Nathan Gopee

Professor Suchy

CPS 393

25 September 2024

Using the Soccer Database from class

1. Get a list of players names and shirt numbers that play on the team with id = 1

```
SELECT firstName, lastName, shirtNumber
FROM PLAYER
WHERE teamId = 1;
```

2. Get a list of players names and shirt numbers that play on the team named “Arsenal"

```
SELECT PLAYER.firstName, PLAYER.lastName, PLAYER.shirtNumber
FROM PLAYER
JOIN TEAM ON PLAYER.teamId = TEAM.id
WHERE TEAM.name = 'Arsenal';
```

3. Find the team name with the most players

```
SELECT TEAM.name, COUNT(PLAYER.id) AS player_count
FROM PLAYER
JOIN TEAM ON PLAYER.teamId = TEAM.id
GROUP BY TEAM.name
ORDER BY player_count DESC
LIMIT 1;
```

4. Order the matches that have occurred (happened before today date) in chronological order:

```
SELECT *
FROM MATCH
WHERE matchDate < CURDATE()
ORDER BY matchDate ASC;
```

5. Get last and first names of the main referee and the match date of each match

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.matchDate
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id;
```

6. Repeat 5 but for matches that have not happened

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.matchDate
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate >= CURDATE();
```

7. Repeat 5 but for matches that have happened

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.matchDate
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate < CURDATE();
```

8. Get last, first name of the main ref, home score , away score from games that happened

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.homeScore,
MATCH.awayScore
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate < CURDATE();
```

9. Get last, first name of the main ref, home score , away score from games that happened and home team won

```
SELECT REFEREE.lastName, REFEREE.firstName, MATCH.homeScore,
MATCH.awayScore
FROM MATCH
JOIN REFEREE ON MATCH.mainRefereeId = REFEREE.id
WHERE MATCH.matchDate < CURDATE() AND MATCH.homeScore >
MATCH.awayScore;
```

10. Get the team name, home score , away score from games that happened

```
SELECT TEAM.name, MATCH.homeScore, MATCH.awayScore
FROM MATCH
JOIN TEAM ON MATCH.homeTeamId = TEAM.id
WHERE MATCH.matchDate < CURDATE();
```

11. Get team name,home score, away score from games that happened and home team won

```
SELECT TEAM.name, MATCH.homeScore, MATCH.awayScore
FROM MATCH
JOIN TEAM ON MATCH.homeTeamId = TEAM.id
WHERE MATCH.matchDate < CURDATE() AND MATCH.homeScore >
MATCH.awayScore;
```

12. Get team names for teams that competed in match id = 5

```
SELECT homeTeam.name AS HomeTeam, awayTeam.name AS AwayTeam
FROM MATCH
JOIN TEAM AS homeTeam ON MATCH.homeTeamId = homeTeam.id
JOIN TEAM AS awayTeam ON MATCH.awayTeamId = awayTeam.id
WHERE MATCH.id = 5;
```

Using the world database

1. Get the country name with the most cities.

```
SELECT Country.Name, COUNT(City.ID) AS city_count
FROM City
JOIN Country ON City.CountryCode = Country.Code
GROUP BY Country.Name
ORDER BY city_count DESC
LIMIT 1;
```

2. Get the city with the highest population.

```
SELECT Name, Population
FROM City
ORDER BY Population DESC
```

```
LIMIT 1;
```

3. Get the language spoken in the country of your choosing.

```
SELECT Language
FROM CountryLanguage
WHERE CountryCode = 'USA';
```

Using either database

1. Come up with 3 questions of your own
    a. Which players have a shirt number higher than 10 on the team named "Manchester United"?
    b. Which countries have more than 3 languages?
    c. Which referees have more than 20 years of experience?

2. Come up with queries to get the answer

    a.

```
SELECT PLAYER.firstName, PLAYER.lastName, PLAYER.shirtNumber
FROM PLAYER
JOIN TEAM ON PLAYER.teamId = TEAM.id
WHERE TEAM.name = 'Manchester United' AND PLAYER.shirtNumber
> 10;
```

    b.

```
SELECT Country.Name, COUNT(CountryLanguage.Language) AS
language_count
FROM CountryLanguage
JOIN Country ON Country.Code = CountryLanguage.CountryCode
WHERE CountryLanguage.IsOfficial = 'T'
GROUP BY Country.Name
HAVING language_count > 3;
```

    c.

```
SELECT firstName, lastName, yearsOfExperiance
FROM REFEREE
WHERE yearsOfExperiance > 20;
```

1. What users/roles do we have?

    Students

    Professors

2. What tables should we have?

    Students

    Professors

    Departments

    Courses

    Lectures

    Lecture_Offerings

    Student_Registrations

3. What columns in each table?

    - Students
        - student_id (Primary Key)
        - name
        - registration_number (Unique)
        - course_id (Foreign Key from Courses)
    - Professors
        - professor_id (Primary Key)
        - name
        - department_id (Foreign Key from Departments)
    - Departments
        - department_id (Primary Key)
        - department_name
    - Courses
        - course_id (Primary Key)
        - course_name
        - department_id (Foreign Key from Departments)
    - Lectures
        - lecture_id (Primary Key)
        - title

        - credits
        - course_id (Foreign Key from Courses)
        - prerequisite_lecture_id (Foreign Key from Lectures, Nullable)
    - Lecture_Offerings
        - offering_id (Primary Key)
        - lecture_id (Foreign Key from Lectures)
        - professor_id (Foreign Key from Professors)
        - day_of_week
        - time
        - room_number
        - year
        - semester
    - Student_Registrations
        - registration_id (Primary Key)
        - student_id (Foreign Key from Students)
        - offering_id (Foreign Key from Lecture_Offerings)

4. Make sure you identify your primary and foreign keys

5. Create a database.

6. Create your tables in MySql.

```
CREATE DATABASE university;
USE university;
CREATE TABLE Students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    registration_number VARCHAR(50) UNIQUE,
    course_id INT,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
CREATE TABLE Professors (
    professor_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    department_id INT,
```

```
    FOREIGN KEY (department_id) REFERENCES
Departments(department_id)
);
CREATE TABLE Departments (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    department_name VARCHAR(100)
);
CREATE TABLE Courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(100),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES
Departments(department_id)
);
CREATE TABLE Lectures (
    lecture_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100),
    credits INT,
    course_id INT,
    prerequisite_lecture_id INT NULL,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (prerequisite_lecture_id) REFERENCES
Lectures(lecture_id)
);
CREATE TABLE Lecture_Offerings (
    offering_id INT AUTO_INCREMENT PRIMARY KEY,
    lecture_id INT,
    professor_id INT,
    day_of_week VARCHAR(10),
    time TIME,
```

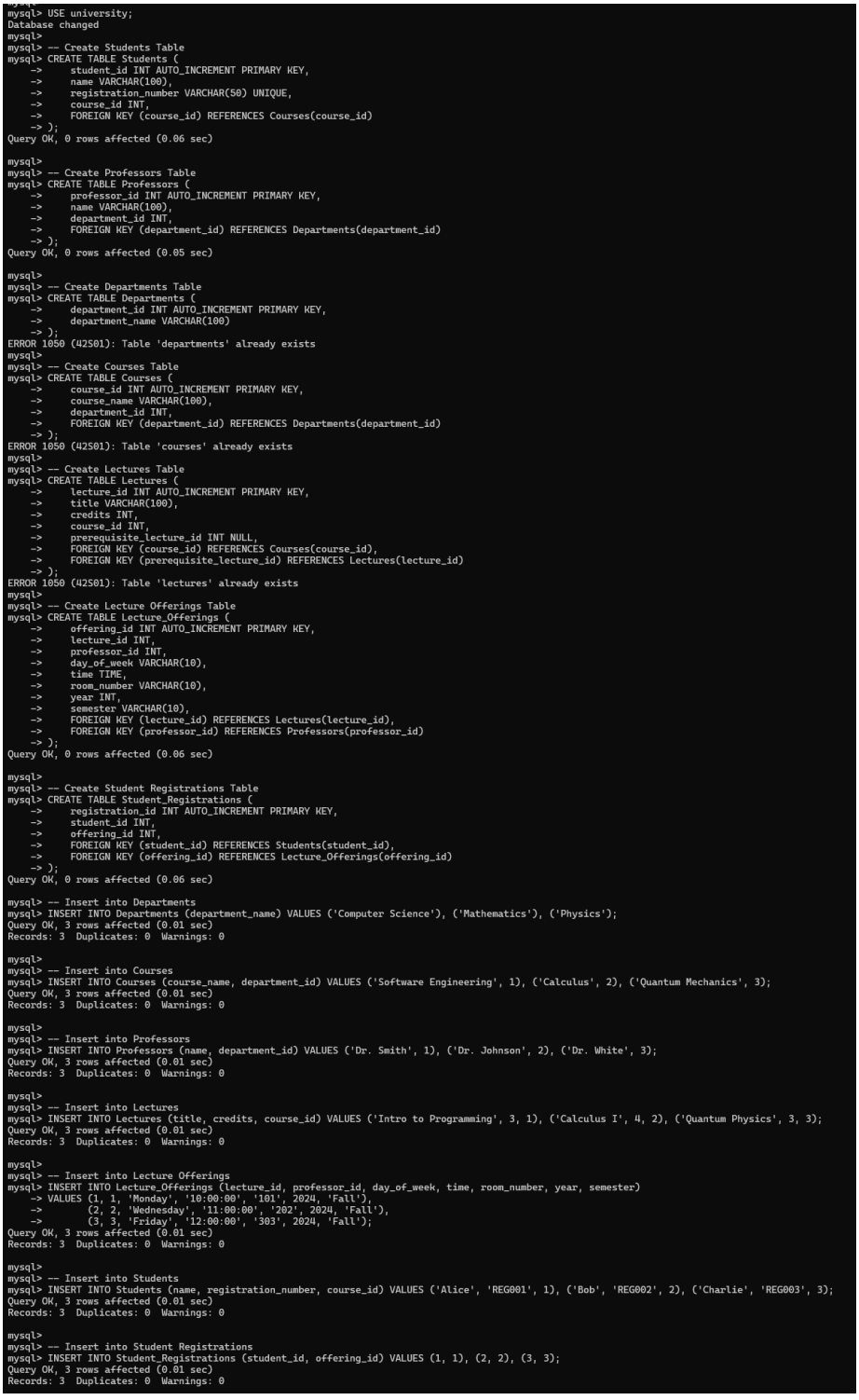
```
    room_number VARCHAR(10),
    year INT,
    semester VARCHAR(10),
    FOREIGN KEY (lecture_id) REFERENCES
Lectures(lecture_id),
    FOREIGN KEY (professor_id) REFERENCES
Professors(professor_id)
);
CREATE TABLE Student_Registrations (
    registration_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    offering_id INT,
    FOREIGN KEY (student_id) REFERENCES
Students(student_id),
    FOREIGN KEY (offering_id) REFERENCES
Lecture_Offerings(offering_id)
);
```

7. Populate each table with 3-5 rows of data.

```
INSERT INTO Departments (department_name) VALUES ('Computer
Science'), ('Mathematics'), ('Physics');
INSERT INTO Courses (course_name, department_id) VALUES
('Software Engineering', 1), ('Calculus', 2), ('Quantum
Mechanics', 3);
INSERT INTO Professors (name, department_id) VALUES ('Dr.
Smith', 1), ('Dr. Johnson', 2), ('Dr. White', 3);
INSERT INTO Lectures (title, credits, course_id) VALUES
('Intro to Programming', 3, 1), ('Calculus I', 4, 2),
('Quantum Physics', 3, 3);
INSERT INTO Lecture_Offerings (lecture_id, professor_id,
day_of_week, time, room_number, year, semester)
VALUES (1, 1, 'Monday', '10:00:00', '101', 2024, 'Fall'),
       (2, 2, 'Wednesday', '11:00:00', '202', 2024, 'Fall'),
       (3, 3, 'Friday', '12:00:00', '303', 2024, 'Fall');
```

```
INSERT INTO Students (name, registration_number, course_id)
VALUES ('Alice', 'REG001', 1), ('Bob', 'REG002', 2),
('Charlie', 'REG003', 3);

INSERT INTO Student_Registrations (student_id, offering_id)
VALUES (1, 1), (2, 2), (3, 3);
```



```
mysql> USE university;
Database changed
mysql>
mysql> -- Create Students Table
mysql> CREATE TABLE Students (
    ->     student_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100),
    ->     registration_number VARCHAR(50) UNIQUE,
    ->     course_id INT,
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> -- Create Professors Table
mysql> CREATE TABLE Professors (
    ->     professor_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100),
    ->     department_id INT,
    ->     FOREIGN KEY (department_id) REFERENCES Departments(department_id)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> -- Create Departments Table
mysql> CREATE TABLE Departments (
    ->     department_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     department_name VARCHAR(100)
    -> );
ERROR 1050 (42S01): Table 'departments' already exists
mysql>
mysql> -- Create Courses Table
mysql> CREATE TABLE Courses (
    ->     course_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     course_name VARCHAR(100),
    ->     department_id INT,
    ->     FOREIGN KEY (department_id) REFERENCES Departments(department_id)
    -> );
ERROR 1050 (42S01): Table 'courses' already exists
mysql>
mysql> -- Create Lectures Table
mysql> CREATE TABLE Lectures (
    ->     lecture_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     title VARCHAR(100),
    ->     credits INT,
    ->     course_id INT,
    ->     prerequisite_lecture_id INT NULL,
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    ->     FOREIGN KEY (prerequisite_lecture_id) REFERENCES Lectures(lecture_id)
    -> );
ERROR 1050 (42S01): Table 'lectures' already exists
mysql>
mysql> -- Create Lecture Offerings Table
mysql> CREATE TABLE Lecture_Offerings (
    ->     offering_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     lecture_id INT,
    ->     professor_id INT,
    ->     day_of_week VARCHAR(10),
    ->     time TIME,
    ->     room_number VARCHAR(10),
    ->     year INT,
    ->     semester VARCHAR(10),
    ->     FOREIGN KEY (lecture_id) REFERENCES Lectures(lecture_id),
    ->     FOREIGN KEY (professor_id) REFERENCES Professors(professor_id)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> -- Create Student Registrations Table
mysql> CREATE TABLE Student_Registrations (
    ->     registration_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     student_id INT,
    ->     offering_id INT,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->     FOREIGN KEY (offering_id) REFERENCES Lecture_Offerings(offering_id)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> -- Insert into Departments
mysql> INSERT INTO Departments (department_name) VALUES ('Computer Science'), ('Mathematics'), ('Physics');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Courses
mysql> INSERT INTO Courses (course_name, department_id) VALUES ('Software Engineering', 1), ('Calculus', 2), ('Quantum Mechanics', 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Professors
mysql> INSERT INTO Professors (name, department_id) VALUES ('Dr. Smith', 1), ('Dr. Johnson', 2), ('Dr. White', 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Lectures
mysql> INSERT INTO Lectures (title, credits, course_id) VALUES ('Intro to Programming', 3, 1), ('Calculus I', 4, 2), ('Quantum Physics', 3, 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Lecture Offerings
mysql> INSERT INTO Lecture_Offerings (lecture_id, professor_id, day_of_week, time, room_number, year, semester)
    -> VALUES (1, 1, 'Monday', '10:00:00', '101', 2024, 'Fall'),
    ->        (2, 2, 'Wednesday', '11:00:00', '202', 2024, 'Fall'),
    ->        (3, 3, 'Friday', '12:00:00', '303', 2024, 'Fall');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Students
mysql> INSERT INTO Students (name, registration_number, course_id) VALUES ('Alice', 'REG001', 1), ('Bob', 'REG002', 2), ('Charlie', 'REG003', 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Insert into Student Registrations
mysql> INSERT INTO Student_Registrations (student_id, offering_id) VALUES (1, 1), (2, 2), (3, 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

8. Upload your work to BitBucket

https://github.com/ndg8743/PPC2024

https://ndg8743.atlassian.net/jira/software/projects/PPC2024/

9. Delete a row of data.

```
DELETE FROM Students WHERE student_id = 3;
```

10. Update a row of data.

```
UPDATE Students SET name = 'Byrd Kat' WHERE student_id = 1;
```

11. Delete a table.

```
DROP TABLE Student_Registrations;
```

12. Delete your whole database

```
DROP DATABASE university;
```

13. Reget your database from BitBucket.

```
git clone
```

14. Practice joining tables and answering questions using your database

```
SELECT Students.name, Lectures.title

FROM Students

JOIN Student_Registrations ON Students.student_id =
Student_Registrations.student_id

JOIN Lecture_Offerings ON Student_Registrations.offering_id
= Lecture_Offerings.offering_id

JOIN Lectures ON Lecture_Offerings.lecture_id =
Lectures.lecture_id;

SELECT Professors.name, Lectures.title

FROM Professors

JOIN Lecture_Offerings ON Professors.professor_id =
Lecture_Offerings.professor_id

JOIN Lectures ON Lecture_Offerings.lecture_id =
Lectures.lecture_id;
```