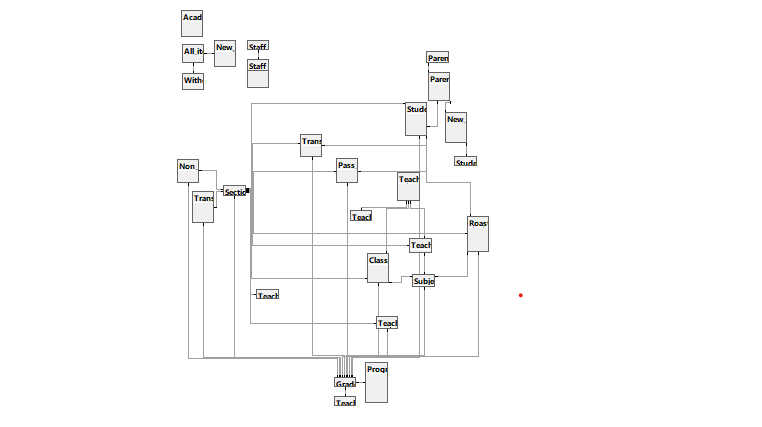
# SECTION 2: Database Design and implementation

### 2.1 Design

ER diagrams with their proper descriptions

Mapping of ER diagrams to relational database and Normalized table relationships (logical model)

Database schema diagram (generated by MS SQL Server management studio)



## Implementation ( full implementation part )

The following code

CREATE DATABASE Shumabo\_Secondary\_school

-- use the created database as a working database

USE Shumabo\_Secondary\_school

-- **creating tables in per each schema and inserting appropriate values**

-- table 1: all parents (of all students) list in schema Stud\_data

CREATE TABLE Stud\_data.Parent (

    PID\_no *VARCHAR*(15) NOT NULL,

    F\_name *VARCHAR*(50),

    L\_name *VARCHAR*(50),

    M\_name *VARCHAR*(50),

    Gender *VARCHAR*(10),

    Birth\_date *DATE*,

    Relation *VARCHAR*(255),

    Sub\_city *VARCHAR*(255),

    Kebele *VARCHAR*(255),

    CONSTRAINT PK\_PID PRIMARY KEY (PID\_no)

);

-- table 2: list of parents' phone number, since we assumed that it will necessary to store more than one phone numbers for each parent

CREATE TABLE Stud\_data.Parent\_phone (

    PID\_no *VARCHAR*(15) NOT NULL,

    Phone\_number *VARCHAR*(20),

    CONSTRAINT PK\_Parnent\_phone PRIMARY KEY (PID\_no, Phone\_number),

    CONSTRAINT FK\_Parent\_Phone FOREIGN KEY (PID\_no) REFERENCES Stud\_data.Parent(PID\_no);

);

-- table 3: list of Grade levels namely:- 9th, 10th, 11th and 12th

CREATE TABLE Assignment.Grade\_level (

  Grade\_level\_ID *VARCHAR*(6) NOT NULL,

  Grade\_level\_name *VARCHAR*(6) NOT NULL,

  CONSTRAINT PK\_Grade\_level PRIMARY KEY (Grade\_level\_ID)

);

-- table 4: list of section which belongs to specific grade level and contain students

CREATE TABLE Assignment.Section (

  Section\_code *VARCHAR*(6) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6),

  CONSTRAINT PK\_Section PRIMARY KEY (Section\_code),

  CONSTRAINT FK\_Section FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level (Grade\_level\_ID)

);

-- table 5: list of all new students which are sregistered in each year

CREATE TABLE Stud\_data.New\_Student (

  Ac\_year *INT*,

  Registration\_date *DATE*,

  Stud\_ID *VARCHAR*(10) NOT NULL,

  F\_name *VARCHAR*(50),

  L\_name *VARCHAR*(50),

  M\_name *VARCHAR*(50),

  Gender *VARCHAR*(6),

  Birth\_date *DATE*,

  Sub\_city *VARCHAR*(50),

  Kebele *VARCHAR*(50),

  PID\_no *VARCHAR*(15) NOT NULL

  CONSTRAINT PK\_Stud PRIMARY KEY (Stud\_ID),

  CONSTRAINT FK\_Stud\_PID FOREIGN KEY (PID\_no) REFERENCES Stud\_data.Parent(PID\_no)

  );

-- table 6: list of all students attended in class

CREATE TABLE Stud\_data.Student (

  Ac\_year *INT*,

  Stud\_ID *VARCHAR*(10) NOT NULL,

  F\_name *VARCHAR*(50),

  L\_name *VARCHAR*(50),

  M\_name *VARCHAR*(50),

  Gender *VARCHAR*(6),

  Birth\_date *DATE*,

  Sub\_city *VARCHAR*(50),

  Kebele *VARCHAR*(50),

  PID\_no *VARCHAR*(15) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6),

  Section\_code *VARCHAR*(6),

  CONSTRAINT PK\_Student PRIMARY KEY (Stud\_ID),

  CONSTRAINT FK\_Student\_PID FOREIGN KEY (PID\_no) REFERENCES Stud\_data.Parent(PID\_no),

  CONSTRAINT FK\_Student\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

  CONSTRAINT FK\_Student\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

  );

-- table 7: list of students' phone number, since we assumed that it will necessary to store more than one phone numbers for each student

CREATE TABLE Stud\_data.Student\_phone (

    phone\_ID *INT* IDENTITY(1,1),

    Stud\_ID *VARCHAR*(10) NOT NULL,

    Phone\_number *VARCHAR*(20) NOT NULL,

    CONSTRAINT PK\_SP PRIMARY KEY (Phone\_ID),

    CONSTRAINT FK\_SPstud FOREIGN KEY (Stud\_ID) REFERENCES Stud\_data.New\_Student(Stud\_ID),

);

-- table 8: a table that contains the yearly results of all students

CREATE TABLE Assignment.Subject(

  Subject\_code *VARCHAR*(6) NOT NULL,

  Subject\_name *VARCHAR*(50) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6),

  CONSTRAINT PK\_Sub PRIMARY KEY (Subject\_code),

  CONSTRAINT FK\_Sub\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID)

);

-- table 9: the transcript list that contains the grade results(marks) of students in differet semester in different year (maximum for 4 years) and each year transcript is identified by the Trans\_ser\_no uniquely given number

CREATE TABLE Result.Roaster (

  Roaster\_ID *INT* IDENTITY(1,1),

  Ac\_year *INT* NOT NULL,

  Stud\_ID *VARCHAR*(10) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6),

  Section\_code *VARCHAR*(6),

  Subject\_code *VARCHAR*(6),

  Frist\_sem\_mid\_exam\_25 *INT*,

  First\_sem\_quiz\_10 *INT*,

  First\_sem\_assignment\_15 *INT*,

  First\_sem\_final\_exam\_50 *INT*,

  Second\_sem\_mid\_exam\_25 *INT*,

  Second\_sem\_quiz\_10 *INT*,

  Second\_sem\_assignment\_15 *INT*,

  Second\_sem\_final\_exam\_50 *INT*,

  CONSTRAINT PK\_Roaster\_Stud PRIMARY KEY (Roaster\_ID),

  CONSTRAINT FK\_Roaster\_Stud FOREIGN KEY (Stud\_ID) REFERENCES Stud\_data.Student(Stud\_ID),

  CONSTRAINT FK\_Roaster\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

  CONSTRAINT FK\_Roaster\_Subject FOREIGN KEY (Subject\_code) REFERENCES Assignment.Subject(Subject\_code),

  CONSTRAINT FK\_Roaster\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

);

-- table 10: Transcript list of  which is the conbind in formation containing

CREATE TABLE Stud\_data.Transcript (

    Trans\_Ser\_no *INT* NOT NULL IDENTITY(1001,2),

    Ac\_Year *INT*,

    Stud\_ID *VARCHAR*(15) NOT NULL,

    Grade\_level\_ID *VARCHAR*(6),

    Section\_code *VARCHAR*(6),

    First\_semester\_total FLOAT,

    Second\_semester\_total FLOAT,

    First\_semester\_avg float,

    Second\_semester\_avg float,

    Final\_avg float,

    Conduct *CHAR*(1)

    CONSTRAINT PK\_Trnas PRIMARY KEY (Trans\_Ser\_no),

    CONSTRAINT FK\_Trans\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

    CONSTRAINT FK\_Trans\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

);

-- table 11: list of transform form lists

CREATE TABLE Stud\_data.Transform\_form (

    TSer\_no *VARCHAR*(10) NOT NULL,

    Stud\_ID *varchar*(10) NOT NULL,

    Grade\_level\_ID *VARCHAR*(6),

    Section\_code *VARCHAR*(6),

    Reason *TEXT*,

    Application\_date *DATE*,

    Target\_school\_name *VARCHAR*(100),

    CONSTRAINT PK\_TForm PRIMARY KEY (TSer\_no),

    CONSTRAINT FK\_Tform\_Stud FOREIGN KEY (Stud\_ID) REFERENCES Stud\_data.Student(Stud\_ID),

    CONSTRAINT FK\_Tform\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

    CONSTRAINT FK\_Tform\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

);

-- table 12: calculated Progress reports for each grade

CREATE TABLE Shumabo.Progress\_report (

    Report\_no *INT* NOT null IDENTITY(111,3),

    Ac\_year *INT* NOT NULL,

    Grade\_level\_Id *VARCHAR*(6) not null,

    Num\_of\_section *INT*,

    Num\_of\_stud *int*,

    Num\_of\_male\_stud *INT*,

    Num\_of\_female\_stud *INT*,

    Max\_avg FLOAT,

    Min\_avg FLOAT,

    Passed\_stud\_percentage FLOAT,

    Failed\_stud\_percentage FLOAT,

    Passed\_male\_percentage FLOAT,

    Failed\_male\_percentage FLOAT,

    Passed\_female\_percentage FLOAT,

    Failed\_female\_percentage FLOAT,

    CONSTRAINT PK\_Progress PRIMARY KEY (Report\_no),

    CONSTRAINT FK\_Progress\_Grade FOREIGN key (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID)

);

-- table 13: list of students with passed and failed students

CREATE TABLE Stud\_data.Pass\_fail\_Student (

  Roll\_no *int* IDENTITY(1,1),

  Ac\_year *INT*,

  Stud\_ID *VARCHAR*(10) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6) NOT NULL,

  Section\_code *VARCHAR*(6) NOT NULL,

  Gender *VARCHAR*(10),

  Final\_avg FLOAT,

  Stud\_status *VARCHAR*(20),

  CONSTRAINT PK\_pfs PRIMARY KEY (Roll\_no),

  CONSTRAINT FK\_pfs\_Stud FOREIGN KEY (Stud\_ID) REFERENCES Stud\_data.Student,

  CONSTRAINT FK\_pfs\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

  CONSTRAINT FK\_pfs\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

);

-- table 14: list of withdrew students

CREATE TABLE Stud\_data.Non\_attendant (

  Ac\_Year *INT*,

  Stud\_ID *VARCHAR*(10) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6) NOT NULL,

  Section\_code *VARCHAR*(6) NOT NULL,

  Gender *VARCHAR*(10),

  Stud\_status *VARCHAR*(20),

  Reason *TEXT*,

  Leave\_date *DATE*,

  CONSTRAINT PK\_NA PRIMARY KEY (Stud\_ID),

  CONSTRAINT FK\_NA\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

  CONSTRAINT FK\_NA\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

)

-- table 15: list of all teachers

CREATE TABLE Teacher\_data.Teacher (

  Teacher\_ID *VARCHAR*(10) NOT NULL,

  F\_name *VARCHAR*(50),

  L\_name *VARCHAR*(50),

  M\_name *VARCHAR*(50),

  Gender *VARCHAR*(8),

  Birth\_date *DATE*,

  Degree\_level *VARCHAR*(50),

  Staff\_role *VARCHAR*(50),

  Sub\_city *VARCHAR*(50),

  Kebele *VARCHAR*(50),

  CONSTRAINT PK\_Teacher PRIMARY key (Teacher\_ID)

);

-- table 16: list of teachers' phone number

CREATE TABLE Teacher\_data.Teacher\_phone (

    phone\_ID *INT* IDENTITY(1,1),

    Teacher\_ID *VARCHAR*(10),

    Phone\_number *varchar*(20),

    CONSTRAINT PK\_Teacher\_phone PRIMARY KEY (phone\_ID),

    CONSTRAINT FK\_Teacher FOREIGN KEY (Teacher\_ID) REFERENCES Teacher\_data.Teacher(Teacher\_ID)

);

-- table 17: assigning Teachers on on to Grade levels

CREATE TABLE Assignment.Teacher\_Grade\_Level (

  Teacher\_ID *VARCHAR*(10) NOT NULL,

  Grade\_level\_ID *VARCHAR*(6),

  CONSTRAINT PK\_TS PRIMARY KEY (Teacher\_ID),

  CONSTRAINT FK\_TS\_Grade FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID)

)

-- table 18: assining a teacher to a section

CREATE TABLE Assignment.Teacher\_section (

  Teacher\_ID *VARCHAR*(10) NOT NULL,

  Section\_code *VARCHAR*(6),

  CONSTRAINT PK\_TSec PRIMARY KEY (Teacher\_ID),

  CONSTRAINT FK\_TS\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

)

-- table 19: teacher grade\_level and Section // it will be better to make this table a view table

 CREATE TABLE Teacher\_Grade\_Section (

  Teacher\_ID *VARCHAR*(10),

  Grade\_level\_ID *VARCHAR*(6),

  Section\_code *VARCHAR*(6),

  CONSTRAINT PK\_TGS PRIMARY KEY (Teacher\_ID),

  CONSTRAINT FK\_TGS\_Grade FOREIGN key (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

  CONSTRAINT FK\_TGS\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

 );

-- table 20: assignining a teacher with specific subjects  check !!

CREATE TABLE Assignment.Teacher\_Section\_Subject (

  Assign\_ID *INT* IDENTITY(1000, 2) not null,

  Teacher\_ID *VARCHAR*(10),

  Section\_code *VARCHAR*(6),

  Subject\_code *VARCHAR*(6),

  CONSTRAINT PK\_TSS PRIMARY KEY (Assign\_ID),

  CONSTRAINT FK\_TSS\_Teacher FOREIGN KEY (Teacher\_ID) REFERENCES Teacher\_data.Teacher(Teacher\_ID),

  CONSTRAINT FK\_TSS\_Subject FOREIGN KEY (Subject\_code) REFERENCES Assignment.Subject(Subject\_code),

  CONSTRAINT FK\_TSS\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code)

);

-- table 21: list of acadamic calendars

CREATE TABLE Shumabo.Academic\_calendar (

    Calendar\_ID *VARCHAR*(6) NOT NULL,

    Ac\_Year *INT*,

    Activity\_name *VARCHAR*(255),

    Activity\_description *VARCHAR*(1000),

    Activity\_start\_date *DATE*,

    Activity\_end\_date *DATE*,

    Comment *VARCHAR*(1000),

    CONSTRAINT PK\_Calendar PRIMARY KEY(Calendar\_ID)

);

-- table 22: list of class schedules for each class

CREATE TABLE Assignment.Class\_Schedule (

    Schedule\_ID *VARCHAR*(10),

    Ac\_year *INT*,

    Section\_code *VARCHAR*(6) NOT NULL,

    Grade\_level\_ID *VARCHAR*(6),

    Subject\_code *VARCHAR*(6),

    Teacher\_ID *VARCHAR*(10),

    Day\_name *VARCHAR*(10),

    period\_no *INT*,

    Start\_time *TIME*,

    End\_time *TIME*,

    CONSTRAINT PK\_Class\_schedule PRIMARY KEY (Schedule\_ID),

    CONSTRAINT FK\_Class\_schedule\_Grade\_level FOREIGN KEY (Grade\_level\_ID) REFERENCES Assignment.Grade\_level(Grade\_level\_ID),

    CONSTRAINT FK\_Class\_schedule\_Section FOREIGN KEY (Section\_code) REFERENCES Assignment.Section(Section\_code),

    CONSTRAINT FK\_Class\_schedule\_Subject FOREIGN KEY (Subject\_code) REFERENCES Assignment.Subject(Subject\_code),

    CONSTRAINT Fk\_Class\_schedule\_Teacher FOREIGN KEY (Teacher\_ID) REFERENCES Teacher\_data.Teacher(Teacher\_ID)

);

-- table 23: list of items in the resource room

CREATE TABLE Resource.All\_item (

    ISer\_no *VARCHAR*(8) not null,

    Item\_name *VARCHAR*(50),

    Item\_type *varchar*(50),

    Unit\_price FLOAT,

    Total\_price FLOAT,

    Current\_quantity *INT*,

    CONSTRAINT PK\_ALL PRIMARY KEY (ISer\_no),

    CONSTRAINT FK\_NI\_New FOREIGN KEY (ISer\_no) REFERENCES Resource.New\_item(ISer\_no),

);

-- table 24: list of newly add items

CREATE TABLE Resource.New\_item (

    ISer\_no *VARCHAR*(8) not null,

    Item\_name *VARCHAR*(50),

    Item\_type *varchar*(50),

    Unit\_price FLOAT,

    Total\_price FLOAT,

    Recieving\_date *Date*,

    Added\_quantity *INT*,

    Item\_resource *TEXT*,

    Comment *TEXT*,

    CONSTRAINT PK\_NewItem PRIMARY KEY (ISer\_no)

);

-- table 25: list of withdrawed itmes whith their description // can be a view table

CREATE TABLE Resource.Withdrew\_Item (

    ISer\_no *VARCHAR*(8) not null,

    Withdrawing\_date *Date*,

    Withdrawed\_quantity *INT*,

    Withdrawing\_reason *TEXT*,

    Withdrawer\_name *VARCHAR*(100),

    CONSTRAINT PK\_WI PRIMARY KEY (ISer\_no),

    CONSTRAINT FK\_WI\_All FOREIGN KEY (ISer\_no) REFERENCES Resource.All\_item(ISer\_no)

);

-- table 26: list of Staff\_data.Staff members

CREATE TABLE Staff\_data.Staff (

  Staff\_ID *VARCHAR*(10) NOT NULL,

  F\_name *VARCHAR*(50),

  L\_name *VARCHAR*(50),

  M\_name *VARCHAR*(50),

  Gender *VARCHAR*(8),

  Birth\_date *DATE*,

  Degree\_level *VARCHAR*(50),

  Staff\_role *VARCHAR*(50),

  Sub\_city *VARCHAR*(50),

  Kebele *VARCHAR*(50),

  CONSTRAINT PK\_Staff\_member PRIMARY KEY (Staff\_ID)

)

-- table 27: list of phone numbers for staff members

CREATE TABLE Staff\_data.Staff\_Phone (

  Phone\_ID *INT* IDENTITY(1,1),

  Staff\_ID *VARCHAR*(10) NOT NULL,

  Phone\_number *VARCHAR*(20),

  CONSTRAINT PK\_SP PRIMARY KEY (Phone\_ID),

  CONSTRAINT FK\_SP\_Staff FOREIGN KEY (Staff\_ID) REFERENCES Staff\_data.Staff(Staff\_ID)

)

-- Inserting values for each tables created

-- inserting values for Table 1

INSERT INTO Stud\_data.Parent (PID\_no, F\_name, L\_name, M\_name, Gender, Birth\_date, Relation, Sub\_city, Kebele)

VALUES ('BDR001', 'Abebe', 'Shiferaw', 'Abiy', 'Male', '1980-01-01', 'Father', 'Tana', 'Kebele 1'),

       ('BDR002', 'Almi', 'Desalegn', 'Abrham', 'Female', '1985-03-15', 'Mother', 'Facilo', 'Kebele 1'),

       ('BDR003', 'Wohy', 'Workie', 'Jemal', 'Male', '1975-05-20', 'Father', 'Gish Abay', 'Kebele 2'),

       ('BDR004', 'Meroda', 'Gedefaw', 'Smachew', 'Female', '1978-07-30', 'Mother', 'Belay Zeleke', 'Kebele 2'),

       ('BDR005', 'Sisay', 'Muhye', 'Solomon', 'Male', '1970-09-12', 'Father', 'Dagmawi Menelek', 'Kebele 3'),

       ('BDR006', 'Selam', 'Ayenew', 'Yordanos', 'Female', '1973-11-22', 'Mother', 'Atse Tewodros', 'Kebele 3');

-- check

-- SELECT \* FROM Stud\_data.Parent

-- inserting values for Table 2

INSERT INTO Stud\_data.Parent\_phone (PID\_no, Phone\_number)

VALUES ('BDR001', '+251 911 123 461'),

       ('BDR001', '+251 911 123 462'),

       ('BDR001', '+251 911 123 463'),

       ('BDR002', '+251 911 123 464'),

       ('BDR002', '+251 911 123 465'),

       ('BDR003', '+251 911 123 466'),

       ('BDR003', '+251 911 123 467'),

       ('BDR004', '+251 911 123 468'),

       ('BDR004', '+251 911 123 469'),

       ('BDR005', '+251 911 123 471'),

       ('BDR006', '+251 911 123 472'),

       ('BDR006', '+251 911 123 473');

-- check

-- SELECT \* FROM Stud\_data.Parent\_phone

-- SELECT \* FROM Stud\_data.Parent JOIN Parent\_phone ON Stud\_data.Parent.PID\_no = Parent\_phone.PID\_no;

-- inserting values for Table 3

INSERT INTO Assignment.Grade\_level (Grade\_level\_ID, Grade\_level\_name)

VALUES ('GID9', '9th'),

       ('GID10', '10th'),

       ('GID11', '11th'),

       ('GID12', '12th');

-- check

-- SELECT \* FROM Assignment.Grade\_level

-- inserting values for Table 4

INSERT INTO Assignment.Section (Section\_code, Grade\_level\_ID)

VALUES ('9A', 'gid9'),

       ('9B', 'gid9'),

       ('10A', 'gid10'),

       ('10B', 'gid10'),

       ('11A', 'gid11'),

       ('11B', 'gid11'),

       ('12A', 'gid12'),

       ('12B', 'gid12');

-- check

-- SELECT \* FROM Assignment.Section

-- inserting values for Table 5

INSERT INTO Stud\_data.Student (Stud\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Sub\_city, Kebele, PID\_no, Grade\_level\_ID, Section\_code)

VALUES ('ST00001', 'Habtsh','Shiferaw','Alemu', 'female', '2005-01-01', 'Tana', 'Kebele 1', 'BDR001','GID11', '11A'),

       ('ST00002', 'Bezabh','Bogale', 'Desalegn', 'Male', '2005-03-15', 'Facilo', 'Kebele 1', 'BDR002', 'GID11', '11B'),

       ('ST00003', 'Gebrie','Wohy', 'Workie', 'Male', '2007-05-20', 'Gish Abay', 'Kebele 2', 'BDR003', 'GID9', '9A'),

       ('ST00004', 'Seblewongel','Meshesha', 'Gedefaw', 'Female', '2006-07-30', 'Belay Zeleke', 'Kebele 2', 'BDR004', 'GID10', '10A'),

       ('ST00005', 'Gudukassa','Sisay', 'Muhye', 'Male', '2007-09-12', 'Dagmawi Menelek', 'Kebele 3', 'BDR005', 'GID9', '9B'),

       ('ST00006', 'Tiruaynet','Selameneh', 'Ayenew', 'Female', '2006-11-22', 'Atse Tewodros', 'Kebele 3', 'BDR006', 'GID10', '12A');

-- cheack

-- SELECT \* FROM Stud\_data.Student

-- for Table 7

INSERT INTO Assignment.Subject (Subject\_code, Subject\_name, Grade\_level\_ID)

VALUES ('SA9', 'Amharic', 'GID9'),

       ('SE9', 'English', 'GID9'),

       ('SM9', 'Maths', 'GID9'),

       ('SB9', 'Biology', 'GID9'),

       ('SP9', 'Physics', 'GID9'),

       ('SCI9', 'Civics', 'GID9'),

       ('SCH9', 'Chemistry', 'GID9'),

       ('SH9', 'History', 'GID9'),

       ('SG9', 'Geography', 'GID9'),

       ('SS9', 'Sport Science', 'GID9'),

       ('SA10', 'Amharic', 'GID10'),

       ('SE10', 'English', 'GID10'),

       ('SM10', 'Maths', 'GID10'),

       ('SB10', 'Biology', 'GID10'),

       ('SP10', 'Physics', 'GID10'),

       ('SCI10', 'Civics', 'GID10'),

       ('SCH10', 'Chemistry', 'GID10'),

       ('SH10', 'History', 'GID10'),

       ('SG10', 'Geography', 'GID10'),

       ('SS10', 'Sport Science', 'GID10'),

       ('SA11', 'Amharic', 'GID11'),

       ('SE11', 'English', 'GID11'),

       ('SM11', 'Maths', 'GID11'),

       ('SB11', 'Biology', 'GID11'),

       ('SP11', 'Physics', 'GID11'),

       ('SCI11', 'Civics', 'GID11'),

       ('SCH11', 'Chemistry', 'GID11'),

       ('SH11', 'History', 'GID11'),

       ('SG11', 'Geography', 'GID11'),

       ('SS11', 'Sport Science', 'GID11'),

       ('SA12', 'Amharic', 'GID12'),

       ('SE12', 'English', 'GID12'),

       ('SM12', 'Maths', 'GID12'),

       ('SB12', 'Biology', 'GID12'),

       ('SP12', 'Physics', 'GID12'),

       ('SCI12', 'Civics', 'GID12'),

       ('SCH12', 'Chemistry', 'GID12'),

       ('SH12', 'History', 'GID12'),

       ('SG12', 'Geography', 'GID12'),

       ('SS12', 'Sport Science', 'GID12');

-- check

--select \* from Assignment.Subject

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES  (2021, 'ST00001', 'GID9', '9A', 'SA9', 14, 10, 15, 40, 10, 15, 10, 35),

        (2021, 'ST00001', 'GID9', '9A', 'SE9', 15, 9, 12, 50, 70, 15, 10, 35),

        (2021, 'ST00001', 'GID9', '9A', 'SM9', 15, 7, 14, 45, 70, 25, 10, 45),

        (2021, 'ST00001', 'GID9', '9A', 'SB9', 11, 6, 13, 46, 70, 25, 10, 45),

        (2021, 'ST00001', 'GID9', '9A', 'SP9', 13, 5, 15, 47, 70, 21, 10, 35),

        (2021, 'ST00001', 'GID9', '9A', 'SCI9', 13, 10, 15, 47, 24, 75, 10, 25),

        (2021, 'ST00001', 'GID9', '9A', 'SCH9', 12, 9, 12, 37, 22, 75, 10, 35),

        (2021, 'ST00001', 'GID9', '9A', 'SH9', 11, 7, 14, 50, 20, 75, 10, 45),

        (2021, 'sT00001', 'GID9', '9A', 'SG9', 13, 8, 13, 49, 24, 10, 11, 50),

        (2021, 'ST00001', 'GID9', '9A', 'SS9', 14, 10, 15, 30, 20, 75, 13, 35);

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES

          (2022, 'ST00001', 'GID10', '10A', 'SA10', 24, 10, 15, 40, 10, 15, 10, 35),

          (2022, 'ST00001', 'GID10', '10A', 'SE10', 25, 9, 12, 50, 70, 15, 10, 35),

          (2022, 'ST00001', 'GID10', '10A', 'SM10', 25, 7, 14, 45, 70, 25, 10, 45),

          (2022, 'ST00001', 'GID10', '10A', 'SB10', 21, 6, 13, 46, 70, 25, 10, 45),

          (2022, 'ST00001', 'GID10', '10A', 'SP10', 23, 5, 15, 47, 70, 21, 10, 35),

          (2022, 'ST00001', 'GID10', '10A', 'SCI10', 23, 10, 15, 47, 24, 75, 10, 25),

          (2022, 'ST00001', 'GID10', '10A', 'SCH10', 22, 9, 12, 37, 22, 75, 10, 35),

          (2022, 'ST00001', 'GID10', '10A', 'SH10', 21, 7, 14, 50, 20, 75, 10, 45),

          (2022, 'sT00001', 'GID10', '10A', 'SG10', 21, 8, 13, 49, 24, 10, 11, 50),

          (2022, 'ST00001', 'GID10', '10A', 'SS10', 24, 10, 15, 30, 20, 75, 13, 35);

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES

            (2023, 'ST00001', 'GID11', '11A', 'SA11', 24, 10, 15, 40, 10, 15, 10, 35),

            (2023, 'ST00001', 'GID11', '11A', 'SE11', 19, 9, 12, 50, 70, 15, 10, 35),

            (2023, 'ST00001', 'GID11', '11A', 'SM11', 15, 7, 14, 45, 70, 25, 10, 45),

            (2023, 'ST00001', 'GID11', '11A', 'SB11', 21, 6, 13, 46, 70, 25, 10, 45),

            (2023, 'ST00001', 'GID11', '11A', 'SP11', 23, 5, 15, 47, 70, 21, 10, 35),

            (2023, 'ST00001', 'GID11', '11A', 'SCI11', 23, 10, 15, 47, 24, 75, 10, 25),

            (2023, 'ST00001', 'GID11', '11A', 'SCH11', 22, 9, 12, 37, 22, 75, 10, 35),

            (2023, 'ST00001', 'GID11', '11A', 'SH11', 21, 7, 14, 50, 20, 75, 10, 45),

            (2023, 'sT00001', 'GID11', '11A', 'SG11', 23, 8, 13, 49, 24, 10, 11, 50),

            (2023, 'ST00001', 'GID11', '11A', 'SS11', 24, 10, 15, 30, 20, 75, 13, 35);

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES

        (2021, 'ST00002', 'GID9', '9B', 'SA9', 24, 10, 15, 40, 10, 15, 10, 35),

        (2021, 'ST00002', 'GID9', '9B', 'SE9', 25, 9, 12, 50, 70, 15, 10, 35),

        (2021, 'ST00002', 'GID9', '9B', 'SM9', 25, 7, 14, 45, 70, 25, 10, 45),

        (2021, 'ST00002', 'GID9', '9B', 'SB9', 19, 6, 13, 46, 70, 25, 10, 45),

        (2021, 'ST00002', 'GID9', '9B', 'SP9', 25, 5, 15, 47, 70, 21, 10, 35),

        (2021, 'ST00002', 'GID9', '9B', 'SCI9', 24, 10, 15, 47, 24, 75, 10, 25),

        (2021, 'ST00002', 'GID9', '9B', 'SCH9', 23, 9, 12, 37, 22, 75, 10, 35),

        (2021, 'ST00002', 'GID9', '9B', 'SH9', 22, 7, 14, 50, 20, 75, 10, 45),

        (2021, 'sT00002', 'GID9', '9B', 'SG9', 23, 8, 13, 49, 24, 10, 11, 50),

        (2021, 'ST00002', 'GID9', '9B', 'SS9', 24, 10, 15, 30, 20, 75, 13, 35);

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES

          (2022, 'ST00002', 'GID10', '10B', 'SA10', 24, 10, 15, 40, 10, 15, 10, 35),

          (2022, 'ST00002', 'GID10', '10B', 'SE10', 25, 9, 12, 50, 70, 15, 10, 35),

          (2022, 'ST00002', 'GID10', '10B', 'SM10', 25, 7, 14, 45, 70, 25, 10, 45),

          (2022, 'ST00002', 'GID10', '10B', 'SB10', 22, 6, 13, 46, 70, 25, 10, 45),

          (2022, 'ST00002', 'GID10', '10B', 'SP10', 22, 5, 15, 47, 70, 21, 10, 35),

          (2022, 'ST00002', 'GID10', '10B', 'SCI10', 23, 10, 15, 47, 24, 75, 10, 25),

          (2022, 'ST00002', 'GID10', '10B', 'SCH10', 22, 9, 12, 37, 22, 75, 10, 35),

          (2022, 'ST00002', 'GID10', '10B', 'SH10', 19, 7, 14, 50, 20, 75, 10, 45),

          (2022, 'sT00002', 'GID10', '10B', 'SG10', 29, 8, 13, 49, 24, 10, 11, 50),

          (2022, 'ST00002', 'GID10', '10B', 'SS10', 17, 10, 15, 30, 20, 75, 13, 35);

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES

            (2023, 'ST00002', 'GID11', '11B', 'SA11', 17, 10, 15, 40, 10, 15, 10, 35),

            (2023, 'ST00002', 'GID11', '11B', 'SE11', 16, 9, 12, 50, 70, 15, 10, 35),

            (2023, 'ST00002', 'GID11', '11B', 'SM11', 15, 7, 14, 45, 70, 25, 10, 45),

            (2023, 'ST00002', 'GID11', '11B', 'SB11', 22, 6, 13, 46, 70, 25, 10, 45),

            (2023, 'ST00002', 'GID11', '11B', 'SP11', 21, 5, 15, 47, 70, 21, 10, 35),

            (2023, 'ST00002', 'GID11', '11B', 'SCI11', 25, 10, 15, 47, 24, 75, 10, 25),

            (2023, 'ST00002', 'GID11', '11B', 'SCH11', 24, 9, 12, 37, 22, 75, 10, 35),

            (2023, 'ST00002', 'GID11', '11B', 'SH11', 23, 7, 14, 50, 20, 75, 10, 45),

            (2023, 'sT00002', 'GID11', '11B', 'SG11', 21, 8, 13, 49, 24, 10, 11, 50),

            (2023, 'ST00002', 'GID11', '11B', 'SS11', 25, 10, 15, 30, 20, 75, 13, 35);

INSERT INTO Result.Roaster (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50)

VALUES

        (2023, 'ST00003', 'GID9', '9A', 'SA9', 14, 10, 15, 40, 10, 15, 10, 35),

        (2023, 'ST00003', 'GID9', '9A', 'SE9', 15, 9, 12, 50, 70, 15, 10, 35),

        (2023, 'ST00003', 'GID9', '9A', 'SM9', 15, 7, 14, 45, 70, 25, 10, 45),

        (2023, 'ST00003', 'GID9', '9A', 'SB9', 11, 6, 13, 46, 70, 25, 10, 45),

        (2023, 'ST00003', 'GID9', '9A', 'SP9', 13, 5, 15, 47, 70, 21, 10, 35),

        (2023, 'ST00003', 'GID9', '9A', 'SCI9', 13, 10, 15, 47, 24, 75, 10, 25),

        (2023, 'ST00003', 'GID9', '9A', 'SCH9', 12, 9, 12, 37, 22, 75, 10, 35),

        (2023, 'ST00003', 'GID9', '9A', 'SH9', 11, 7, 14, 50, 20, 75, 10, 45),

        (2023, 'sT00003', 'GID9', '9A', 'SG9', 13, 8, 13, 49, 24, 10, 11, 50),

        (2023, 'ST00003', 'GID9', '9A', 'SS9', 14, 10, 15, 30, 20, 75, 13, 35),

        (2022, 'ST00004', 'GID9', '9A', 'SA9', 14, 10, 15, 40, 10, 15, 10, 35),

        (2022, 'ST00004', 'GID9', '9A', 'SE9', 15, 9, 12, 50, 70, 15, 10, 35),

        (2022, 'ST00004', 'GID9', '9A', 'SM9', 15, 7, 14, 45, 70, 25, 10, 45),

        (2022, 'ST00004', 'GID9', '9A', 'SB9', 11, 6, 13, 46, 70, 25, 10, 45),

        (2022, 'ST00004', 'GID9', '9A', 'SP9', 13, 5, 15, 47, 70, 21, 10, 35),

        (2022, 'ST00004', 'GID9', '9A', 'SCI9', 13, 10, 15, 47, 24, 75, 10, 25),

        (2022, 'ST00004', 'GID9', '9A', 'SCH9', 12, 9, 12, 37, 22, 75, 10, 35),

        (2022, 'ST00004', 'GID9', '9A', 'SH9', 11, 7, 14, 50, 20, 75, 10, 45),

        (2022, 'sT00004', 'GID9', '9A', 'SG9', 13, 8, 13, 49, 24, 10, 11, 50),

        (2022, 'ST00004', 'GID9', '9A', 'SS9', 14, 10, 15, 30, 20, 75, 13, 35),

          (2023, 'ST00004', 'GID1O', '1OA', 'SA10', 14, 10, 15, 40, 10, 15, 10, 35),

          (2023, 'ST00004', 'GID1O', '1OA', 'SE10', 15, 9, 12, 50, 70, 15, 10, 35),

          (2023, 'ST00004', 'GID1O', '1OA', 'SM10', 15, 7, 14, 45, 70, 25, 10, 45),

          (2023, 'ST00004', 'GID1O', '1OA', 'SB10', 11, 6, 13, 46, 70, 25, 10, 45),

          (2023, 'ST00004', 'GID1O', '1OA', 'SP10', 13, 5, 15, 47, 70, 21, 10, 35),

          (2023, 'ST00004', 'GID1O', '1OA', 'SCI10', 13, 10, 15, 47, 24, 75, 10, 25),

          (2023, 'ST00004', 'GID1O', '1OA', 'SCH10', 12, 9, 12, 37, 22, 75, 10, 35),

          (2023, 'ST00004', 'GID1O', '1OA', 'SH10', 11, 7, 14, 50, 20, 75, 10, 45),

          (2023, 'sT00004', 'GID1O', '1OA', 'SG10', 13, 8, 13, 49, 24, 10, 11, 50),

          (2023, 'ST00004', 'GID1O', '1OA', 'SS10', 14, 10, 15, 30, 20, 75, 13, 35),

        (2023, 'ST00005', 'GID9', '9B', 'SA9', 24, 10, 15, 40, 10, 15, 10, 35),

        (2023, 'ST00005', 'GID9', '9B', 'SE9', 15, 9, 12, 50, 70, 15, 10, 35),

        (2023, 'ST00005', 'GID9', '9B', 'SM9', 25, 7, 14, 45, 70, 25, 10, 45),

        (2023, 'ST00005', 'GID9', '9B', 'SB9', 11, 6, 13, 46, 70, 25, 10, 45),

        (2023, 'ST00005', 'GID9', '9B', 'SP9', 23, 5, 15, 47, 70, 21, 10, 35),

        (2023, 'ST00005', 'GID9', '9B', 'SCI9', 23, 10, 15, 47, 24, 75, 10, 25),

        (2023, 'ST00005', 'GID9', '9B', 'SCH9', 22, 9, 12, 37, 22, 75, 10, 35),

        (2023, 'ST00005', 'GID9', '9B', 'SH9', 11, 7, 14, 50, 20, 75, 10, 45),

        (2023, 'sT00005', 'GID9', '9B', 'SG9', 13, 8, 13, 49, 24, 10, 11, 50),

        (2023, 'ST00005', 'GID9', '9B', 'SS9', 14, 10, 15, 30, 20, 75, 13, 35),

        (2020, 'ST00006', 'GID9', '9A', 'SA9', 24, 10, 15, 40, 10, 15, 10, 35),

        (2020, 'ST00006', 'GID9', '9A', 'SE9', 15, 9, 12, 50, 70, 15, 10, 35),

        (2020, 'ST00006', 'GID9', '9A', 'SM9', 25, 7, 14, 45, 70, 25, 10, 45),

        (2020, 'ST00006', 'GID9', '9A', 'SB9', 11, 6, 13, 46, 70, 25, 10, 45),

        (2020, 'ST00006', 'GID9', '9A', 'SP9', 23, 5, 15, 47, 70, 21, 10, 35),

        (2020, 'ST00006', 'GID9', '9A', 'SCI9', 13, 10, 15, 47, 24, 75, 10, 25),

        (2020, 'ST00006', 'GID9', '9A', 'SCH9', 12, 9, 12, 37, 22, 75, 10, 35),

        (2020, 'ST00006', 'GID9', '9A', 'SH9', 11, 7, 14, 50, 20, 75, 10, 45),

        (2020, 'sT00006', 'GID9', '9A', 'SG9', 13, 8, 13, 49, 24, 10, 11, 50),

        (2020, 'ST00006', 'GID9', '9A', 'SS9', 14, 10, 15, 30, 20, 75, 13, 35),

          (2021, 'ST00006', 'GID10', '10B', 'SA10', 24, 10, 15, 40, 10, 15, 10, 35),

          (2021, 'ST00006', 'GID10', '10B', 'SE10', 5, 9, 12, 50, 70, 15, 10, 35),

          (2021, 'ST00006', 'GID10', '10B', 'SM10', 25, 7, 14, 45, 70, 25, 10, 45),

          (2021, 'ST00006', 'GID10', '10B', 'SB10', 1, 6, 13, 46, 70, 25, 10, 45),

          (2021, 'ST00006', 'GID10', '10B', 'SP10', 23, 5, 15, 47, 70, 21, 10, 35),

          (2021, 'ST00006', 'GID10', '10B', 'SCI10',13, 10, 15, 47, 24, 75, 10, 25),

          (2021, 'ST00006', 'GID10', '10B', 'SCH10', 12, 9, 12, 37, 22, 75, 10, 35),

          (2021, 'ST00006', 'GID10', '10B', 'SH10', 21, 7, 14, 50, 20, 75, 10, 45),

          (2021, 'sT00006', 'GID10', '10B', 'SG10', 13, 8, 13, 49, 24, 10, 11, 50),

          (2021, 'ST00006', 'GID10', '10B', 'SS10', 24, 10, 15, 30, 20, 75, 13, 35),

            (2022, 'ST00006', 'GID11', '11B', 'SA11', 24, 10, 15, 40, 10, 15, 10, 35),

            (2022, 'ST00006', 'GID11', '11B', 'SE11', 15, 9, 12, 50, 70, 15, 10, 35),

            (2022, 'ST00006', 'GID11', '11B', 'SM11', 25, 7, 14, 45, 70, 25, 10, 45),

            (2022, 'ST00006', 'GID11', '11B', 'SB11', 11, 6, 13, 46, 70, 25, 10, 45),

            (2022, 'ST00006', 'GID11', '11B', 'SP11', 23, 5, 15, 47, 70, 21, 10, 35),

            (2022, 'ST00006', 'GID11', '11B', 'SCI11', 23, 10, 15, 47, 24, 75, 10, 25),

            (2022, 'ST00006', 'GID11', '11B', 'SCH11', 12, 9, 12, 37, 22, 75, 10, 35),

            (2022, 'ST00006', 'GID11', '11B', 'SH11', 12, 7, 14, 50, 20, 75, 10, 45),

            (2022, 'sT00006', 'GID11', '11B', 'SG11', 13, 8, 13, 49, 24, 10, 11, 50),

            (2022, 'ST00006', 'GID11', '11B', 'SS11', 12, 10, 15, 30, 20, 75, 13, 35),

              (2023, 'ST00006', 'GID12', '12B', 'SA12', 14, 10, 15, 40, 10, 15, 10, 35),

              (2023, 'ST00006', 'GID12', '12B', 'SE12', 15, 9, 12, 50, 70, 15, 10, 35),

              (2023, 'ST00006', 'GID12', '12B', 'SM12', 24, 7, 14, 45, 70, 25, 10, 45),

              (2023, 'ST00006', 'GID12', '12B', 'SB12', 11, 6, 13, 46, 70, 25, 10, 45),

              (2023, 'ST00006', 'GID12', '12B', 'SP12', 24, 5, 15, 47, 70, 21, 10, 35),

              (2023, 'ST00006', 'GID12', '12B', 'SCI12', 13, 10, 15, 47, 24, 75, 10, 25),

              (2023, 'ST00006', 'GID12', '12B', 'SCH12', 12, 9, 12, 37, 22, 75, 10, 35),

              (2023, 'ST00006', 'GID12', '12B', 'SH12', 21, 7, 14, 50, 20, 75, 10, 45),

              (2023, 'sT00006', 'GID12', '12B', 'SG12', 23, 8, 13, 49, 24, 10, 11, 50),

              (2023, 'ST00006', 'GID12', '12B', 'SS12', 24, 10, 15, 30, 20, 75, 13, 35);

-- check

-- select \* from Result.Roaster

-- for Table

INSERT INTO Teacher\_data.Teacher (Teacher\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Degree\_level, Sub\_city, Kebele)

VALUES ('T001', 'Abdi', 'Mohammed', 'Alemu', 'Male', '1980-01-01', 'Degree', 'Facilo', 'Kebele 4'),

       ('T002', 'Roman', 'Bealu', 'Girma', 'Female', '1985-03-15', 'Master', 'Tana', 'Kebele 13'),

       ('T003', 'Yesewku', 'Ewunetu', 'Manyazewal', 'Male', '1975-05-20', 'Degree', 'Gish Abay', 'Kebele 3'),

       ('T004', 'Jemila', 'Gedefaw', 'Smachew', 'Female', '1978-07-30', 'Degree', 'Belay Zeleke', 'Kebele 4'),

       ('T005', 'Yeab', 'Neged', 'Eleizier', 'Male', '1970-09-12', 'Master', 'Dagmawi Menelek', 'Kebele 10'),

       ('T006', 'Tangut', 'Kasa', 'Gondere', 'Female', '1973-11-22', 'Degree', 'Atse Tewodros', 'Kebele 14');

  -- check

  -- SELECT \* FROM Teacher\_data.Teacher

-- for Table

INSERT INTO Teacher\_data.Teacher\_phone (Teacher\_ID, Phone\_number)

VALUES ('T001', '+251 989 143 563'),

       ('T001', '+251 989 143 564'),

       ('T001', '+251 989 143 565'),

       ('T002', '+251 989 143 566'),

       ('T002', '+251 989 143 567'),

       ('T003', '+251 989 143 568'),

       ('T003', '+251 989 143 569'),

       ('T004', '+251 989 143 572'),

       ('T004', '+251 989 143 571'),

       ('T005', '+251 989 143 578'),

       ('T006', '+251 989 143 576'),

       ('T006', '+251 989 143 573');

  -- check

  -- SELECT \* FROM Teacher\_data.Teacher\_phone

-- for Table

INSERT INTO Assignment.Teacher\_Grade\_Level (Teacher\_ID, Grade\_level\_ID)

VALUES  ('T001', 'GID9'),

        ('T002', 'GID10'),

        ('T003', 'GID11'),

        ('T004', 'GID12'),

        ('T005', 'GID11'),

        ('T006', 'GID12');

  -- check

  -- SELECT \* FROM Assignment.Teacher\_Grade\_Level

-- for Table

INSERT INTO Assignment.Teacher\_section (Teacher\_ID, Section\_code)

VALUES  ('T001', '9A'),

        ('T002', '10B'),

        ('T003', '11A'),

        ('T004', '12B'),

        ('T005', '11A'),

        ('T006', '12A');

  -- check

  -- SELECT \* FROM Assignment.Teacher\_section

-- for Table

INSERT INTO Assignment.Teacher\_Section\_Subject (Teacher\_ID, Section\_code, Subject\_code)

VALUES  ('T001', '9A', 'SP9'),

        ('T002', '10B', 'SM10'),

        ('T003', '11A', 'SB11'),

        ('T004', '12A', 'SM12'),

        ('T005', '9A', 'SA9'),

        ('T005', '10B', 'SG10'),

        ('T006', '11A', 'SCH11'),

        ('T006', '12A', 'SP12');

 -- check

 -- SELECT \* FROM Teacher\_Section\_Subject

-- for Table

INSERT INTO Shumabo.Academic\_calendar (Calendar\_ID, Ac\_Year, Activity\_name, Activity\_description, Activity\_start\_date, Activity\_end\_date, Comment)

VALUES ('C2023', 2023, 'Cleaning Day', 'by this program almost all parts of the school are going to be cleaned by both students and staff members', '2023-02-01', '2023-02-05', 'Commete, which constain 3 members, should me constructed ')

 -- check

 -- SELECT \* FROM Shumabo.Academic\_calendar

-- for Table

INSERT INTO Resource.New\_item (ISer\_no, Item\_name, Item\_type, Unit\_price, Total\_price, Recieving\_date, Added\_quantity, Item\_resource, Comment)

VALUES ('IT1112', 'Grade 12 Physices Teacher guede book', 'I\_Alaki', 59.50, 5950, '2022-09-12', 100, 'From Amhara Education Bureau', 'there is no any budgent withdrwal for them.');

 -- check

 -- SELECT \* FROM Resource.New\_item

-- for table

INSERT INTO Resource.Withdrew\_item (ISer\_no, Withdrawed\_quantity, Withdrawing\_date, Withdrawing\_reason, Withdrawer\_name)

VALUES ('IT1112', '45', '2022-09-14', 'teachers need a updated version', 'Tr. Mohammed Alemu')

 -- cheack

 -- SELECT \* FROM Resource.Withdrew\_item

-- for Table

INSERT INTO Staff\_data.Staff (Staff\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Degree\_level, Staff\_role,Sub\_city, Kebele)

VALUES ('SF001', 'Abdi', 'Mohammed', 'Alemu', 'Male', '1980-01-01', 'Degree', 'Teacher', 'Facilo', 'Kebele 4'),

       ('SF002', 'Roman', 'Bealu', 'Girma', 'Female', '1985-03-15', 'Master', 'Unit-leader', 'Tana', 'Kebele 13'),

       ('SF003', 'Mesafint', 'Ewunetu', 'Manyazewal', 'Male', '1975-05-20', 'Director', 'Degree', 'Gish Abay', 'Kebele 3'),

       ('SF004', 'Jemila', 'Gedefaw', 'Smachew', 'Female', '1978-07-30', 'Degree', 'Teacher and Chairman', 'Belay Zeleke', 'Kebele 4'),

       ('SF005', 'Ras', 'Neged', 'Eleizier', 'Male', '1970-09-12', 'Master', 'Librarian', 'Dagmawi Menelek', 'Kebele 10'),

       ('SF006', 'Tangut', 'Kasa', 'Gondere', 'Female', '1973-11-22', 'Degree', 'Lab assistant', 'Atse Tewodros', 'Kebele 14');

  -- check

  -- SELECT \* FROM Staff\_data.staff

-- for Table

INSERT INTO Staff\_data.Staff\_phone (Staff\_ID, Phone\_number)

VALUES ('SF001', '+251 918 567 893'),

       ('SF001', '+251 918 567 894'),

       ('SF001', '+251 918 567 895'),

       ('SF002', '+251 918 567 896'),

       ('SF002', '+251 918 567 897'),

       ('SF003', '+251 918 567 898'),

       ('SF003', '+251 918 567 899'),

       ('SF004', '+251 918 567 892'),

       ('SF004', '+251 918 567 891'),

       ('SF005', '+251 918 567 810'),

       ('SF006', '+251 918 567 811'),

       ('SF006', '+251 918 567 812');

  -- check

  -- SELECT \* FROM Staff\_data.Staff\_phone

-- **Setting up Database Security**

--Login 1: Create System\_Admin login

    CREATE LOGIN System\_Admin WITH PASSWORD = 'sysadmin1234';

--Login 2: Create Director login

    CREATE LOGIN Director WITH PASSWORD = 'director1234';

--Login 3: Create Teacher login

    CREATE LOGIN Teacher WITH PASSWORD = 'teacher1234';

--Login 4: Create Staff login

    CREATE LOGIN Staffs WITH PASSWORD = 'staff1234';

--Login 5: Create Resource Officer login

    CREATE LOGIN Resource\_officer WITH PASSWORD = 'RO1234';

--Login 6: Create Registrar login

    CREATE LOGIN Registrar WITH PASSWORD = 'registrar1234';

--Login 7: create Unit\_leader login

    CREATE LOGIN Unit\_leader WITH  PASSWORD = 'UL1234'

-- Login 8: Create a Security\_admin login

CREATE Login Security\_Admin WITH PASSWORD = 'secadmin1234'

-- **Grant permissions for users based on the schema they own**

USE Shumabo\_secondary\_school

-- Grant permissions for users based on the schema they own

 -- For user Mesafint | login Director

  -- Grant CREATE TABLE permission to Mesafint on Shumabo schema

    -- grant creating

        GO

        GRANT CREATE TABLE

        TO Mesafint

    -- Grant selecting tables from schema dbo

        GO

        GRANT SELECT

        ON schema::[dbo]

        TO [Mesafint];

    -- Grant referencing permission

        GO

        GRANT REFERENCES

        ON SCHEMA::[dbo]

        TO [AlemGezi];

        GO

        GRANT REFERENCES

        ON SCHEMA::[Assignment]

        TO [Mesafint];

    -- grant execution permision

        GO

        GRANT EXECUTE TO Mesafint

    -- Grant Create Procedre to Mesafint on Shumabo schema

        GO

        Grant CREATE PROCEDURE

        TO Mesafint

    -- Grant CREATE FUNCTION permission to Mesafint on Shumabo schema

        GO

        GRANT CREATE FUNCTION

        TO Mesafint

    -- Grant CREATE TRIGGER permission to Mesafint on Shumabo schema

 -- For user AlemGezi | user for login Registrar

  -- Grant CREATE TABLE permission to AlemGezi on Stud\_data schema

    -- grant creating

        GO

        GRANT CREATE TABLE

        TO AlemGezi

    -- grant selecting tables from schema dbo

        GO

        GRANT SELECT

        ON schema::[dbo]

        TO [AlemGezi];

    -- grant referencing schema Assignment

        GO

        GRANT REFERENCES

        ON schema::[Assignment]

        TO [AlemGezi];

    -- grant references permission

        GO

        GRANT REFERENCES

        ON SCHEMA::[dbo]

        TO [AlemGezi];

        GO

        GRANT REFERENCES

        ON SCHEMA::[Assignment]

        TO [AlemGezi];

    -- Grant EXECUTE permission to AlemGezi on Stud\_data schema

    GO

        GRANT EXECUTE TO AlemGezi

    -- Grant Create Procedre function to AlemGezi on Stud\_data schema

    GO

        Grant CREATE PROCEDURE

        TO AlemGezi

    -- Grant CREATE VIEW permission

        GO

        GRANT CREATE VIEW

        TO AlemGezi

    -- Grant CREATE FUNCTION permission to AlemGezi on Stud\_data schema

        GO

        GRANT CREATE FUNCTION

        TO AlemGezi

    -- Grant CREATE TRIGGER permission to AlemGezi on Stud\_data schema

    -- Grant alter permission

    GO

        GRANT ALTER

        ON SCHEMA::[dbo]

        TO [AlemGezi];

 -- For user Abiy | user for login Unit\_leader

    -- grant creating

    GO

    GRANT CREATE TABLE

    TO Abiy

    -- grant selecting tables from schema dbo

    GRANT SELECT

    ON schema::[dbo]

    TO [Abiy];

    -- grant a create view permission

    GRANT CREATE VIEW TO Abiy;

    -- Grant executing permission

    GRANT EXECUTE TO Abiy

    -- Grant Create Procedre function to Abiy on Assignment schema

    GO

    Grant CREATE PROCEDURE

    TO Abiy

    -- Grant CREATE FUNCTION permission to Abiy on Assignment schema

    GO

    GRANT CREATE FUNCTION

    TO Abiy

    -- Grant reference permission on schema Teacher\_Data

    GRANT REFERENCES

    ON SCHEMA::Teacher\_Data

    TO Abiy

-- For user Zemenay | user for login Resource\_Officer

 -- Grant CREATE TABLE permission to Zemenay on Assignment schema

    -- grant creating table

        GO

        GRANT CREATE TABLE

        TO Zemenay

    -- grant creating schema

        GO

        GRANT CREATE SCHEMA

        TO Zemenay

    -- grant selecting tables from schema dbo

    GRANT SELECT

    ON schema::[dbo]

    TO [Zemenay];

    -- Grant EXECUTE permission to Abiy on Assignment schema

        GO

        GRANT EXECUTE

        TO Zemenay

    -- Grant Create Procedre function to Abiy on Assignment schema

        GO

        Grant CREATE PROCEDURE

        TO Zemenay

    -- Grant CREATE FUNCTION permission to Abiy on Assignment schema

        GO

        GRANT CREATE FUNCTION

        TO Zemenay

-- **Creating Views**

-- list of Views

-- view 1

-- full information of Teachers

GO

    CREATE VIEW Teacher\_data.Teacher\_data AS

    SELECT  Teacher\_data.Teacher.Teacher\_ID,

            Teacher\_data.Teacher.F\_name,

            Teacher\_data.Teacher.L\_name,

            Teacher\_data.Teacher.M\_name,

            Teacher\_data.Teacher.Gender,

            Teacher\_data.Teacher.Birth\_date,

            Teacher\_data.Teacher.Degree\_level,

            Teacher\_data.Teacher.Sub\_city,

            Teacher\_data.Teacher.Kebele,

            Teacher\_data.Teacher\_phone.Phone\_number,

            Assignment.Teacher\_Grade\_Level.Grade\_level\_ID,

            Assignment.Teacher\_section\_subject.Section\_code,

            Assignment.Teacher\_section\_subject.Subject\_code

    FROM Teacher\_data.Teacher

    LEFT JOIN Assignment.Teacher\_Grade\_Level ON Teacher\_data.Teacher.Teacher\_ID = Assignment.Teacher\_Grade\_Level.Teacher\_ID

    LEFT JOIN Assignment.Teacher\_section\_subject ON Teacher\_data.Teacher.Teacher\_ID = Assignment.Teacher\_section\_subject.Teacher\_ID

    LEFT JOIN Teacher\_phone ON Teacher\_data.Teacher.Teacher\_ID = Teacher\_phone.Teacher\_ID

GO

-- view 2

-- list of subjects and there Teachers

GO

    CREATE VIEW Assignment.Subject\_Teacher AS

    SELECT  Assignment.Subject.Subject\_code AS Subject\_code,

            Assignment.Subject.Subject\_name AS Subject\_name,

            Assignment.Teacher\_section\_subject.Section\_code AS Section\_code,

            Assignment.Subject.Grade\_level\_ID As Grade\_level\_Id,

            Assignment.Teacher\_section\_subject.Teacher\_ID AS Teacher\_ID,

            Teacher\_data.Teacher.F\_name AS Teacher\_name

    FROM Assignment.Subject

    LEFT JOIN Assignment.Teacher\_section\_subject ON Assignment.Subject.Subject\_code = Assignment.Teacher\_section\_subject.Subject\_code

    LEFT JOIN Teacher\_data.Teacher ON Assignment.Teacher\_section\_subject.Teacher\_ID = Teacher\_data.Teacher.Teacher\_ID

GO

-- view 3

-- Submitted roasters

GO

    CREATE VIEW Stud\_data.Submittable\_roaster AS

    SELECT  Stud\_data.Roaster.Ac\_year,

            Stud\_data.Roaster.Stud\_ID,

            Stud\_data.Roaster.Grade\_level\_ID,

            Stud\_data.Roaster.Section\_code,

            Stud\_data.Roaster.Subject\_code,

            (Stud\_data.Roaster.Frist\_sem\_mid\_exam\_25 + Stud\_data.Roaster.First\_sem\_quiz\_10 + Stud\_data.Roaster.First\_sem\_assignment\_15 + Stud\_data.Roaster.First\_sem\_final\_exam\_50) AS First\_sem\_Sum,

            (Stud\_data.Roaster.Second\_sem\_mid\_exam\_25 + Stud\_data.Roaster.Second\_sem\_quiz\_10 + Stud\_data.Roaster.Second\_sem\_assignment\_15 + Stud\_data.Roaster.Second\_sem\_final\_exam\_50) AS Second\_sem\_Sum

    FROM Stud\_data.Roaster

GO

SELECT \* FROM Stud\_data.Submittable\_roaster

-- view 4

-- Current status of Items

GO

    CREATE VIEW Resource.Current\_Item\_Status AS

    SELECT  Resource.All\_item.ISer\_no,

            Resource.All\_item.Item\_name,

            Resource.All\_item.Item\_type,

            Resource.All\_item.Unit\_price,

            Resource.New\_item.Recieving\_date,

            Resource.New\_item.Added\_quantity,

            (Resource.New\_item.Added\_quantity - Resource.All\_item.Current\_quantity) AS Withdrew\_quantity,

            Resource.All\_item.Current\_quantity,

            Resource.All\_item.Total\_price

    FROM Resource.All\_item

    LEFT JOIN Resource.New\_item ON Resource.All\_item.ISer\_no = Resource.New\_item.ISer\_no

    LEFT JOIN Resource.Withdrew\_Item ON Resource.All\_item.ISer\_no = Resource.Withdrew\_Item.ISer\_no;

GO

-- view 5

-- containing the full information of the grade\_levels

GO

    CREATE VIEW  Assignment.Grade\_level\_data AS

    SELECT  Stud\_data.Roaster.Ac\_year,

            Assignment.Grade\_level.Grade\_level\_ID,

            Assignment.Grade\_level.Grade\_level\_name,

            Assignment.Section.Section\_Code,

            Stud\_data.Student.Stud\_ID,

            Stud\_data.Student.Gender

    FROM Stud\_data.Roaster

    LEFT JOIN Assignment.Grade\_level ON Stud\_data.Roaster.Grade\_level\_ID = Assignment.Grade\_level.Grade\_level\_ID

    LEFT JOIN Assignment.Section ON Assignment.Grade\_level.Grade\_level\_ID = Assignment.Section.Grade\_level\_ID

    LEFT JOIN Stud\_data.Student ON Assignment.Section.Section\_code = Stud\_data.Student.Section\_code

GO

-- view 6

-- store the detail information of class schedule for each section

GO

    CREATE VIEW Assignment.Class\_schedule\_detail AS

    SELECT

            Assignment.Class\_Schedule.Schedule\_ID,

            Assignment.Class\_Schedule.Ac\_year,

            Assignment.Class\_Schedule.Section\_code,

            Assignment.Class\_Schedule.Grade\_level\_ID,

            Assignment.Grade\_level.Grade\_level\_name,

            Assignment.Class\_Schedule.period\_no,

            Assignment.Class\_Schedule.Subject\_code,

            Assignment.Subject.Subject\_name,

            Assignment.Class\_Schedule.Teacher\_ID,

            Teacher\_data.Teacher.F\_name as Teacher\_F\_name,

            Assignment.Class\_Schedule.Day\_name,

            Assignment.Class\_Schedule.Start\_time,

            Assignment.Class\_Schedule.End\_time

    FROM Assignment.Class\_Schedule

    LEFT JOIN Assignment.Grade\_level ON Assignment.Class\_Schedule.Grade\_level\_ID = Assignment.Grade\_level.Grade\_level\_ID

    LEFT JOIN Assignment.Subject ON Assignment.Class\_Schedule.Subject\_code = Assignment.Subject.Subject\_code

    LEFT JOIN Teacher\_data.Teacher ON Assignment.Class\_Schedule.Teacher\_ID = Teacher\_data.Teacher.Teacher\_ID

GO

-- view 7

-- formalized Report card from each student

GO

    CREATE VIEW Stud\_data.Report\_card AS

    SELECT  Stud\_data.Transcript.Ac\_year,

            Stud\_data.Transcript.Stud\_ID,

            Stud\_data.Transcript.Grade\_level\_ID,

            Stud\_data.Student.F\_name,

            Stud\_data.Student.L\_name,

            Stud\_data.Student.M\_name,

            Stud\_data.Student.Gender,

            Stud\_data.Student.Age,

            Stud\_data.Transcript.First\_semester\_avg,

            Stud\_data.Transcript.Second\_semester\_avg,

            Stud\_data.Transcript.Final\_avg,

            Stud\_data.Transcript.Conduct

    FROM Assignment.Stud\_data.Transcript

    JOIN Stud\_data.Student ON Stud\_data.Transcript.Stud\_ID = Stud\_data.Student.Stud\_ID

GO

-- **Operating on data or list of Functionalities**

-- Functionality 1: Add new Parent { with Stored procedure}

GO

  CREATE PROCEDURE Stud\_data.Add\_parent (

    @PID\_no *VARCHAR*(15),

    @F\_Name *VARCHAR*(50),

    @L\_Name *VARCHAR*(50),

    @M\_Name *VARCHAR*(50),

    @Gender *VARCHAR*(10),

    @Birth\_date *DATE*,

    @Relation *VARCHAR*(255),

    @Sub\_city *VARCHAR*(255),

    @Kebele *VARCHAR*(255)

  )

  AS

  BEGIN

    INSERT INTO Stud\_data.Parent (PID\_no, F\_name, L\_name, M\_name, Gender, Birth\_date, Relation, Sub\_city, Kebele)

    VALUES (@PID\_no, @F\_Name, @L\_Name, @M\_name, @Gender, @Birth\_date, @Relation, @Sub\_city, @Kebele);

  END

GO

EXEC Stud\_data.Add\_parent 'BDR007', 'Chernet', 'Abegaz', 'Sewale', 'Male', '2000-01-01', 'Father', 'Gish Abay', 'Kebele 3';

EXEC Stud\_data.Add\_parent 'BDR008', 'Tsega', 'Abera', 'Wakene', 'Male', '1988-01-01', 'Father', 'Atse Tewodros', 'Kebele 3';

-- Functionality 2: Display data of specific parent

GO

  CREATE FUNCTION Get\_parent (@PID\_no *VARCHAR*(15))

  RETURNS TABLE

  AS

  RETURN

  (

    SELECT \*

    FROM Stud\_data.Parent

    WHERE PID\_no = @PID\_no

  );

GO

SELECT \* FROM dbo.Get\_Parent('BDR007')

-- Functionality 3: Update specific Information of a specified Parent

GO

  CREATE PROCEDURE Stud\_data.Update\_parent (@PID\_no *VARCHAR*(10), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Stud\_data.Parent SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE PID\_no = ' + '''' + @PID\_no + ''''

    EXECUTE sp\_executesql @SQL

  END

GO

-- Explaination

    /\* In the above code, NVARCHAR(MAX) is used as the data type

    for the input parameter @Attribute\_name in the stored procedure.

    NVARCHAR(MAX) is a variable-length Unicode string data type

    that can store up to 2^30-1 characters.

    This data type is used to accommodate a wide range of values for

    the input parameter and handle updates for different data types.

    The purpose of using NVARCHAR(MAX) in the stored procedure

    is to allow the input value to be passed as a string,

    and then converted to the correct data type for the targeted

    column in the Stud\_data.Parent table.

    sp\_executesql is a system stored procedure in SQL Server that

    is used to execute a dynamically constructed SQL statement or batch.

    The procedure takes parameters for the dynamic SQL string and any

    parameters that need to be passed to the dynamic statement, and

    returns the result set of the executed statement. This is useful in

    situations where the exact SQL statement to be executed is not known

    at design time, or when the statement needs to be constructed based on

    user input or other dynamic data.

    The SQL\_VARIANT data type is a special data type in SQL Server that

    can store values of different data types, including:-

     - Numeric data types (int, float, decimal, etc.)

     - Character strings (char, varchar, nchar, nvarchar)

     - Binary data (binary, varbinary)

     - Date and time (date, time, datetime, datetime2, datetimeoffset)

    The purpose of using the SQL\_VARIANT data type is to provide a flexible

    data type that can store multiple data types in a single column or variable.

    This allows for more efficient storage and manipulation of data, as well as

    more flexible querying and reporting. However, it is important to note that

    the SQL\_VARIANT data type also has some limitations, such as less efficient

    data storage and lower performance compared to using specific data types.

    \*/

EXEC Stud\_data.Update\_parent 'BDR007', 'Sub\_city', 'Shumabo'

EXEC Stud\_data.Update\_parent 'BDR001', 'Birth\_date', '1981-01-01'

EXEC Stud\_data.Update\_parent 'BDR001', 'PID\_no', 'BDR000'  -- cannot update the PID\_no of any parent since its concerns with referential integrity.

-- Functionality 4: Register new student

GO

  CREATE PROCEDURE Stud\_data.Add\_new\_student (

    @Ac\_Year *INT*,

    @Registration\_data *DATE*,

    @Stud\_ID *VARCHAR*(10),

    @F\_Name *VARCHAR*(50),

    @L\_Name *VARCHAR*(50),

    @M\_Name *VARCHAR*(50),

    @Gender *VARCHAR*(6),

    @Birth\_date *DATE*,

    @Sub\_city *VARCHAR*(50),

    @Kebele *VARCHAR*(50),

    @PID\_no *VARCHAR*(15)

  )

  AS

  BEGIN

    INSERT INTO Stud\_data.New\_Student (Ac\_year, Registration\_date, Stud\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Sub\_city, Kebele, PID\_no)

    VALUES(@Ac\_Year, @Registration\_data, @Stud\_ID, @F\_Name, @L\_Name, @M\_name, @Gender, @Birth\_date, @Sub\_city, @Kebele, @PID\_no);

  END

GO

EXEC Stud\_data.Add\_new\_student 2023, '2023-08-03', 'ST0008', 'Alemayehu', 'Chernet', 'Abegaz', 'Male', '2000-01-01', 'Shumabo', 'Kebele 3', 'BDR007' ;

EXEC Stud\_data.Add\_new\_student 2023, '2023-08-03', 'ST0009', 'Chaltu', 'Tsega', 'Abera', 'female', '2000-02-03', 'Atse Tewodros', 'Kebele 3', 'BDR008' ;

-- Functionality 5: Update a specified information of specified student

GO

  CREATE PROCEDURE Stud\_data.Update\_Student (@Stud\_ID *VARCHAR*(15), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Stud\_data.Student SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Stud\_ID = ' + '''' + @Stud\_ID + ''''

    EXECUTE sp\_executesql @SQL

  END

GO

EXEC Stud\_data.Update\_Student 'ST0009', 'Grade\_level\_Id', 'GID11'

EXEC Stud\_data.Update\_Student 'ST0009', 'Section\_code', '11A'

-- Functionality 6: Display a specified Student information // with stored procedure

-- without phone number

GO

  CREATE PROCEDURE Display\_student (@Stud\_ID *VARCHAR*(10))

  AS

  BEGIN

    SELECT \*

    FROM Stud\_data.Student

    WHERE Stud\_ID = @Stud\_ID

  END

GO

-- with phone number

GO

 CREATE PROCEDURE show\_Student (@Stud\_ID *VARCHAR*(10))

 AS

 BEGIN

  SELECT

    s.Stud\_ID,

    s.F\_name,

    s.L\_name,

    s.M\_name,

    s.Gender,

    s.Birth\_date,

    s.Sub\_city,

    s.Kebele,

    s.PID\_no,

    s.Grade\_level\_ID,

    s.Section\_code,

    sp.phone\_number

  FROM

    Stud\_data.Student s

  INNER JOIN

    Stud\_data.Student\_phone sp ON s.Stud\_ID = sp.Stud\_ID

  WHERE

    s.Stud\_ID = @Stud\_ID

 END

GO

EXEC show\_Student 'ST00001'  -- With there phone number

EXEC Display\_student 'ST0008'  -- without phone number

Select \* FROM Stud\_data.Student\_phone

-- Functionality 7: Add new subject

GO

  CREATE PROCEDURE Assignment.Add\_Subject (@Subject\_code *VARCHAR*(6), @Subject\_name *VARCHAR*(50), @Grade\_level\_ID *VARCHAR*(6))

  AS

  BEGIN

    INSERT INTO Assignment.Subject (Subject\_code, Subject\_name, Grade\_level\_ID)

    VALUES (@Subject\_code, @Subject\_name, @Grade\_level\_ID);

  END

GO

EXEC Assignment.Add\_Subject 'STD11', 'Technical Drawing', 'GID11'

-- functionlity 8: display data about specific subject

GO

  CREATE PROCEDURE Get\_Subject (@Subject\_code *VARCHAR*(6))

  AS

  BEGIN

    SELECT \* FROM

    Assignment.Subject\_Teacher where Subject\_code = @Subject\_code

  END

GO

EXEC Get\_Subject 'STD11'

-- Fun tionality 9: Update specific information of specific Subject

GO

  CREATE PROCEDURE Assignment.Update\_Subject (@Subject\_code *VARCHAR*(6), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Assignment.Subject SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Subject\_code = ' + '''' + @Subject\_code + ''''

    EXECUTE sp\_executesql @SQL

  END

GO

EXEC Assignment.Update\_Subject 'STD11', 'Subject\_name', 'Drawing'

-- Functionlity 10: Delete/remove a specific Subject from the list

GO

  CREATE PROCEDURE Assignment.Delete\_Subject (@Subject\_code *VARCHAR*(6))

  AS

  BEGIN

    DELETE FROM Assignment.Subject

    WHERE Subject\_code = @Subject\_code;

  END

GO

EXEC Assignment.Delete\_Subject 'STD11'

select \* from Assignment.Subject

-- functionality 11: Add student result

GO

  CREATE PROCEDURE Stud\_data.Add\_Roaster(

    @Ac\_year FLOAT,

    @Stud\_ID *VARCHAR*(10),

    @Grade\_level\_ID *VARCHAR*(6),

    @Section\_code *VARCHAR*(6),

    @Subject\_code *VARCHAR*(6),

    @Frist\_sem\_mid\_exam\_25 FLOAT,

    @First\_sem\_quiz\_10 FLOAT,

    @First\_sem\_assignment\_15 FLOAT,

    @First\_sem\_final\_exam\_50 FLOAT,

    @Second\_sem\_mid\_exam\_25 FLOAT,

    @Second\_sem\_quiz\_10 FLOAT,

    @Second\_sem\_assignment\_15 FLOAT,

    @Second\_sem\_final\_exam\_50 FLOAT )

  AS

  BEGIN

  if (@Frist\_sem\_mid\_exam\_25 > 25)

    BEGIN

      RA

    END

    INSERT INTO Stud\_data.Roaster(Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Subject\_code, Frist\_sem\_mid\_exam\_25, First\_sem\_quiz\_10, First\_sem\_assignment\_15, First\_sem\_final\_exam\_50, Second\_sem\_mid\_exam\_25, Second\_sem\_quiz\_10, Second\_sem\_assignment\_15, Second\_sem\_final\_exam\_50 )

    VALUES (@Ac\_year, @Stud\_ID, @Grade\_level\_ID, @Section\_code, @Subject\_code, @Frist\_sem\_mid\_exam\_25, @First\_sem\_quiz\_10, @First\_sem\_assignment\_15, @First\_sem\_final\_exam\_50, @Second\_sem\_mid\_exam\_25, @Second\_sem\_quiz\_10, @Second\_sem\_assignment\_15, @Second\_sem\_final\_exam\_50)

  END

GO

EXEC Stud\_data.Add\_Roaster '2023','ST00004', 'GID10', '10A', 'SA10', 24, 10, 15, 44, 23, 9, 13, 50

-- functionality 12: display specific student's result

GO

    CREATE PROCEDURE Display\_Roaster(@Stud\_ID *VARCHAR*(10), @Ac\_year *INT*)

    AS

    BEGIN

        SELECT \* FROM Stud\_data.Roaster

        WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_year

    END

GO

EXEC Display\_Roaster 'ST00004', 2023

-- Functionlity 13: update specific information of specific student

GO

    CREATE PROCEDURE Stud\_data.Update\_Roaster (@Ac\_year *INT*, @Stud\_ID *VARCHAR*(10), @Subject\_code *VARCHAR*(6), @attribute\_name NVARCHAR(MAX), @New\_value SQL\_VARIANT)

    AS

    BEGIN

        DECLARE @SQL NVARCHAR(MAX)

        SET @SQL = N'UPDATE Stud\_data.Roaster

            SET ' + @attribute\_name + ' = ' + CAST(@New\_value AS NVARCHAR(MAX)) + '

            WHERE Ac\_year = ' + CAST(@Ac\_year AS NVARCHAR(10)) + '

            AND Stud\_ID = ''' + @Stud\_ID + '''

            AND Subject\_code = ''' + @Subject\_code + ''''

        EXEC sp\_executesql @SQL

    END

GO

EXEC Stud\_data.Update\_Roaster 2022,'ST00006' , 'SCH11', 'Frist\_sem\_mid\_exam\_25', 24

-- Functionality 14: generating a Transcript for all students

GO

  CREATE PROCEDURE Stud\_data.Generate\_Transcript (@Stud\_ID *VARCHAR*(15), @Ac\_Year *INT*, @Conduct *CHAR*(1))

  AS

  BEGIN

    DECLARE @Grade\_level\_ID *VARCHAR*(6),

            @Section\_code *VARCHAR*(6),

            @First\_semester\_total FLOAT,

            @Second\_semester\_total FLOAT,

            @First\_semester\_avg FLOAT,

            @Second\_semester\_avg FLOAT,

            @Final\_avg FLOAT,

            @Number\_of\_Subjects *INT*

    SET @Grade\_level\_ID = (SELECT TOP 1 @Grade\_level\_ID FROM Stud\_data.Submittable\_roaster WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_Year)

    SET @Section\_code = (SELECT TOP 1 @Section\_code FROM Stud\_data.Submittable\_roaster WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_Year)

    SET @First\_semester\_total = (SELECT SUM(First\_sem\_sum) From Stud\_data.Submittable\_roaster WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_Year)

    SET @Second\_semester\_total = (SELECT SUM(Second\_sem\_sum) From Stud\_data.Submittable\_roaster WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_Year)

    SET @Number\_of\_Subjects = (select COUNT(Subject\_code) From Stud\_data.Submittable\_roaster WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_Year)

    SET @First\_semester\_avg = ((@First\_semester\_total/ @Number\_of\_Subjects))

    SET @Second\_semester\_avg = ((@Second\_semester\_total/ @Number\_of\_Subjects))

    SET @Final\_avg = (@First\_semester\_avg + @Second\_semester\_avg) / 2;

    INSERT INTO Stud\_data.Transcript (Ac\_Year, Stud\_ID, Grade\_level\_ID, Section\_Code,First\_semester\_total, Second\_semester\_total, First\_semester\_avg, Second\_semester\_avg, Final\_avg, Conduct)

    VALUES (@Ac\_Year, @Stud\_ID, @Grade\_level\_ID, @Section\_code, @First\_semester\_total, @Second\_semester\_total, @First\_semester\_avg, @Second\_semester\_total, @Final\_avg, @Conduct)

    select \* from  Stud\_data.Transcript

  END

GO

EXEC Stud\_data.Generate\_Transcript 'ST00001', '2022', 'A'

EXEC Stud\_data.Generate\_Transcript 'ST00004', '2023', 'A'

-- Functionality 15: updating the Transcript of a specific student when there is a change  of his/her in the roaster\_list

GO

  CREATE PROCEDURE Stud\_data.Update\_Transcript (@Stud\_ID *VARCHAR*(15), @Ac\_year *INT*, @attribute\_name NVARCHAR(50), @New\_value SQL\_VARIANT)

  AS

  BEGIN

      SET NOCOUNT ON;

      DECLARE @SQL NVARCHAR(MAX)

      SET @SQL =  N'UPDATE Stud\_data.Transcript

                  SET ' + @attribute\_name + ' = ' + CAST(@New\_value AS NVARCHAR(MAX)) + '

                  WHERE Ac\_year = ' + CAST(@Ac\_year AS NVARCHAR(10)) + '

                  AND Stud\_ID = ''' + @Stud\_ID + ''''

      EXECUTE sp\_executesql @SQL

      SELECT \* FROM Stud\_data.Transcript WHERE Stud\_ID  = @Stud\_ID AND Ac\_Year = @Ac\_Year

  END

GO

EXEC Stud\_data.Update\_Transcript '','',''

-- Functionality 16: display the Transcript of a specific Student

GO

    CREATE FUNCTION Stud\_data.Display\_Transcript(@Stud\_ID *VARCHAR*(10), @Ac\_year *INT*)

    RETURNS TABLE

    AS

    RETURN (

      SELECT \* FROM Stud\_data.Transcript

        WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_year

    )

GO

SELECT \* FROM Stud\_data.Display\_Transcript('ST00001', '2022')

-- Functionality 17: create the transform form for specific student

GO

  CREATE PROCEDURE Stud\_data.ADD\_Transform\_form(

    @TSer\_no *VARCHAR*(10),

    @Stud\_ID *varchar*(10),

    @Grade\_level\_ID *VARCHAR*(6),

    @Section\_code *VARCHAR*(6),

    @Reason *TEXT*,

    @Application\_date *DATE*,

    @Target\_school\_name *VARCHAR*(100))

  AS

  BEGIN

    INSERT INTO Stud\_data.Transform\_form(TSer\_no, Stud\_ID, Grade\_level\_ID, Section\_code , Reason, Application\_date, Target\_school\_name)

    VALUES (@TSer\_no, @Stud\_ID, @Grade\_level\_ID , @Section\_code, @Reason, @Application\_date, @Target\_school\_name)

  END

GO

EXEC Stud\_data.ADD\_Transform\_form 'TF1234', 'ST00001', 'GID11', '11A', 'Changing address', '2023-03-04', 'Belay Zeleke Secondary school'

-- functionlity 18: display/provide the transform form of specific studnet

GO

  CREATE PROCEDURE Stud\_data.Dispaly\_Transform\_form ( @Stud\_ID *VARCHAR*(10))

  AS

  BEGIN

    SELECT \* FROM Stud\_data.Transform\_form

    WHERE Stud\_ID = @Stud\_ID

  END

GO

EXEC Stud\_data.Dispaly\_Transform\_form 'ST00001'

-- Functionality 19: Add Teacher { with Stored procedure}

GO

  CREATE PROCEDURE Teacher\_data.Add\_Teacher (

    @Teacher\_ID *VARCHAR*(15),

    @F\_Name *VARCHAR*(50),

    @L\_Name *VARCHAR*(50),

    @M\_Name *VARCHAR*(50),

    @Gender *VARCHAR*(10),

    @Birth\_date *DATE*,

    @Degree\_level *VARCHAR*(255),

    @Sub\_city *VARCHAR*(255),

    @Kebele *VARCHAR*(255)

  )

  AS

  BEGIN

    INSERT INTO Teacher\_data.Teacher (Teacher\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Degree\_level, Sub\_city, Kebele)

    VALUES (@Teacher\_ID, @F\_Name, @L\_Name, @M\_name, @Gender, @Birth\_date, @Degree\_level, @Sub\_city, @Kebele);

  END

GO

EXEC Teacher\_data.Add\_Teacher 'T007', 'Bewketu', 'Sewmehon', 'gash Takele', 'Male', '1980-01-01', 'Master', 'Gish Abay', 'Kebele 3';

-- Functionality 20: Display a specific teacher's information with Table valued function

GO

  CREATE FUNCTION Get\_Teacher (@Teacher\_ID *VARCHAR*(15))

  RETURNS TABLE

  AS

  RETURN

  (

    SELECT \*

    FROM Teacher\_data.Teacher

    WHERE Teacher\_ID = @Teacher\_ID

  );

GO

SELECT \* FROM dbo.Get\_Teacher('T007')

-- Functionality 21: Update specific Information of a specified Teacher

GO

  CREATE PROCEDURE Teacher\_data.Update\_Teacher (@Teacher\_ID *VARCHAR*(10), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Teacher\_data.Teacher SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Teacher\_ID = ' + '''' + @Teacher\_ID + ''''

    EXECUTE sp\_executesql @SQL

  END

GO

EXEC Teacher\_data.Update\_Teacher 'T007', 'M\_name', 'Takele'

-- Assigning teachers to specific Grade level , displaying the assignment and updating the assignment

-- Functionlity 22: Assignign a particular teacher to specific Grade\_level

GO

  CREATE PROCEDURE Assignment.Assign\_Teacher\_to\_Grade (@Teacher\_ID *VARCHAR*(10), @Grade\_Level\_ID *VARCHAR*(6))

  AS

  BEGIN

    INSERT INTO Assignment.Teacher\_Grade\_Level(Teacher\_ID, Grade\_level\_ID)

    VALUES (@Teacher\_ID, @Grade\_Level\_ID)

  END

GO

EXEC Assign\_Teacher 'T007', 'GID10'

-- functionality 23: Displaying

GO

  CREATE FUNCTION Display\_Assigne\_Teacher (@Teacher\_ID *VARCHAR*(15))

  RETURNS TABLE

  AS

  RETURN (

    SELECT \* FROM Assignment.Teacher\_Grade\_Level

    WHERE Teacher\_ID = @Teacher\_ID

  )

GO

select \* from dbo.Display\_Assigne\_Teacher ('T007')

-- Functionality 24: Updating assignment of teachers to the grade level

GO

  CREATE PROCEDURE Assignment.Update\_Grade\_Assignment(@Teacher\_ID *VARCHAR*(10), @New\_Grade\_level\_ID *VARCHAR*(10))

  AS

  BEGIN

    UPDATE Assignment.Teacher\_Grade\_Level

    SET Grade\_level\_ID = @New\_Grade\_level\_ID

    WHERE Teacher\_ID = @Teacher\_ID

    PRINT 'You Update an Information Successfully!'

  END

GO

EXEC Assignment.Update\_Grade\_Assignment 'T007', 'GID10'

-- Assigning teachers to specific Section and subject

-- functionality 25: Assigning a teacher to a Section and subject

GO

  CREATE PROCEDURE Assignment.Assign\_Teacher\_to\_Sec\_Sub (@Teacher\_ID *VARCHAR*(10), @Section\_code *VARCHAR*(6), @Subject\_code *VARCHAR*(6))

  AS

  BEGIN

    INSERT INTO  Assignment.Teacher\_Section\_Subject(Teacher\_ID, Section\_code, Subject\_code)

    VALUES (@Teacher\_ID, @Section\_code, @Subject\_code)

  END

GO

EXEC Assignment.Assign\_Teacher\_to\_Sec\_Sub 'T007', '10A', 'SA10'

-- functionality 26: Displaying

GO

  CREATE FUNCTION Display\_Assigned\_Teacher\_to\_Sec\_Sub (@Teacher\_ID *VARCHAR*(15))

  RETURNS TABLE

  AS

  RETURN (

    SELECT \* FROM

    WHERE Teacher\_ID = @Teacher\_ID

  )

GO

select \* from dbo.Display\_Assigned\_Teacher\_to\_Sec\_Sub ('T007')

-- Functionality 27: Updating assignment of teachers to the grade level

GO

  CREATE PROCEDURE Assignment.Update\_Sec\_Sub\_Assignment (@Teacher\_ID *VARCHAR*(10), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Assignment.Teacher\_Section\_subject SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Teacher\_ID = ' + '''' + @Teacher\_ID + ''''

    EXECUTE sp\_executesql @SQL

    PRINT 'You Update an Information Successfully!'

  END

GO

EXEC Assignment.Update\_Sec\_Sub\_Assignment 'T007', 'Subject\_code', 'SE10'

-- creating, displaying, updating and Deleting an Academic Calendar

-- functionality 28: Creating an academic Calendar

GO

  CREATE PROCEDURE Shumabo.Add\_Calendar (@Calendar\_ID *VARCHAR*(6), @Ac\_Year *INT*, @Activity\_name *VARCHAR*(255), @Activity\_description *TEXT*, @Activity\_start\_date *DATE*, @Activity\_end\_date *DATE*, @Comment *TEXT*)

  AS

  BEGIN

    SET NOCOUNT ON;

    INSERT INTO Shumabo.Academic\_Calendar (Calendar\_ID, Ac\_Year, Activity\_name, Activity\_description, Activity\_start\_date, Activity\_end\_date, Comment)

    VALUES (@Calendar\_ID, @Ac\_Year, @Activity\_name, @Activity\_description, @Activity\_start\_date, @Activity\_end\_date, @Comment)

  END

GO

EXEC shumabo.Add\_Calendar 'Cal2023', '2023', 'Semester Break', 'since we all have ...', '2023-02-20', '2023-03-04', 'no more than 14 days'

EXEC Shumabo.Add\_Calendar 'C2023b', '2023', 'Studnets Movement', 'this will be the movement of ...', '2023-02-20', '2023-02-21', 'teacheing learning process will be cloase for only 1 day'

-- functionality 29: Updating a created calendar

GO

  CREATE PROCEDURE Shumabo.Update\_Calendar (@Calendar\_ID *VARCHAR*(6), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Shumabo.Academic\_calendar SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Calendar\_ID = ' + '''' + @Calendar\_ID + ''''

    EXECUTE sp\_executesql @SQL

  END

GO

EXEC Shumabo.Update\_Calendar 'Cal202', 'Ac\_Year', '2020'

-- Functionality 30: display the details of specific Activity in the calendar

GO

  CREATE FUNCTION Get\_Activity (@Calendar\_ID *VARCHAR*(6))

  RETURNS TABLE

  AS

  RETURN

  (

    SELECT \* FROM Shumabo.Academic\_Calendar

    WHERE Calendar\_ID = @Calendar\_ID

  )

GO

SELECT \* FROM dbo.Get\_Activity('Cal202')

-- Functionality 31: display the all activities and thir details which are planned for specific year

GO

  CREATE FUNCTION Get\_Yearly\_Calendar (@Ac\_Year *INT*)

  RETURNS TABLE

  AS

  RETURN

  (

    SELECT \* FROM Shumabo.Academic\_Calendar

    WHERE Ac\_Year = @Ac\_Year

  )

GO

SELECT \* FROM dbo.Get\_Yearly\_Calendar('2023')

SELECT \* FROM dbo.Get\_Yearly\_Calendar('2020')

-- Functionlity 32: Removing a certain year calender from the Calendar list

GO

  CREATE PROCEDURE Shumabo.Delete\_Yearly\_Calendar (@Ac\_Year *INT*)

  AS

  BEGIN

    SET NOCOUNT ON;

    DELETE FROM Shumabo.Academic\_Calendar WHERE Ac\_Year = @Ac\_Year

    PRINT 'You have Removed Calendar of  one year'

  END

GO

EXEC Shumabo.Delete\_Yearly\_Calendar '2020'

-- Manipulating the Staff members

-- Functionlity 33: Add Data of new staff members like director, Record\_Officer, ...

GO

  CREATE PROCEDURE Staff\_data.Add\_New\_Staff(

  @Staff\_ID *VARCHAR*(10),

  @F\_name *VARCHAR*(50),

  @L\_name *VARCHAR*(50),

  @M\_name *VARCHAR*(50),

  @Gender *VARCHAR*(8),

  @Birth\_date *DATE*,

  @Degree\_level *VARCHAR*(50),

  @Staff\_role *VARCHAR*(50),

  @Sub\_city *VARCHAR*(50),

  @Kebele *VARCHAR*(50))

  AS

  BEGIN

    INSERT INTO Staff(Staff\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Degree\_level, Staff\_role, Sub\_city, Kebele)

    VALUES (@Staff\_ID, @F\_name, @L\_name, @M\_name, @Gender, @Birth\_date, @Degree\_level, @Staff\_role, @Sub\_city, @Kebele)

    PRINT 'You have added new Staff member Successfully.'

  END

GO

EXEC Staff\_data.Add\_New\_Staff 'STA002','R-memihr', 'Mesafint', 'Tefera','Male', '1980-04-12', 'Master', 'Director', 'Gish Abay', 'Kebele 5'

-- Functionality 34: Display the Data of specific Staff member

GO

  CREATE FUNCTION Get\_Staff (@Staff\_ID *VARCHAR*(10))

  RETURNS TABLE

  AS

    RETURN (

      SELECT \* FROM Staff

      WHERE Staff\_ID = @Staff\_ID

    )

GO

SELECT \* FROM dbo.Get\_Staff ('STA002')

-- Functionality 35: Update the specific data of a specific Staff member

GO

  CREATE PROCEDURE Staff\_data.Update\_staff (@Staff\_ID *VARCHAR*(10), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Staff\_data.Staff SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Staff\_ID = ' + '''' + @Staff\_ID + ''''

    EXECUTE sp\_executesql @SQL

    PRINT 'You have updated the data successfully, and the data of the Update Staff is:-'

    SELECT \* FROM Staff\_data.Staff WHERE Staff\_ID = @Staff\_ID

  END

GO

EXEC Staff\_data.Update\_staff 'STA002', 'F\_name', 'Director'

-- Functionality 36: Add new Items

GO

  CREATE PROCEDURE Resourse.Add\_Item(@ISer\_no *VARCHAR*(8), @Item\_name *VARCHAR*(50), @Item\_type *varchar*(50), @Unit\_price FLOAT, @Total\_price FLOAT, @Recieving\_date *Date*, @Added\_quantity *INT*, @Item\_resource *TEXT*, @Comment *TEXT*)

  AS

  BEGIN

    INSERT INTO Resource.New\_item

    VALUES (@ISer\_no, @Item\_name, @Item\_type, @Unit\_price, @Total\_price, @Recieving\_date, @Added\_quantity, @Item\_resource, @Comment)

    PRINT 'You have added new item.'

    INSERT INTO Resource.All\_item

    VALUES (@ISer\_no, @Item\_name, @Item\_type, @Unit\_price, @Total\_price, @Added\_quantity)

  END

GO

EXEC Resource.Add\_Item 'IT1111', 'Grade 10 Amharic text Book', 'Alaki', '40.50', '4050', '2023-03-02', '100', 'gift from BDU University', 'No Maney is invested'

EXEC Resource.Add\_Item 'IT1113', 'Board Cleaner', 'Alaki', '15', '1500', '2023-05-02', '100', 'Bought', 'Maney is from the yearly collested fee'

-- Functionality 37:  Update the data of the available Item in the store

GO

  CREATE PROCEDURE Resource.Update\_Item (@ISer\_no *VARCHAR*(8), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    DECLARE @SQLB NVARCHAR(Max)

    SET @SQL = N'UPDATE Resource.New\_item SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE ISer\_no = ' + '''' + @ISer\_no + ''''

    EXECUTE sp\_executesql @SQL

    PRINT 'You have updated the data successfully, and the data of the Update Staff is:-'

    SELECT \* FROM Resource.New\_item WHERE ISer\_no = @ISer\_no

    IF (@Attribute\_name = 'Item\_name' OR @Attribute\_name = 'Item\_type' OR @Attribute\_name = 'Unit\_price' OR @Attribute\_name = 'Total\_price' OR @attribute\_name = 'Added\_quantity')

    BEGIN

      SET @SQLB = N'UPDATE Resource.All\_item SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE ISer\_no = ' + '''' + @ISer\_no + ''''

      EXECUTE sp\_executesql @SQLB

    END

  END

GO

DROP PROCEDURE Resource.Update\_Item

EXEC Resource.Update\_Item 'IT1111', 'Item\_type', 'I\_Alaki'

EXEC Resource.Update\_Item 'I20231', 'Item\_type', 'I\_Alaki'

exec Resource.Update\_Item 'I20231', 'Item\_name', 'Grade 11 English text book.'

EXEC Resource.Update\_Item 'I20231', 'ISer\_no', 'IT1112'  --updates the Serial no of the Item

-- Manipullating table withdrawed Itmes list

-- Functionlity 38: withdrowing items

GO

  CREATE PROCEDURE Resource.Withdraw\_Item(@ISer\_no *VARCHAR*(8), @withdrawing\_date *Date*, @Withdrawed\_quantity *INT*, @Withdrawing\_reason *TEXT*, @Withdrawer\_name *VARCHAR*(100))

  AS

  BEGIN

    INSERT INTO Resource.Withdrew\_item

    VALUES (@ISer\_no, @Withdrawing\_date, @Withdrawed\_quantity, @Withdrawing\_reason, @Withdrawer\_name)

    PRINT 'You have Withdrawed an item.'

    -- update the available amount

    UPDATE Resource.All\_item

    SET Current\_quantity = Current\_quantity - @Withdrawed\_quantity where ISer\_no = @ISer\_no

    -- update the total price of the available amount

    UPDATE Resource.All\_item

    SET Total\_price = Unit\_price \* Current\_quantity WHERE ISer\_no = @ISer\_no

  END

GO

EXEC Resource.Withdraw\_Item 'IT1113', '2023-02-02', 13, 'Class room Education', 'Tr. chernet'

EXEC Resource.Withdraw\_Item 'IT1111', '2022-02-02', 2, 'losting the priviously given one', 'St. Jemila smachew'

-- Functinality 39: display the withdrawing Items in specific year, month, week or days

GO

  CREATE FUNCTION Get\_Withdrawed\_Item (@Date\_one *DATE*, @Date\_two *DATE*)

  RETURNS TABLE

  AS

  RETURN (

    SELECT \* FROM Resource.Withdrew\_item

    WHERE Withdrawing\_date BETWEEN @Date\_one AND  @Date\_two

  )

GO

SELECT \* FROM dbo.Get\_Withdrawed\_Item('2022-01-01', '2022-12-30')

-- Manipulating all available items in All\_Item

-- Functinality 40: display the Current status of an Item

GO

  CREATE FUNCTION Resource.Get\_Item (@ISer\_no *VARCHAR*(10))

  RETURNS TABLE

  AS

  RETURN (

    SELECT \* FROM Resource.Current\_Item\_Status

    WHERE ISer\_no = @ISer\_no

  )

GO

SELECT \* FROM Resource.Get\_Item('IT1111')

-- Generating, displaying and updating yearly Progress reports

-- Functionality 41: generating progress level

GO

  CREATE PROCEDURE Shumabo.Generate\_Progress\_report (@Ac\_Year *INT*, @Grade\_level\_ID *VARCHAR*(6))

  AS

  BEGIN

    DECLARE @Number\_of\_section *INT*,

            @Number\_of\_student *INT*,

            @Number\_of\_male\_stud *INT*,

            @Number\_of\_female\_stud *INT*,

            @Number\_of\_passed\_stud *INT*,

            @Number\_of\_failed\_stud *INT*,

            @Passed\_stud\_percentage FLOAT,

            @Number\_of\_passed\_male\_stud *int*,

            @Passed\_male\_percentage FLOAT,

            @Number\_of\_passed\_female\_stud *INT*,

            @Passed\_female\_percentage FLOAT,

            @Failed\_stud\_percentage FLOAT,

            @Number\_of\_failed\_female\_stud *INT*,

            @Number\_of\_failed\_male\_stud *int*,

            @Failed\_male\_percentage FLOAT,

            @Failed\_female\_percentage FLOAT,

            @Max\_avg FLOAT,

            @Min\_avg FLOAT,

            @Avg\_range FLOAT

      SET @Number\_of\_section = (SELECT COUNT(DISTINCT Section\_code) FROM Grade\_level\_data WHERE Grade\_level\_ID = @Grade\_level\_ID AND Ac\_year = @Ac\_year )

      SET @Number\_of\_student = (SELECT COUNT(DISTINCT Stud\_ID) FROM Grade\_level\_data WHERE Grade\_level\_ID = @Grade\_level\_ID AND Ac\_year = @Ac\_year )

      SET @Number\_of\_male\_stud = (SELECT COUNT(DISTINCT Stud\_ID) FROM Grade\_level\_data WHERE Grade\_level\_data.Gender = 'male' AND Grade\_level\_data.Grade\_level\_ID = @Grade\_level\_ID AND Grade\_level\_data.Ac\_year = @Ac\_year )

      SET @Number\_of\_female\_stud = (@Number\_of\_student - @Number\_of\_male\_stud)

      SET @Number\_of\_passed\_stud = (SELECT COUNT(Stud\_status) FROM Stud\_data.Pass\_fail\_student WHERE Stud\_status = 'passed' AND Stud\_data.Pass\_fail\_student.Grade\_level\_ID = @Grade\_level\_ID AND Stud\_data.Pass\_fail\_student.Ac\_year = @Ac\_Year)

      SET @Number\_of\_failed\_stud = (SELECT COUNT(Stud\_status) FROM Stud\_data.Pass\_fail\_student WHERE Stud\_status = 'failed' AND Stud\_data.Pass\_fail\_student.Grade\_level\_ID = @Grade\_level\_ID AND Stud\_data.Pass\_fail\_student.Ac\_year = @Ac\_Year)

      SET @Passed\_stud\_percentage = (@Number\_of\_passed\_stud / (@Number\_of\_passed\_stud + @number\_of\_failed\_stud)) \* 100

      SET @Number\_of\_passed\_male\_stud = (SELECT COUNT(Stud\_status) FROM Stud\_data.Pass\_fail\_student WHERE Stud\_status = 'passed' AND Gender = 'male' AND Stud\_data.Pass\_fail\_student.Grade\_level\_ID = @Grade\_level\_ID AND Stud\_data.Pass\_fail\_student.Ac\_year = @Ac\_Year)

      SET @Passed\_male\_percentage = (@Number\_of\_passed\_male\_stud / (@Number\_of\_passed\_male\_stud + @Number\_of\_passed\_female\_stud)) \* 100

      SET @Number\_of\_passed\_female\_stud = (SELECT COUNT(Stud\_status) FROM Stud\_data.Pass\_fail\_student WHERE Stud\_status = 'passed' AND Gender = 'female' AND Stud\_data.Pass\_fail\_student.Grade\_level\_ID = @Grade\_level\_ID AND Stud\_data.Pass\_fail\_student.Ac\_year = @Ac\_Year)

      SET @Passed\_female\_percentage = (@Number\_of\_passed\_female\_stud / (@Number\_of\_passed\_female\_stud + @Number\_of\_passed\_female\_stud)) \* 100

      SET @Failed\_stud\_percentage = (@Number\_of\_passed\_stud / (@Number\_of\_passed\_stud + @number\_of\_failed\_stud)) \* 100

      SET @Number\_of\_failed\_male\_stud = (SELECT COUNT(Stud\_status) FROM Stud\_data.Pass\_fail\_student WHERE Stud\_status = 'failed' AND Gender = 'male' AND Stud\_data.Pass\_fail\_student.Grade\_level\_ID = @Grade\_level\_ID AND Stud\_data.Pass\_fail\_student.Ac\_year = @Ac\_Year)

      SET @Failed\_male\_percentage = (@Number\_of\_failed\_male\_stud / (@Number\_of\_failed\_male\_stud + @Number\_of\_failed\_female\_stud)) \* 100

      SET @Number\_of\_failed\_female\_stud = (SELECT COUNT(Stud\_status) FROM Stud\_data.Pass\_fail\_student WHERE Stud\_status = 'failed' AND Gender = 'female' AND Stud\_data.Pass\_fail\_student.Grade\_level\_ID = @Grade\_level\_ID AND Stud\_data.Pass\_fail\_student.Ac\_year = @Ac\_Year)

      SET @Failed\_female\_percentage = (@Number\_of\_passed\_female\_stud / (@Number\_of\_passed\_female\_stud + @Number\_of\_passed\_female\_stud)) \* 100

      SET @Max\_avg = (SELECT MAX(Final\_avg) FROM Transcript WHERE Grade\_level\_ID = @Grade\_level\_ID AND Ac\_year = @Ac\_year)

      SET @Min\_avg = (SELECT MIN(Final\_avg) FROM Transcript WHERE Grade\_level\_ID = @Grade\_level\_ID AND Ac\_year = @Ac\_year)

    INSERT INTO Shumabo.Progress\_report (Ac\_year, Grade\_level\_Id, Num\_of\_section, Num\_of\_stud, Num\_of\_male\_stud, Num\_of\_female\_stud, Max\_avg, Min\_avg, Passed\_stud\_percentage, Failed\_stud\_percentage, Passed\_male\_percentage, Failed\_male\_percentage, Passed\_female\_percentage, Failed\_female\_percentage)

    VALUES (@Ac\_Year, @Grade\_level\_ID, @Number\_of\_section, @Number\_of\_student, @Number\_of\_male\_stud, @Number\_of\_female\_stud, @Max\_avg, @Min\_avg, @Passed\_stud\_percentage, @Failed\_stud\_percentage, @Passed\_male\_percentage, @Failed\_male\_percentage, @Passed\_female\_percentage, @Failed\_female\_percentage)

    PRINT 'the progress report is generated successfully.'

  END

GO

EXEC Shumabo.Generate\_Progress\_report 2023, 'GID12'

-- Functionality 42: displaying Progress levels of a specific Grade\_level

GO

  CREATE FUNCTION Get\_Progress\_report (@Ac\_year *INT*, @Grade\_Level\_ID *VARCHAR*(6))

  RETURNS TABLE

  AS

    RETURN (

      SELECT \* FROM Shumabo.Progress\_report

      WHERE Ac\_year = @Ac\_year AND Grade\_level\_ID = @Grade\_Level\_ID

    )

GO

SELECT \* FROM dbo.Get\_Progress\_report(2023, 'GID12')

-- Functionality 43: removing specific registered Progress report

GO

  CREATE PROCEDURE Shumabo.Remove\_progress\_report (@Ac\_year *INT*, @Grade\_Level\_ID *VARCHAR*(6))

  AS

  BEGIN

    DELETE FROM Shumabo.Progress\_report

    WHERE Ac\_year = @Ac\_year AND Grade\_level\_ID = @Grade\_Level\_ID

  END

GO

EXEC  Shumabo.Remove\_progress\_report '2023', 'GID11'

-- Functionality 44: Recording passed and failed students (trigger 5)

-- Functionality 45: Displaying passed and failed students

GO

  CREATE FUNCTION Stud\_data.Get\_Stud\_status (@Stud\_ID *varchar*(10), @Ac\_Year *INT*)

  RETURNS TABLE

  AS

  RETURN (

    SELECT \* FROM Stud\_data.Pass\_fail\_student

    WHERE Stud\_ID =  @Stud\_ID AND @Ac\_Year = Ac\_year

  )

GO

-- operating on the non\_attendat table

-- functionality  46: withdraw a student from the lise

GO

  CREATE PROCEDURE Stud\_data.Withdraw\_student(@Ac\_Year *INT*, @Stud\_ID *VARCHAR*(10))

  AS

  BEGIN

    IF EXISTS (Select \* from Stud\_data.Student WHERE Ac\_year = @Ac\_year AND Stud\_ID = @Stud\_ID)

      BEGIN

        DELETE FROM Stud\_data.Student

        WHERE Ac\_year = @Ac\_Year AND Stud\_ID = @Stud\_ID

      END

    ELSE

      BEGIN

        RAISERROR('There is no student with you specification, please be sure about you are trying to operate on the existing student.', 16, 1)

        ROLLBACK

      END

  END

GO

EXEC Stud\_data.Withdraw\_student '2023', 'ST0009'

-- Functionality 47: Update the withdrew students | add reason and leave\_date

GO

  CREATE PROCEDURE Stud\_data.Update\_Non\_attendant (@Ac\_Year *INT*, @Stud\_ID *VARCHAR*(10), @Reason *TEXT*, @leave\_date *DATE*)

  AS

  BEGIN

    IF EXISTS (SELECT \* FROM Stud\_data.Non\_attendant WHERE Ac\_year = @Ac\_Year AND Stud\_ID = @Stud\_ID)

      BEGIN

        UPDATE Stud\_data.Non\_attendant

        SET Reason = @Reason,

            Leave\_date = @leave\_date

        WHERE Ac\_year = @Ac\_Year and Stud\_Id = @Stud\_ID

      END

    ELSE

      BEGIN

        RAISERROR('The is no one in the withdrawn list with your specification.', 16,1)

        ROLLBACK

      END

  END

GO

EXEC Stud\_data.Update\_Non\_attendant 2023, 'ST0008', 'Health case', '2023-08-07'

EXEC Stud\_data.Update\_Non\_attendant 2023, 'ST0009', 'Transfer to other school', '2023-01-23'

-- Generating, displaying and updating class schedule

-- functionality 48: store class schedule

GO

  CREATE PROCEDURE Assignment.Add\_Class\_schedule (@Schedule\_Id *VARCHAR*(10), @Ac\_year *INT*, @Section\_code *VARCHAR*(6), @Grade\_level\_ID *VARCHAR*(6), @Subject\_code *VARCHAR*(6), @Teacher\_ID *VARCHAR*(10), @Day\_name *VARCHAR*(10), @period\_no *INT*, @Start\_time *TIME*, @End\_time *TIME*)

  AS

  BEGIN

    INSERT INTO Assignment.Class\_Schedule (Schedule\_Id, Ac\_year, Section\_code, Grade\_level\_ID, Subject\_code, Teacher\_Id, Day\_name, period\_no, Start\_time, End\_time)

    VALUES (@Schedule\_Id, @Ac\_year, @Section\_code, @Grade\_level\_ID, @Subject\_code, @Teacher\_ID, @Day\_name, @period\_no, @Start\_time, @End\_time);

    PRINT 'You have added a class schedule'

  END

GO

drop PROCEDURE Assignment.Add\_Class\_schedule;

EXEC Assignment.Add\_Class\_schedule '2312AMon1', 2023, '12A', 'GID12', 'SA12', 'T001', 'Monday', 1, '02:00:00', '02:40:00'

EXEC Assignment.Add\_Class\_schedule '2312AMon2', 2023, '12A', 'GID12', 'SE12', 'T004', 'Monday', 2, '02:40:00', '03:20:00'

EXEC Assignment.Add\_Class\_schedule '2312AMon3', 2023, '12A', 'GID12', 'SCH12', 'T003', 'Monday', 3, '03:20:00', '04:00:00'

EXEC Assignment.Add\_Class\_schedule '2312AMon4', 2023, '12A', 'GID12', 'SCI12', 'T002', 'Monday', 4, '04:15:00', '04:55:00'

EXEC Assignment.Add\_Class\_schedule '2312AMon5', 2023, '12A', 'GID12', 'SP12', 'T005', 'Monday', 5, '04:55:00', '05:35:00'

EXEC Assignment.Add\_Class\_schedule '2312AMon6', 2023, '12A', 'GID12', 'SM12', 'T006', 'Monday', 6, '02:35:00', '06:15:00'

-- Functionality 49: Displaying class schedule

GO

  CREATE FUNCTION Get\_class\_schedule (@Ac\_Year *INT*, @Section\_code *VARCHAR*(6), @Day\_name *VARCHAR*(10))

  RETURNS TABLE

  AS

  RETURN (

    SELECT Period\_no, Start\_time, End\_time, Subject\_name, Teacher\_F\_name

    From Assignment.Class\_schedule\_detail

    WHERE  Ac\_year = @Ac\_Year AND Section\_code = @Section\_code AND Day\_name = @Day\_name

  )

GO

SELECT \* FROM dbo.Get\_class\_schedule(2023, '12A', 'Monday')

-- OR

SELECT \* FROM Assignment.Class\_schedule\_detail

WHERE Ac\_year = 2023 AND Grade\_level\_ID = '12A'

-- Functionality 50: Updating class schedule

GO

  CREATE PROCEDURE Assignment.Update\_class\_schedule (@Schedule\_ID *VARCHAR*(10), @Attribute\_name NVARCHAR(50), @New\_value sql\_variant)

  AS

  BEGIN

    SET NOCOUNT ON;

    DECLARE @SQL NVARCHAR(MAX)

    SET @SQL = N'UPDATE Class\_schedule SET ' + @Attribute\_name + ' = ' + '''' + CAST(@New\_value AS NVARCHAR(MAX)) + '''' + ' WHERE Schedule\_Id = ' + '''' + @Schedule\_ID + ''''

    EXECUTE sp\_executesql @SQL

    PRINT 'You have updated the class schedule successfully.'

  END

GO

EXEC Assignment.Update\_class\_schedule '2312AMon1', 'Subject\_code', 'SB12'

-- functionlaity 51: Generate Report card (view 7)

-- functionlaity 52: display the report card of a specified student

GO

  CREATE FUNCTION Stud\_data.Get\_report\_card(@Ac\_year *INT*, @Stud\_ID *VARCHAR*(10))

  RETURNS TABLE

  AS

  RETURN (

    select \* FROM Stud\_data.Report\_card

    WHERE Ac\_year = @Ac\_year AND Stud\_ID = @Stud\_ID

  )

GO

SELECT \* FROM dbo.Get\_report\_card(2022, 'ST00003')

--**Triggers**

-- DML Triggers

-- Trigger 1

-- Functionlity : donot allow to update the Serial number of items

GO

  CREATE TRIGGER No\_update\_SerNo\_TotalPrice

  ON Resource.New\_item

  for UPDATE

  AS

  IF(UPDATE(ISer\_no) OR UPDATE(Total\_price))

    BEGIN

        RAISERROR('The Serial number of any Items should not be modified',14,12)

        ROLLBACK TRANSACTION

    END

GO

-- Triggers that update the a tabel as soon as othe table is updated

-- Trigger 2

-- Functionlality  : insert New items into All\_Item ist or Update items as new itmes are added.

GO

    CREATE TRIGGER tr\_update\_all\_items\_list

    ON Resource.New\_item

    AFTER INSERT

    AS

    BEGIN

        DECLARE

            @ISer\_no *VARCHAR*(8),

            @Item\_name *VARCHAR*(50),

            @Item\_type *varchar*(50),

            @Unit\_price FLOAT,

            @Total\_price FLOAT,

            @Added\_quantity *INT*,

            @Recieving\_date *DATE*

        SELECT

            @ISer\_no = ISer\_no,

            @Item\_name = Item\_name,

            @Item\_type = Item\_type,

            @Unit\_price = Unit\_price,

            @Total\_price = Total\_price,

            @Added\_quantity = Added\_quantity,

            @Recieving\_date = Recieving\_date

        FROM inserted

        IF NOT EXISTS (SELECT \* FROM Resource.All\_item WHERE ISer\_no = @ISer\_no)  -- if the added items not exist in the All\_Item table

            BEGIN

                INSERT INTO Resource.All\_item (ISer\_no, Item\_name, Item\_type, Unit\_price, Total\_price, Current\_quantity)

                VALUES (@ISer\_no, @Item\_name, @Item\_type, @Unit\_price, @Total\_price, @Added\_quantity)

            END

        ELSE

            BEGIN

                UPDATE Resource.All\_item

                SET Current\_quantity = Current\_quantity + @Added\_quantity

                WHERE ISer\_no = @ISer\_no

            END

    END

GO

-- Trigger 3

-- Functionality  : update the the current amount of a certain item as some of it is withdrawed.

GO

    CREATE TRIGGER tr\_update\_all\_items\_as\_withdrawed

    ON Resource.Withdrawed\_Item

    AFTER INSERT

    AS

    BEGIN

        DECLARE

            @ISer\_no *VARCHAR*(8),

            @Withdrawed\_quantity *INT*,

            @Withdrawing\_date *DATE*

        SELECT

            @ISer\_no = ISer\_no,

            @Withdrawed\_quantity = Withdrawed\_quantity,

            @Withdrawing\_date = Withdrawing\_date

        FROM inserted

        UPDATE Resource.All\_item

        SET Current\_quantity = Current\_quantity - @Withdrawed\_quantity

        WHERE ISer\_no = @ISer\_no

        UPDATE Resource.All\_item

        SET Total\_price = Unit\_price \* Current\_quantity

        WHERE ISer\_no = @ISer\_no

    END

GO

-- Trigger 4

-- add New studetns to student list

GO

    CREATE TRIGGER Stud\_data.tr\_add\_new\_student\_to\_all\_Student

    ON Stud\_data.New\_Student

    AFTER INSERT

    AS

    BEGIN

        DECLARE

            @Ac\_Year *INT*,

            @Stud\_ID *VARCHAR*(10),

            @F\_Name *VARCHAR*(50),

            @L\_Name *VARCHAR*(50),

            @M\_Name *VARCHAR*(50),

            @Gender *VARCHAR*(6),

            @Birth\_date *DATE*,

            @Sub\_city *VARCHAR*(50),

            @Kebele *VARCHAR*(50),

            @PID\_no *VARCHAR*(15)

        SELECT

            @Ac\_Year = Ac\_Year,

            @Stud\_ID = Stud\_ID,

            @F\_Name = F\_name,

            @L\_Name = L\_name,

            @M\_Name = M\_name,

            @Gender = Gender,

            @Birth\_date = Birth\_date,

            @Sub\_city = Sub\_city,

            @Kebele = Kebele,

            @PID\_no = PId\_no

        FROM inserted

        IF NOT EXISTS (SELECT \* FROM Stud\_data.Student WHERE Stud\_ID = @Stud\_ID AND Ac\_year = @Ac\_Year)

            BEGIN

                INSERT INTO Stud\_data.Student (Ac\_year, Stud\_ID, F\_name, L\_name, M\_name, Gender, Birth\_date, Sub\_city, Kebele, PID\_no)

                VALUES(@Ac\_Year, @Stud\_ID, @F\_Name, @L\_Name, @M\_name, @Gender, @Birth\_date, @Sub\_city, @Kebele, @PID\_no);

            END

    END

GO

-- Trigger

-- Functionality  : store the information of the withdrew students

GO

    CREATE TRIGGER Stud\_data.tr\_Set\_Withdraw

    ON Stud\_data.Student

    AFTER DELETE

    AS

    BEGIN

        DECLARE

            @Ac\_year *INT*,

            @Stud\_ID *VARCHAR*(10),

            @Grade\_level\_ID *VARCHAR*(10),

            @Section\_code *VARCHAR*(6),

            @Gender *VARCHAR*(10),

            @Stud\_status *VARCHAR*(20)

        SELECT

            @Ac\_year = Ac\_year,

            @Stud\_ID = Stud\_ID,

            @Grade\_level\_ID = Grade\_level\_ID,

            @Section\_code = Section\_code,

            @Gender = Gender

        FROM deleted

        SET @Stud\_status = 'Withdrew'

        INSERT INTO Stud\_data.Non\_attendant (Ac\_year, Stud\_ID, Grade\_level\_ID, Section\_code, Gender, Stud\_status)

        VALUES (@Ac\_year, @Stud\_ID, @Grade\_level\_ID, @Section\_code, @Gender, @Stud\_status)

        PRINT 'student has withdrew, fill some data about the withdrawn student'

    END

GO

-- Trigger 7

-- functionality  : store the last status (passed or failed) of students

GO

    CREATE TRIGGER Stud\_data.tr\_Set\_stud\_status

    ON Stud\_data.Transcript

    AFTER INSERT

    AS

    BEGIN

        DECLARE

            @Ac\_Year *INT*,

            @Stud\_ID *VARCHAR*(10),

            @Grade\_level\_ID *VARCHAR*(10),

            @Section\_code *VARCHAR*(6),

            @Gender *VARCHAR*(10),

            @final\_avg FLOAT,

            @Stud\_status *VARCHAR*(20)

        SELECT

            @Ac\_year = Ac\_Year,

            @Stud\_ID = Stud\_ID,

            @final\_avg = Final\_avg

        FROM inserted

        SELECT

            @Grade\_level\_ID = Grade\_level\_ID,

            @Section\_code = Section\_code,

            @Gender = Gender

        FROM Stud\_data.Student WHERE Stud\_ID = @Stud\_ID

        IF EXISTS (SELECT \* FROM Stud\_data.Pass\_fail\_student WHERE Ac\_year = @Ac\_year AND Stud\_ID = @Stud\_ID)

            BEGIN

                RAISERROR('try to set a status for the same studetn.', 12,3)

                ROLLBACK

            END

        ELSE

            BEGIN

                IF @final\_avg >= 48

                    BEGIN

                        SET @Stud\_status = 'Passed'

                        INSERT INTO Stud\_data.Pass\_fail\_student (Ac\_Year, Stud\_ID, Grade\_level\_Id, Section\_code, Gender, Final\_avg, Stud\_status)

                        VALUES (@Ac\_Year, @Stud\_ID, @Grade\_level\_ID, @Section\_code, @Gender, @final\_avg,  @Stud\_status)

                    END

                ELSE

                    BEGIN

                        SET @Stud\_status = 'Failed'

                        INSERT INTO Stud\_data.Pass\_fail\_student (Ac\_Year, Stud\_ID, Grade\_level\_Id, Section\_code, Gender, Final\_avg, Stud\_status)

                        VALUES (@Ac\_Year, @Stud\_ID, @Grade\_level\_ID, @Section\_code, @Gender, @final\_avg,  @Stud\_status)

                    END

                PRINT 'Student status is set.'

            END

    END

GO

-- Triggers that stores the deleted tables

-- Trigger 7:

-- store the deleted calanders

GO

    CREATE TRIGGER Shumabo.tr\_deleted\_Academic\_calendar

    ON Shumabo.Academic\_calendar

    AFTER DELETE

    AS

    BEGIN

        SET NOCOUNT ON;

        INSERT INTO Shumabo.Deleted\_Academic\_calendar

        SELECT \* FROM deleted;

    END;

GO

/\*

The SET NOCOUNT ON statement is used in T-SQL (Transact-SQL) to prevent

the display of the number of rows affected by a query.

The NOCOUNT option is used to suppress the message returned by the SQL Server

after each Transact-SQL statement that updates data.

The purpose of using SET NOCOUNT ON is to improve the performance of a

database by reducing the amount of data that needs to be sent between

the server and the client. When NOCOUNT is set to ON, the number of affected

 rows is not returned, reducing network traffic and increasing the performance

 of the database.

 \*/

-- Trigger 8

-- Storing the deleted Progress report

GO

    CREATE TRIGGER Shumabo.tr\_store\_deleted\_progress\_report

    ON Shumabo.Progress\_report

    AFTER DELETE

    AS

    BEGIN

        INSERT INTO Shumabo.Deleted\_progress\_report (Ac\_year, Grade\_level\_Id, Num\_of\_section, Num\_of\_stud, Num\_of\_male\_stud, Num\_of\_female\_stud, Max\_avg, Min\_avg, Passed\_stud\_percentage, Failed\_stud\_percentage, Passed\_male\_percentage, Failed\_male\_percentage, Passed\_female\_percentage, Failed\_female\_percentage)

        SELECT Ac\_year, Grade\_level\_Id, Num\_of\_section, Num\_of\_stud, Num\_of\_male\_stud, Num\_of\_female\_stud, Max\_avg, Min\_avg, Passed\_stud\_percentage, Failed\_stud\_percentage, Passed\_male\_percentage, Failed\_male\_percentage, Passed\_female\_percentage, Failed\_female\_percentage FROM deleted;

    END

GO

-- Trigger 8

-- donnot allow to insert doublcate rows in the transcript

GO

    CREATE TRIGGER Tr\_no\_duble\_Transcript

    ON Stud\_data.Transcript

    AFTER INSERT

    AS

    BEGIN

        DECLARE @Stud\_ID *VARCHAR*(10),

                @Ac\_Year *INT*;

        SELECT  @stud\_id = Stud\_ID,

                @Ac\_year = Ac\_year

        FROM inserted;

        IF EXISTS (SELECT \* FROM Stud\_data.Transcript

        WHERE Stud\_ID = @stud\_id AND Ac\_year = @Ac\_year)

            BEGIN

                RAISERROR('YOU ARE TRYING TO GENERATE DOUBLCATE ROASTER', 11,1)

                ROLLBACK

            END

    END

GO

-- DDL Triggers

-- donot update

GO

    CREATE TRIGGER tr\_No\_Table\_Change

    ON Stud\_data.Grade\_report

    INSTEAD OF UPDATE

    AS

    PRINT 'Tables should not be modified'

    ROLLBACK;

GO

-- Logon Trigers

-- **list of backup tables**

-- Table 1

-- table for storing deleted Calendars

CREATE TABLE Shumabo.Deleted\_Academic\_calendar (

    Calendar\_ID *VARCHAR*(6) NOT NULL,

    Ac\_Year *INT*,

    Activity\_name *VARCHAR*(255),

    Activity\_description *VARCHAR*(1000),

    Activity\_start\_date *DATE*,

    Activity\_end\_date *DATE*,

    Comment *VARCHAR*(1000),

    CONSTRAINT PK\_Deleted\_Calendar PRIMARY KEY(Calendar\_ID)

);

-- TAble 2

-- for storing deleted progress reports

CREATE TABLE Shumabo.Deleted\_progress\_report (

    Report\_no *INT* NOT null IDENTITY(111,3),

    Ac\_year *INT* NOT NULL,

    Grade\_level\_Id *VARCHAR*(6) not null,

    Num\_of\_section *INT*,

    Num\_of\_stud *int*,

    Num\_of\_male\_stud *INT*,

    Num\_of\_female\_stud *INT*,

    Max\_avg FLOAT,

    Min\_avg FLOAT,

    Passed\_stud\_percentage FLOAT,

    Failed\_stud\_percentage FLOAT,

    Passed\_male\_percentage FLOAT,

    Failed\_male\_percentage FLOAT,

    Passed\_female\_percentage FLOAT,

    Failed\_female\_percentage FLOAT,

);

# Bibliography

* <https://www.techprofree.com/school-management-system-project-database-design/>
* <https://cse.final-year-projects.in/>
* <https://cse.final-year-projects.in/>
* <https://www.codeproject.com/>

Full Resource code: -

* GitHub: <https://github.com/sgc93/School-Management-System>
* File: [/click here for Source code](file:///C:\Users\hp%20pavilion\Documents\GitHub\Advanced_database_system\School-Management-System\T-SQL)

Other related documents: -

* [Click to see pdf format of ER Diagram](file:///C:\Users\hp%20pavilion\Documents\GitHub\Advanced_database_system\School-Management-System\Documentation\ER_Diagram.pdf)
* [Click to see pdf format of Mapped and Normalized Relation](file:///C:\Users\hp%20pavilion\Documents\GitHub\Advanced_database_system\School-Management-System\Documentation\Normalized_Relation.pdf)
* [Click here to open Power point form of this document](file:///C:\Users\hp%20pavilion\Documents\GitHub\Advanced_database_system\School-Management-System\Power_point.pptx)