Задание на проектирование

Курс	Начало занятий	Часы	Преподаватель	Студенты
SQL	01.01.2022	32	Глушенко С.А.	Иванов Петров Сидоров
SQL	01.02.2022	32	Шкодина Т.А.	Степанов
Java	01.02.2022	64	Шкодина Т.А.	Иванов Степанов
Python	01.02.2022	48	Глушенко С.А.	Степанов
Python	01.01.2022	48	Глушенко С.А.	Иванов Петров
Модель ДПО. Курс начинается по мере формирования группы				
			Построить БД, провести нормализацию	
			Запросы:	1) Самый длительный курс
				2) Число курсов по каждому преподавателя
				3) Все студенты
				4) Суммарные часы обучения для каждого студента
				5) Преподаватель, не прочитавший пока ни одного курса
				6) Средний размер группы

Шаги нормализации

1. **Выделим сущности:**
   - Курсы (Courses)
   - Преподаватели (Instructors)
   - Студенты (Students)
   - Участие студентов в курсах (Student_Course)

1. Первая нормальная форма (1NF)

Первая нормальная форма требует, чтобы все значения атрибутов были атомарными, и чтобы у каждой записи была уникальная идентификация (первичный ключ).

Таблица "Courses" (Курсы)

CourseID	CourseName	StartDate	Hours	InstructorID
1	SQL	2022-01-01	32	1
2	SQL	2022-02-01	32	2

3	Java	2022-02-01	64	2
4	Python	2022-02-01	48	1
5	Python	2022-01-01	48	1

Таблица "Instructors" (Преподаватели)

InstructorID	InstructorName
1	Глушенко С.А.
2	Шкодина Т.А.

Таблица "Students" (Студенты)

StudentID	StudentName
1	Иванов
2	Петров
3	Сидоров
4	Степанов

Таблица "Student_Course" (Участие студентов в курсах)

CourseID	StudentID
1	1
1	2
1	3
2	4
3	1
3	4
4	4
5	1
5	2

2. Вторая нормальная форма (2NF)

Вторая нормальная форма требует выполнения требований первой нормальной формы и отсутствия частичных зависимостей между неключевыми атрибутами и частью составного ключа.

Все наши таблицы уже удовлетворяют 2NF, так как в каждой таблице неключевые атрибуты зависят от всего первичного ключа.

3. Третья нормальная форма (3NF)

Третья нормальная форма требует выполнения требований второй нормальной формы и отсутствия транзитивных зависимостей между неключевыми атрибутами.

Все наши таблицы также удовлетворяют 3NF, так как нет транзитивных зависимостей.

SQL-запросы для создания и заполнения таблиц:

```
-- Создание таблицы курсов
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(50),
    StartDate DATE,
    Hours INT,
    InstructorID INT,
    FOREIGN KEY (InstructorID) REFERENCES Instructors(Instr
uctorID)
);

-- Создание таблицы преподавателей
CREATE TABLE Instructors (
    InstructorID INT PRIMARY KEY,
    InstructorName VARCHAR(100)
);

-- Создание таблицы студентов
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    StudentName VARCHAR(50)
);

-- Создание таблицы участия студентов в курсах
CREATE TABLE Student_Course (
    CourseID INT,
```

```
    StudentID INT,
    PRIMARY KEY (CourseID, StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
);

-- Вставка данных в таблицу Instructors
INSERT INTO Instructors (InstructorID, InstructorName) VALU
ES
(1, 'Глушенко С.А.'),
(2, 'Шкодина Т.А.');

-- Вставка данных в таблицу Courses
INSERT INTO Courses (CourseID, CourseName, StartDate, Hour
s, InstructorID) VALUES
(1, 'SQL', '2022-01-01', 32, 1),
(2, 'SQL', '2022-02-01', 32, 2),
(3, 'Java', '2022-02-01', 64, 2),
(4, 'Python', '2022-02-01', 48, 1),
(5, 'Python', '2022-01-01', 48, 1);

-- Вставка данных в таблицу Students
INSERT INTO Students (StudentID, StudentName) VALUES
(1, 'Иванов'),
(2, 'Петров'),
(3, 'Сидоров'),
(4, 'Степанов');

-- Вставка данных в таблицу Student_Course
INSERT INTO Student_Course (CourseID, StudentID) VALUES
(1, 1),
(1, 2),
(1, 3),
(2, 4),
(3, 1),
(3, 4),
(4, 4),
```

```
(5, 1),
(5, 2);
```

Теперь таблица нормализована и готова для выполнения запросов.

Запросы:

1. **Самый длительный курс:**

```
SELECT CourseName, Hours
FROM Courses
ORDER BY Hours DESC
LIMIT 1;
```

2. **Число курсов по каждому преподавателю:**

```
SELECT Instructors.InstructorName, COUNT(Courses.CourseID) AS NumberOfCourses
FROM Courses
JOIN Instructors ON Courses.InstructorID = Instructors.InstructorID
GROUP BY Instructors.InstructorName;
```

3. **Все студенты:**

```
SELECT StudentName
FROM Students;
```

4. **Суммарные часы обучения для каждого студента:**

```
SELECT Students.StudentName, SUM(Courses.Hours) AS TotalHours
FROM Student_Course
JOIN Students ON Student_Course.StudentID = Students.StudentID
JOIN Courses ON Student_Course.CourseID = Courses.CourseID
GROUP BY Students.StudentName;
```

5. **Преподаватель, не прочитавший пока ни одного курса:**

```
SELECT InstructorName
FROM Instructors
WHERE InstructorID NOT IN (SELECT InstructorID FROM Cour
ses);
```

6. **Средний размер группы:**

```
SELECT AVG(GroupSize) AS AverageGroupSize
FROM (
    SELECT CourseID, COUNT(StudentID) AS GroupSize
    FROM Student_Course
    GROUP BY CourseID
) AS GroupSizes;
```

Вот и все! Теперь у тебя есть нормализованная база данных и SQL-запросы для работы с ней.