**Employee ID: SP 11252**

**Employee Name: DEEKSHITA NAIR**

**Project Name: Online Event Management System for Institutions**

**Project background:**

An online event management system for institutions is designed to simplify the process of planning, organizing and managing various events and activities within a school or a college. Such a system can be used for events like tech fest, cultural events or sports events. How the project works? There will be head coordinators for each event who will manage the entire event. The event coordinators are those who prepare events in a meaningful way and head coordinators can either approve, cancel or queue an event. All students can participate in more than one event also an event can have ‘n’ number of students. Here, too head coordinator will check whether the registration of the student is approved or cancelled. Event details table will hold venue details from venue table so as the student get proper idea about where to attend the event.

**Project Objective:**

An online event management system for institutions aims to automate and digitize the entire event management process. It provides a centralized platform for head coordinators, event coordinators and students to create, schedule, and manage events efficiently.

**Activities:**

**1. Creating Tables using Queries**

I have created 9 tables which will help to connect between events and students.

create table Coordinator

(

Co\_id int,

Co\_firstname varchar(50),

Co\_lastname varchar(50),

Contact varchar(50),

Event\_type varchar(50)

)

create table Student

(

S\_id int,

first\_name varchar(50),

middle\_name varchar(50),

last\_name varchar(50),

contact varchar(50),

city varchar(50),

Date\_of\_birth date,

Dept\_code int,

Year INT

)

create table Event\_Coordinator

(

E\_CO\_ID int,

E\_COOR\_name varchar(50),

contact varchar(50)

)

create table Student\_Event

(

S\_id int,

Event\_id int,

status\_id int

)

create table Flag

(

Flag\_id int,

Flag\_name varchar(50)

)

create table department

(

dept\_code int,

dept\_name varchar(50)

)

create table venue

(

venue\_id int,

venue\_name varchar(50),

)

create table status

(

status\_id int,

status\_name varchar(50),

)

create table Event\_Details

(

Event\_id int,

Event\_name varchar(50),

Event\_desc varchar(200),

Reg\_start\_date date,

Reg\_end\_date date,

Event\_start\_date date,

Event\_end\_date date,

Event\_start\_time time,

Event\_end\_time time,

E\_CO\_ID int,

CO\_ID int,

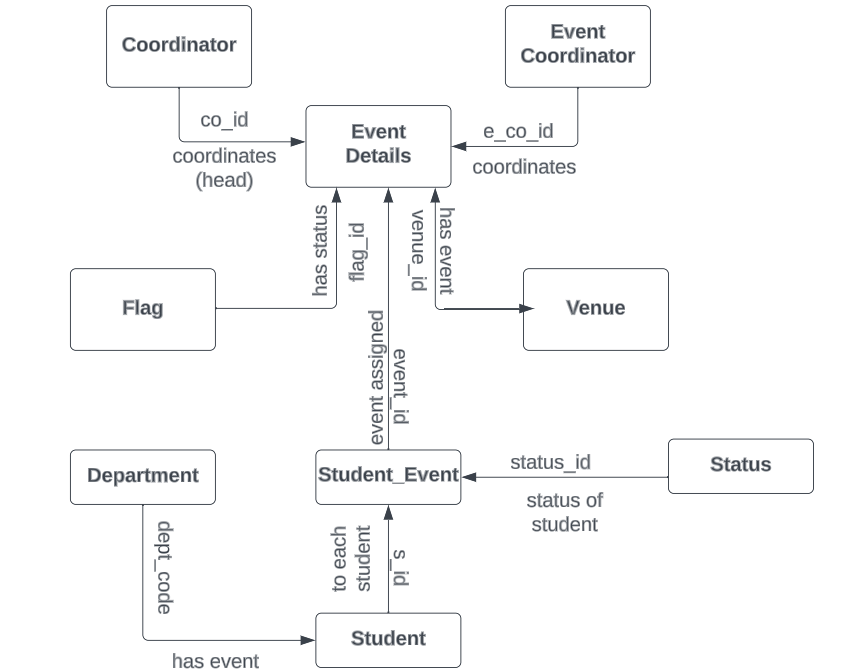
flag\_id int,

venue\_id int,

seats\_left int

)

**2.. Data Flow Diagram**



**3.. Schema / Diagram**



**4. Entering Data in tables using queries and triggers**

* INSERT INTO STUDENT (s\_id, first\_name, middle\_name, last\_name, city, contact, date\_of\_birth, dept\_code, year) VALUES (1, 'John', 'William', 'Doe', 'New York', '555-555-5555', '2000-05-15', 'CS101', 2023)
* INSERT INTO DEPARTMENT (Dept\_code, Dept\_name) VALUES ('CS101', 'Computer Science')
* INSERT INTO COORDINATOR values (1,'Reena','Subhash','45678-89786','CULTURAL')
* INSERT INTO FLAG values(1,'Approved'),(2,'Cancelled'),(3,'Pending')
* INSERT INTO EVENT\_COORDINATOR VALUES(1, 'Sarah Johnson', '555-123-4567')
* INSERT INTO event\_details VALUES(1, 'Annual Charity Gala', 'Fundraising event for local charities', '2023-04-15', '2023-04-16', '2023-03-01', '2023-04-10', '18:00:00', '22:00:00', 1, 1, 1,1,1)
* INSERT INTO STUDENT\_EVENT (s\_id, event\_id, status) VALUES(1, 1, 'approved')
* INSERT INTO VENUE values(1,'abc')
* INSERT INTO STATUS values(1,'abc')

**Now, I have used two triggers in my project.**

First I have used AFTER INSERT trigger to create new tables (APPROVED, CANCELLED, and PENDING) for each event category based on the "inserted" table (which contains the newly inserted rows in the "Event\_details" table)

create trigger triginsert on [dbo].[Event\_details]

after insert

as

begin

drop table APPROVED

drop table CANCELLED

drop table PENDING

SELECT event\_id,event\_name

INTO APPROVED

FROM event\_details

where flag\_id=1

SELECT event\_id,event\_name

INTO CANCELLED

FROM event\_details

where flag\_ID=2

SELECT event\_id,event\_name

INTO PENDING

FROM event\_details

where flag\_ID=3

end

go

The second trigger is to be designed to handle events and their available seats based on the student\_event table and the event\_details table. It updates the seats\_left in the event\_details table when a student registers for an event and the registration become successful, and it updates the status\_id to cancelled in the student\_event table when there are no available seats.

create trigger trigafterinsert

on student\_event

after insert

as

begin

declare @event\_id int;

select @event\_id=i.event\_id from inserted i;

declare @status\_id int;

select @status\_id=i.status\_id from inserted i;

declare @seats int;

select @seats=seats\_left from event\_details where event\_id=@event\_id;

if(@status\_id=1 and @seats>0)

update event\_details set seats\_left=(seats\_left-1) where event\_id=@event\_id

else

update student\_event set status\_id=2

end

**5. Creating Views**

In this view I have used data from three tables (Coordinator,Event\_Details,Venue) so that it becomes easy for anyone to check what event is going on at what place and by whom.

create view venue\_event

as

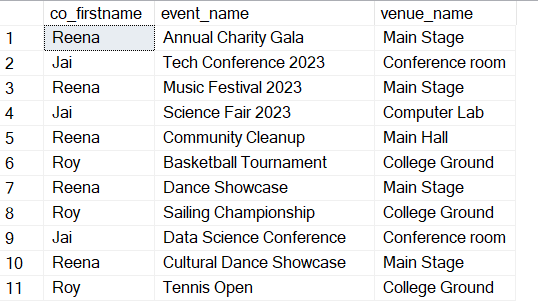
select co\_firstname,event\_name,venue\_name

from Coordinator inner join

Event\_details on Event\_details.co\_id=Coordinator.co\_id

inner join venue on venue.venue\_id=Event\_Details.venue\_id;

select \* from venue\_event



**6. Reports using queries**

1. **Report on the events along with their coordinators and status of approval.**

select event\_id,event\_name,e\_coor\_name,flag\_name

from flag

inner join event\_details

on event\_details.flag\_id=flag.flag\_id

inner join event\_coordinator

on event\_details.E\_CO\_ID=Event\_Coordinator.E\_CO\_ID

1. **Report on retrievel of students in each event who got registration successful**

SELECT e.event\_name, concat(s.first\_name,' ',s.last\_name) as Enrolled\_Students

FROM event\_details e

LEFT JOIN student\_event se ON e.event\_id = se.event\_id

inner JOIN student s ON se.s\_id = s.s\_id AND status\_id=1

1. **Report on number of students in each department according their registration status**

select dept\_name,[Registered],[Cancelled]

from (select dept\_name,status\_name,student.s\_id

from department inner join student

on student.dept\_code=department.dept\_code

inner join student\_event on student.s\_id=student\_event.s\_id

inner join status on status.status\_id=student\_event.status\_id) as Tab1

pivot (count(s\_id) for status\_name in ([Registered],[Cancelled])) as Tab2

1. **Report on getting the details of all the events also the student enrolled for each event with their status of registration**

SELECT E.Event\_ID, E.Event\_Name, E.event\_Start\_Time, E.event\_End\_Time, SE.S\_ID, P.status\_name

FROM Event\_Details E

FULL JOIN Student\_Event SE

ON E.Event\_ID = SE.Event\_ID

full join status P on P.status\_id=SE.status\_id

1. **Report on count of students in each event irrespective of their registration status**

select event\_name,count(s\_id) As Number\_of\_students

from student\_event right join

event\_details on event\_details.event\_id=student\_event.s\_id

group by event\_name

1. **Report on the number of events scheduled in each month**

SELECT datename(month,event\_start\_date) AS Event\_date, COUNT( event\_id)

AS No\_of\_events

FROM event\_details

group by datename(month,event\_start\_date),month(Event\_start\_date)

1. **Report on registered students with their respective events and venue place**

SELECT student.first\_name,student.last\_name, event\_details.event\_name,venue.venue\_name

FROM student

INNER JOIN student\_event ON student\_event.s\_id = student.s\_id

INNER JOIN event\_details ON event\_details.Event\_id = student\_event.event\_id

inner join venue on event\_details.venue\_id=venue.venue\_id

WHERE event\_details.event\_id IN (SELECT event\_id FROM event\_details WHERE flag\_id = 3)

and student\_event.status\_id=1;

1. **Assign username and password to registered students to access important information**

SELECT DISTINCT(CONCAT(student.first\_name, ' ',student.middle\_name, ' ', student.last\_name)) AS full\_name,

CONCAT(LOWER(LEFT(student.first\_name, 1)), '.', lower(student.last\_name), '@gmail.com') AS Username, student.contact AS Password

FROM student

left join student\_event on student.S\_id=student\_event.s\_id

inner join status

on status.status\_id=student\_event.status\_id

where status.status\_name like 'R%';

1. **Make a report on the events datewise that has been approved**

SELECT distinct(event\_details.event\_name), event\_coordinator.e\_coor\_name, event\_details.event\_start\_date

FROM event\_details

INNER JOIN event\_coordinator ON event\_details.E\_CO\_ID = event\_coordinator.E\_CO\_ID

INNER JOIN flag on flag.flag\_id=1

ORDER BY event\_details.event\_start\_date;

1. **Make a report on duration needed for each event**

SELECT distinct(event\_details.event\_name), DATEDIFF(DAY, event\_details.event\_start\_date, event\_details.event\_end\_date) AS duration

FROM event\_details

order BY event\_details.event\_name;

1. **Report on number of events occurring at each venue**

select VENUE\_name,[Cultural],[Sports],[Technical]

from (select venue\_name,event\_type,event\_id

from event\_details inner join venue

on Event\_Details.venue\_id=venue.venue\_id

inner join coordinator on coordinator.co\_id=event\_details.co\_id) as Tab1

pivot (count(event\_id) for event\_type in ([Cultural],[Sports],[Technical])) as Tab2

1. **Report on all events that are fully booked**

SELECT event\_name FROM event\_details e

WHERE NOT EXISTS (

SELECT 1 FROM event\_details

WHERE event\_id = e.event\_id AND seats\_left > 0)