-- ================================

-- Optimized SQL Schema for College\_Mantra

-- ================================

-- 0. Drop and Create Database

DROP DATABASE IF EXISTS College\_Mantra;

CREATE DATABASE College\_Mantra;

USE College\_Mantra;

-- 1. Candidate Table

CREATE TABLE Candidate (

username VARCHAR(150) PRIMARY KEY,

Roll\_No INT,

Candidate\_Name VARCHAR(50),

Gender VARCHAR(10),

DOB DATE,

Candidate\_Rank INT,

XII\_Percentage INT,

Category VARCHAR(20),

Nationality VARCHAR(30),

Address VARCHAR(200),

Email VARCHAR(100),

Phone VARCHAR(15)

);

-- 2. College Table

CREATE TABLE College (

College\_ID INT PRIMARY KEY,

College\_Name VARCHAR(100),

College\_Type VARCHAR(20),

Location VARCHAR(50),

Contact\_No VARCHAR(15),

Website VARCHAR(100),

Email VARCHAR(100)

);

-- 3. Course Table

DROP TABLE IF EXISTS Course;

CREATE TABLE Course (

Course\_ID INT PRIMARY KEY,

Branch\_Name VARCHAR(50),

Program\_Name VARCHAR(50)

);

-- 4. College\_Course Mapping

DROP TABLE IF EXISTS College\_Course;

CREATE TABLE College\_Course (

College\_ID INT,

Course\_ID INT,

PRIMARY KEY (College\_ID, Course\_ID),

FOREIGN KEY (College\_ID) REFERENCES College(College\_ID) ON DELETE CASCADE,

FOREIGN KEY (Course\_ID) REFERENCES Course(Course\_ID) ON DELETE CASCADE

);

-- 5. Seat Matrix with Trigger

DROP TABLE IF EXISTS Seat\_Matrix;

CREATE TABLE Seat\_Matrix (

College\_ID INT,

Course\_ID INT,

General INT, General\_PwD INT,

OBC\_NCL INT, OBC\_NCL\_PwD INT,

SC INT, SC\_PwD INT,

ST INT, ST\_PwD INT,

Total\_Seats INT,

Allocated\_Seats INT DEFAULT 0,

PRIMARY KEY (College\_ID, Course\_ID),

FOREIGN KEY (College\_ID, Course\_ID)

REFERENCES College\_Course(College\_ID, Course\_ID) ON DELETE CASCADE

);

-- Trigger to auto-calculate Total\_Seats on INSERT and UPDATE

DELIMITER $$

CREATE TRIGGER trg\_seat\_total BEFORE INSERT ON Seat\_Matrix

FOR EACH ROW

SET NEW.Total\_Seats = NEW.General + NEW.General\_PwD + NEW.OBC\_NCL + NEW.OBC\_NCL\_PwD +

NEW.SC + NEW.SC\_PwD + NEW.ST + NEW.ST\_PwD;

$$

CREATE TRIGGER trg\_seat\_total\_upd BEFORE UPDATE ON Seat\_Matrix

FOR EACH ROW

SET NEW.Total\_Seats = NEW.General + NEW.General\_PwD + NEW.OBC\_NCL + NEW.OBC\_NCL\_PwD +

NEW.SC + NEW.SC\_PwD + NEW.ST + NEW.ST\_PwD;

$$

DELIMITER ;

-- 6. Preferences & Candidate Mapping

DROP TABLE IF EXISTS Preference;

CREATE TABLE Preference (

Choice\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Choice\_No INT,

College\_ID INT,

Course\_ID INT,

FOREIGN KEY (College\_ID, Course\_ID)

REFERENCES College\_Course(College\_ID, Course\_ID) ON DELETE CASCADE

);

DROP TABLE IF EXISTS can\_pref;

CREATE TABLE can\_pref (

username VARCHAR(150),

Choice\_ID INT,

PRIMARY KEY (username, Choice\_ID),

FOREIGN KEY (username) REFERENCES Candidate(username) ON DELETE CASCADE,

FOREIGN KEY (Choice\_ID) REFERENCES Preference(Choice\_ID) ON DELETE CASCADE

);

-- 7. Allocation Tables

DROP TABLE IF EXISTS Allocation;

CREATE TABLE Allocation (

Allocation\_ID INT AUTO\_INCREMENT PRIMARY KEY,

Choice\_ID INT,

Payment\_Status TINYINT DEFAULT 0,

FOREIGN KEY (Choice\_ID) REFERENCES Preference(Choice\_ID) ON DELETE CASCADE

);

DROP TABLE IF EXISTS can\_alloc;

CREATE TABLE can\_alloc (

username VARCHAR(150),

Allocation\_ID INT,

PRIMARY KEY (username, Allocation\_ID),

FOREIGN KEY (username) REFERENCES Candidate(username) ON DELETE CASCADE,

FOREIGN KEY (Allocation\_ID) REFERENCES Allocation(Allocation\_ID) ON DELETE CASCADE

);

DROP TABLE IF EXISTS col\_allo;

CREATE TABLE col\_allo (

College\_ID INT,

Course\_ID INT,

Allocation\_ID INT,

PRIMARY KEY (College\_ID, Course\_ID, Allocation\_ID),

FOREIGN KEY (College\_ID, Course\_ID)

REFERENCES College\_Course(College\_ID, Course\_ID) ON DELETE CASCADE,

FOREIGN KEY (Allocation\_ID) REFERENCES Allocation(Allocation\_ID) ON DELETE CASCADE

);

DROP TABLE IF EXISTS determines;

CREATE TABLE determines (

Choice\_ID INT,

Allocation\_ID INT,

PRIMARY KEY (Choice\_ID, Allocation\_ID),

FOREIGN KEY (Choice\_ID) REFERENCES Preference(Choice\_ID) ON DELETE CASCADE,

FOREIGN KEY (Allocation\_ID) REFERENCES Allocation(Allocation\_ID) ON DELETE CASCADE

);

-- 8. Payments

DROP TABLE IF EXISTS Payment;

CREATE TABLE Payment (

Transaction\_ID INT,

payment\_no INT PRIMARY KEY,

pay\_date DATE

);

DROP TABLE IF EXISTS candidate\_payment;

CREATE TABLE candidate\_payment (

username VARCHAR(150),

payment\_no INT,

PRIMARY KEY (username, payment\_no),

FOREIGN KEY (username) REFERENCES Candidate(username) ON DELETE CASCADE,

FOREIGN KEY (payment\_no) REFERENCES Payment(payment\_no) ON DELETE CASCADE

);

DROP TABLE IF EXISTS confirms;

CREATE TABLE confirms (

payment\_no INT,

Allocation\_ID INT,

PRIMARY KEY (payment\_no, Allocation\_ID),

FOREIGN KEY (payment\_no) REFERENCES Payment(payment\_no) ON DELETE CASCADE,

FOREIGN KEY (Allocation\_ID) REFERENCES Allocation(Allocation\_ID) ON DELETE CASCADE

);

-- 9. AllocateSeats Procedure

DROP PROCEDURE IF EXISTS AllocateSeats;

DELIMITER $$

CREATE PROCEDURE AllocateSeats()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE done\_choice INT DEFAULT 0;

DECLARE can\_id VARCHAR(150);

DECLARE candidate\_rank INT;

DECLARE can\_category VARCHAR(20);

DECLARE ch\_id INT;

DECLARE col\_id INT;

DECLARE cou\_id INT;

DECLARE ch\_no INT;

DECLARE alloc\_id INT;

DECLARE is\_allocated INT DEFAULT 0;

DECLARE seat\_available INT;

-- Candidate cursor ordered by category priority and rank

DECLARE candidate\_cursor CURSOR FOR

SELECT c.username, c.Candidate\_Rank, c.Category

FROM Candidate c

ORDER BY

CASE c.Category

WHEN 'General' THEN 1

WHEN 'General\_PwD' THEN 2

WHEN 'OBC\_NCL' THEN 3

WHEN 'OBC\_NCL\_PwD' THEN 4

WHEN 'SC' THEN 5

WHEN 'SC\_PwD' THEN 6

WHEN 'ST' THEN 7

WHEN 'ST\_PwD' THEN 8

ELSE 9

END,

c.Candidate\_Rank;

-- Choice cursor

DECLARE choice\_cursor CURSOR FOR

SELECT Choice\_ID, College\_ID, Course\_ID, Choice\_No FROM temp\_choices ORDER BY Choice\_No;

-- Handler

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

START TRANSACTION;

OPEN candidate\_cursor;

candidate\_loop: LOOP

FETCH candidate\_cursor INTO can\_id, candidate\_rank, can\_category;

IF done THEN

LEAVE candidate\_loop;

END IF;

SET done = 0;

SET done\_choice = 0;

SET is\_allocated = 0;

-- Prepare temp\_choices table

DROP TEMPORARY TABLE IF EXISTS temp\_choices;

CREATE TEMPORARY TABLE temp\_choices (

Choice\_ID INT,

College\_ID INT,

Course\_ID INT,

Choice\_No INT

);

INSERT INTO temp\_choices

SELECT p.Choice\_ID, p.College\_ID, p.Course\_ID, p.Choice\_No

FROM can\_pref cp

JOIN Preference p ON cp.Choice\_ID = p.Choice\_ID

WHERE cp.username = can\_id

ORDER BY p.Choice\_No;

OPEN choice\_cursor;

choice\_loop: LOOP

FETCH choice\_cursor INTO ch\_id, col\_id, cou\_id, ch\_no;

IF done THEN

SET done\_choice = 1;

LEAVE choice\_loop;

END IF;

-- Check seat availability for candidate's category

IF can\_category = 'General' THEN

SELECT General > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'General\_PwD' THEN

SELECT General\_PwD > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'OBC\_NCL' THEN

SELECT OBC\_NCL > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'OBC\_NCL\_PwD' THEN

SELECT OBC\_NCL\_PwD > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'SC' THEN

SELECT SC > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'SC\_PwD' THEN

SELECT SC\_PwD > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'ST' THEN

SELECT ST > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'ST\_PwD' THEN

SELECT ST\_PwD > 0 INTO seat\_available

FROM Seat\_Matrix WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSE

SET seat\_available = 0;

END IF;

IF seat\_available > 0 THEN

-- Insert allocation

INSERT INTO Allocation (Choice\_ID, Payment\_Status) VALUES (ch\_id, 0);

SET alloc\_id = LAST\_INSERT\_ID();

INSERT INTO can\_alloc (username, Allocation\_ID) VALUES (can\_id, alloc\_id);

-- Decrease seat count for category

IF can\_category = 'General' THEN

UPDATE Seat\_Matrix SET General = General - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'General\_PwD' THEN

UPDATE Seat\_Matrix SET General\_PwD = General\_PwD - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'OBC\_NCL' THEN

UPDATE Seat\_Matrix SET OBC\_NCL = OBC\_NCL - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'OBC\_NCL\_PwD' THEN

UPDATE Seat\_Matrix SET OBC\_NCL\_PwD = OBC\_NCL\_PwD - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'SC' THEN

UPDATE Seat\_Matrix SET SC = SC - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'SC\_PwD' THEN

UPDATE Seat\_Matrix SET SC\_PwD = SC\_PwD - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'ST' THEN

UPDATE Seat\_Matrix SET ST = ST - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

ELSEIF can\_category = 'ST\_PwD' THEN

UPDATE Seat\_Matrix SET ST\_PwD = ST\_PwD - 1, Allocated\_Seats = Allocated\_Seats + 1

WHERE College\_ID = col\_id AND Course\_ID = cou\_id;

END IF;

SET is\_allocated = 1;

LEAVE choice\_loop;

END IF;

END LOOP;

CLOSE choice\_cursor;

SET done = 0;

DROP TEMPORARY TABLE IF EXISTS temp\_choices;

END LOOP;

CLOSE candidate\_cursor;

COMMIT;

SELECT '✅ Seat Allocation Completed by Category + Rank' AS Status;

END$$

DELIMITER ;

-- 10. Sample Data (Optional, but useful for testing)

INSERT INTO Candidate (username, Roll\_No, Candidate\_Name, Gender, DOB, Candidate\_Rank, XII\_Percentage, Category, Nationality, Address, Email, Phone)

VALUES

('user1', 101, 'Ankit Sharma', 'Male', '2005-04-15', 1, 95, 'General', 'Indian', 'Delhi', 'ankit@example.com', '9876543210'),

('user2', 102, 'Priya Verma', 'Female', '2005-06-22', 2, 93, 'General', 'Indian', 'Mumbai', 'priya@example.com', '9876543211');

INSERT INTO College (College\_ID, College\_Name, College\_Type, Contact\_No, Location, Email, Website)

VALUES

(10001, 'IIT Delhi', 'Government', '01126591234', 'New Delhi', 'contact@iitd.ac.in', 'www.iitd.ac.in'),

(10002, 'NIT Trichy', 'Government', '04312501234', 'Tiruchirappalli', 'info@nitt.edu', 'www.nitt.edu');

INSERT INTO Course (Course\_ID, Branch\_Name, Program\_Name)

VALUES

(201, 'Computer Science', 'B.Tech'),

(202, 'Electrical Engineering', 'B.Tech');

INSERT INTO College\_Course (College\_ID, Course\_ID)

VALUES (10001, 201), (10001, 202), (10002, 201);

INSERT INTO Seat\_Matrix (College\_ID, Course\_ID, General, General\_PwD, OBC\_NCL, OBC\_NCL\_PwD, SC, SC\_PwD, ST, ST\_PwD)

VALUES

(10001, 201, 1,1,1,0,1,0,1,0),

(10001, 202, 1,0,1,0,1,0,1,0),

(10002, 201, 1,1,1,1,1,1,1,1);

SELECT

cp.username,

p.Choice\_No,

p.College\_ID,

p.Course\_ID

FROM

can\_pref cp

JOIN

Preference p ON cp.Choice\_ID = p.Choice\_ID

ORDER BY

cp.username, p.Choice\_No;

-- Preferences

-- user1

-- Choice 1

INSERT INTO Preference (Choice\_No, College\_ID, Course\_ID)

VALUES (1, 10001, 202);

SET @ch1 = LAST\_INSERT\_ID();

INSERT INTO can\_pref (username, Choice\_ID) VALUES ('240001003', @ch1);

-- Choice 2

INSERT INTO Preference (Choice\_No, College\_ID, Course\_ID)

VALUES (2, 10001, 201);

SET @ch2 = LAST\_INSERT\_ID();

INSERT INTO can\_pref (username, Choice\_ID) VALUES ('240001003', @ch2);

-- Choice 3

INSERT INTO Preference (Choice\_No, College\_ID, Course\_ID)

VALUES (3, 10002, 201);

SET @ch3 = LAST\_INSERT\_ID();

INSERT INTO can\_pref (username, Choice\_ID) VALUES ('240001003', @ch3);

-- user2

-- Choice 1

INSERT INTO Preference (Choice\_No, College\_ID, Course\_ID)

VALUES (1, 10001, 202);

SET @ch1 = LAST\_INSERT\_ID();

INSERT INTO can\_pref (username, Choice\_ID) VALUES ('240001004', @ch1);

-- Choice 2

INSERT INTO Preference (Choice\_No, College\_ID, Course\_ID)

VALUES (2, 10001, 201);

SET @ch2 = LAST\_INSERT\_ID();

INSERT INTO can\_pref (username, Choice\_ID) VALUES ('240001004', @ch2);

-- Choice 3

INSERT INTO Preference (Choice\_No, College\_ID, Course\_ID)

VALUES (3, 10002, 201);

SET @ch3 = LAST\_INSERT\_ID();

INSERT INTO can\_pref (username, Choice\_ID) VALUES ('240001004', @ch3);

-- Execute Seat Allocation

CALL AllocateSeats();

SELECT ca.username, a.Allocation\_ID, cl.College\_Name, co.Branch\_Name, p.Choice\_No

FROM can\_alloc ca

JOIN Allocation a ON ca.Allocation\_ID = a.Allocation\_ID

JOIN Preference p ON a.Choice\_ID = p.Choice\_ID

JOIN College cl ON p.College\_ID = cl.College\_ID

JOIN Course co ON p.Course\_ID = co.Course\_ID;