

Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет телекоммуникаций и
информатики» (СибГУТИ)

Кафедра ПМиК

Расчетно-графическая работа по дисциплине
«Базы данных»
Вариант 15
«Центр изучения иностранных языков»

Выполнил: студент 4 курса
ИВТ, гр. ИП-117
Ротов И.В.

Проверил: ассистент кафедры ПМиК
Чупрыно Лев Александрович

Новосибирск 2024

Задание	2
Описание базы данных	5
1.Описание базы данных	5
1.1.Сущности и связи	5
1.2.Данные БД(содержание таблиц)	7
2.Описание веб-интерфейса	8
3.Запросы	11
Запрос 1.1.	11
Запрос 1.2.	12
Запрос 2.	12
Запрос 3.	13
4.Листинг	13
4.1.SQL запросы для создания таблиц с учетом всех ключей и ограничений, вставки данных.	13
4.2.PHP-скрипты	16

Задание

Работа выполняется по вариантам (вариант выбирается по номеру в журнале)

В ходе работы необходимо:

1. Спроектировать произвольную базу данных, соответствующую предметной области (согласно варианту) и позволяющую корректно хранить информацию об объектах этой предметной области.
2. Обеспечить пользовательский веб-интерфейс для добавления записей и просмотра содержимого БД.
3. Сформулировать и выполнить запросы к БД, имеющие смысл в контексте предметной области (согласно варианту). Обеспечить веб-интерфейс для запуска этих запросов.
4. Составить отчет, содержащий описание выполненной работы, в том числе некоторые этапы проектирования БД (см. ниже).

Требования:

Часть 1 БАЗА ДАННЫХ

- База данных должна содержать минимум 3-4 таблицы, представляющие сущности предметной области (по варианту) или какие-либо отношения между этими сущностями.
- Каждая таблица должна состоять как минимум из трех полей, представляющих реально существующие и разумные свойства сущностей или связей.
- Все таблицы в БД должны находиться в третьей нормальной форме, то есть содержать первичные ключи, внешние ключи для связи таблиц, а также обеспечивать необходимые ограничения на поля (например, непустое поле, уникальные значения и т.д.)

Часть 2 ВЕБ-ИНТЕРФЕЙС

- Веб-интерфейс должен быть написан на языке PHP и обеспечивать для пользователя возможность выбора таблицы из БД, а затем действий с выбранной таблицей: вставки новых записей и просмотра ее содержимого.
- При вставке данных в таблицу необходимо предоставить пользователю поля для ввода всех полей новой записи. При этом необходимо проверять допустимые диапазоны вводимых значений, непустые поля и другие ограничения на данные. В случае ввода некорректных данных пользователь должен получать сообщение об ошибке (user-friendly interface).
- Для вставки данных в различные таблицы допустимо использовать отдельные скрипты (веб-страницы).

Часть 3 ЗАПРОСЫ К БД

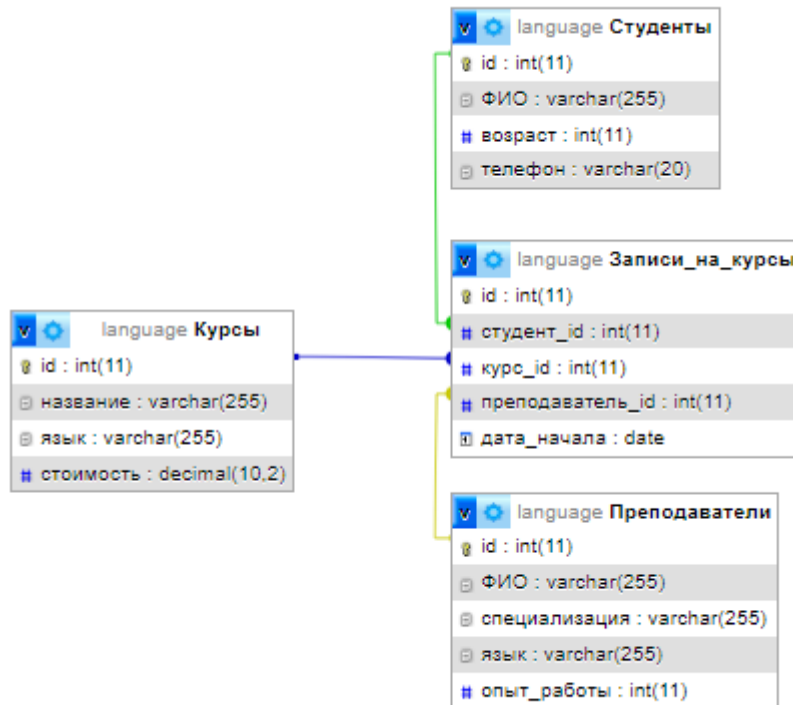
- Сформулировать и написать 2 запроса к БД, осуществляющие выборку минимум из двух таблиц каждый. Запросы должны иметь смысл для конкретной предметной области и конкретную формулировку на русском

языке: например, «вывести имена всех покупателей, совершавших заказы только в августе», «вывести имена всех продавцов, у которых имеется более 3 заказов» и т.д.

- Сформулировать и написать еще 1 запрос к БД, осуществляющих выборку с использованием пользовательских данных. Данные принимаются с веб-страницы. Запрос может использовать одну или более таблиц. Например, «вывести всех покупателей, совершивших заказы ... числа» или «вывести все заказы продавцов из города ...».
- Обеспечить возможность запуска всех 3 различных запросов с веб-страницы (например, по отдельным кнопкам).
- Для запроса 3 обеспечить возможность ввода пользовательских данных (например, с помощью обыкновенного текстового поля). В случае некорректно введенных данных или пустого результата запроса пользователь должен получать соответствующее сообщение (user-friendly). Допустимо (необязательно!!) предоставить для выбора выпадающий список, формирующийся из существующих в БД значений.

Описание базы данных

1.Описание базы данных



1.1.Сущности и связи

Сущность 1	Сущность 2	Тип связи	Описание связи
Курсы	Преподаватели	1:M	Один курс может вести несколько преподавателей.
Курсы	Студенты	M:N	Один курс могут посещать несколько студентов.
Студенты	Записи_на_курсы	1:M	Один студент может записаться на несколько курсов.
Преподаватели	Записи_на_курсы	1:M	Один преподаватель может быть назначен на несколько курсов.
Курсы	Записи_на_курсы	1:M	Один курс может иметь множество записей.

Атрибуты сущностей

Сущность, название таблицы в БД	Название атрибута	Название атрибута в БД (имя поля)	Тип данных	Not Null	Primary Key	Foreign key	Дополнительные ограничения целостности
Курсы (course)	Номер курса	id	INT	1	P		AUTO_INCREMENT
	Название курса	title	VARCHAR (255)	1			NOT NULL
	Язык курса	language	VARCHAR (255)	1			NOT NULL
	Стоимость курса	cost	DECIMAL (10, 2)				NULL
Преподаватели (teachers)	Номер преподавателя	id	INT	1	P		AUTO_INCREMENT
	ФИО преподавателя	name	VARCHAR (255)	1			NOT NULL
	Специализация	specialization	VARCHAR (255)	1			NOT NULL
	Язык преподавания	language	VARCHAR (255)	1			NOT NULL
	Опыт работы	experience	INT	1			CHECK (experience >= 0)
Студенты (students)	Номер студента	id	INT	1	P		AUTO_INCREMENT
	ФИО студента	name	VARCHAR (255)	1			NOT NULL
	Возраст	age	INT	1			CHECK (age >= 16 AND age <= 100)
	Телефон	phone	VARCHAR (255)	1			NOT NULL
Записи на курсы (course_records)	Номер записи	id	INT	1	p		AUTO_INCREMENT
	Номер студента	student_id	INT	1		F	REFERENCES students(id)

	Номер курса	course_id	INT	1		F	REFERENCES courses(id)
	Номер преподавателя	teacher_id	INT	1		F	REFERENCES teachers(id)
	Дата начала курса	start_date	DATE	1			NOT NULL

1.2.Данные БД(содержание таблиц)

id	название	язык	стоимость
1	1 Продвинутый французский	Французский	2000
2	2 Немецкий для начинающих	Немецкий	1800
3	3 Итальянский для путешественников	Итальянский	1200
4	4 Испанский для начинающих	Испанский	1600
5	5 Простой английский	Английский	1000
6	6 Усложненный английский	Английский	1000
7	7 Усложненный английский	Английский	999
8	8 Продвинутый анлийский	Английский	1000

1.1 - Таблица course

id	ФИО	специализация	язык	опыт работы
1	1 Иванов Иван Иванович	Преподаватель английского языка	Английский	5
2	2 Петрова Мария Александровна	Преподаватель французского языка	Французский	7
3	3 Смирнов Сергей Валерьевич	Преподаватель немецкого языка	Немецкий	3
4	4 Кузнецова Ольга Дмитриевна	Преподаватель итальянского языка	Итальянский	10
5	5 Лебедев Алексей Юрьевич	Преподаватель испанского языка	Испанский	6
6	6 Петрова Мария Александровна	Преподаватель английского языка	Английский	3
7	7 Иванов Иван Иванович	Преподаватель итальянского языка	Итальянский	1
8	8 Лебедев Алексей Юрьевич	Преподаватель английского языка	Английский	1

1.2 - Таблица teachers

id	ФИО	возраст	телефон
1	1 Алексеева Наталья Ивановна	25	+7 987 654 3210
2	2 Борисов Дмитрий Владимирович	32	+7 916 234 5678
3	3 Григорьева Екатерина Юрьевна	22	+7 903 876 5432
4	4 Дмитриев Павел Константинович	28	+7 905 123 4567
5	5 Егорова Ирина Павловна	27	+7 911 321 7654
6	6 Илья Ротов	21	+7 912 345 6789
7	7 Султанов Амир	52	+7 965 825 6484

1.3 - Таблица students

id	студент id	курс id	преподаватель id	дата начала
1	1	1	2	3 2024-11-25
2	2	2	1	2 2024-11-13
3	3	1	1	2 2024-11-11
4	4	4	3	7 2024-11-11
5	5	5	6	8 2024-11-10

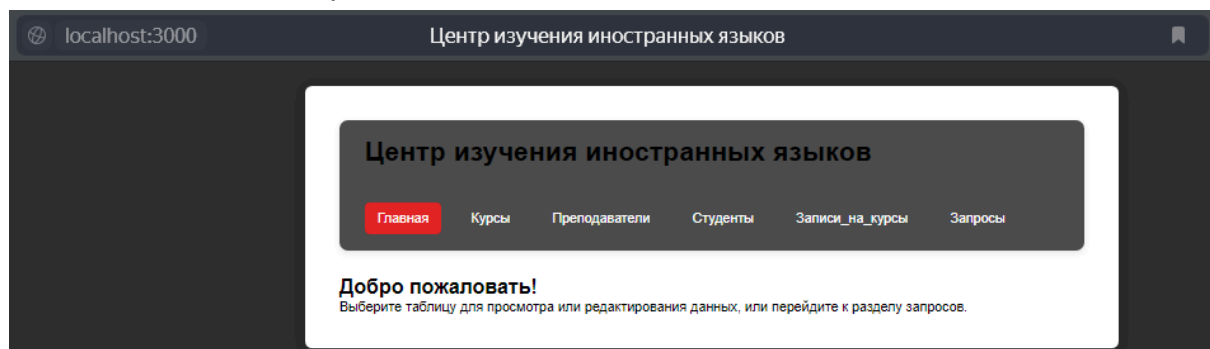
1.4 - Таблица course_records

2.Описание веб-интерфейса

Веб-интерфейс системы управления центром изучения иностранных языков предоставляет пользователям функционал для просмотра, добавления и редактирования данных, разделенных на таблицы: **Курсы**, **Преподаватели**, **Студенты** и **Записи на курсы**.

Главная страница.

На главной странице отображается приветственное сообщение с навигацией, которая позволяет выбрать одну из таблиц для работы или перейти к разделу пользовательских запросов.



2.1 - Главная страница

Таблицы данных.

При выборе таблицы, например **Курсы**, отображается её содержимое в виде таблицы. Пользователь видит данные, такие как название курса, язык и стоимость. Также на этой странице предоставлена форма для добавления новой записи в таблицу, где реализована проверка корректности данных. Например, поле "стоимость" принимает только положительные значения.

[Главная](#) [Курсы](#) [Преподаватели](#) [Студенты](#) [Записи_на_курсы](#) [Запросы](#)

Таблица: Курсы


id	название	язык	стоимость
1	Продвинутый французский	Французский	2000
2	Немецкий для начинающих	Немецкий	1800
3	Итальянский для путешественников	Итальянский	1200
4	Испанский для начинающих	Испанский	1600
5	Простой английский	Английский	1000
6	Усложненный английский	Английский	1000
7	Усложненный английский	Английский	999
8	Продвинутый английский	Английский	1000
9	1	1	1

Добавить новую запись в таблицу Курсы

название:

язык:

стоимость:

 Значение должно быть больше или равно 0.

[Вернуться на главную](#)

2.2 - Таблица данных "Курсы"

Добавление записи.

На странице добавления записей форма адаптирована под структуру выбранной таблицы. Например, для таблицы **Записи на курсы** требуется указать студента, курс, преподавателя и дату начала, где реализованы проверки на совпадение языка курса и специализации преподавателя, а также предотвращение дублирования записей.

ГлавнаяКурсыПреподавателиСтудентыЗаписи_на_курсыЗапросы

Таблица: Записи_на_курсы

id	студент_id	курс_id	преподаватель_id	дата_начала
1	1	2	3	2024-11-25
2	2	1	2	2024-11-13
3	1	1	2	2024-11-11
4	4	3	7	2024-11-11
5	5	6	8	2024-11-10

Добавить новую запись в таблицу Записи_на_курсы

Выберите студента:

Алексеева Наталья Ивановна (ID: 1)

Выберите курс:

Выберите курс

Выберите преподавателя:

Выберите преподавателя

Дата начала курса:

дд . мм . гggg

Добавить запись

[Вернуться на главную](#)

2.3 - Добавление записи в таблице “Записи_на_курсы”

Раздел запросов

В разделе пользовательских запросов отображается функционал выполнения predetermined запросов к базе данных, таких как выборка курсов по определенным критериям или список студентов, записанных на определенные курсы.

Центр изучения иностранных языков

[Главная](#)[Курсы](#)[Преподаватели](#)[Студенты](#)[Записи_на_курсы](#)[Запросы](#)

Запросы

Запрос 1: Студенты и курсы
Вывести имена студентов и названия курсов, на которые они записаны, вместе с датой начала курса.

Выполнить

Запрос 2: Преподаватели и курсы на английском языке
Вывести имена преподавателей, которые ведут курсы на английском языке, и названия этих курсов.

Выполнить

Запрос 3: Студенты по выбранному языку
Выберите язык:

Французский

Выполнить

Запрос 4: Студенты записанные на курс до определенной даты
Выберите дату начала курса:

ДД.ММ.ГГГГ

Выполнить

[Вернуться на главную](#)

2.4 - Запросы

На каждой странице интерфейса отображаются сообщения об ошибках или успехе операций, например, предупреждение о некорректно введенных данных или подтверждение успешного добавления записи.

3.Запросы

Запрос 1.1.

Студенты и курсы.

Вывести имена студентов, названия курсов, на которые они записаны, и даты начала курсов.

SQL:

```
1 SELECT
2     Студенты.ФИО AS Имя_студента,
3     Курсы.название AS Название_курса,
4     Записи_на_курсы.дата_начала AS Дата_начала
5 FROM
6     Студенты
7 JOIN
8     Записи_на_курсы ON Студенты.id = Записи_на_курсы.студент_id
9 JOIN
10    Курсы ON Курсы.id = Записи_на_курсы.курс_id";
```

3.1 - Запрос студентов и курсов

Запрос 1.2.

Преподаватели и курсы на английском языке.

Вывести имена преподавателей, которые ведут курсы на английском языке, и названия этих курсов.

SQL:

```
1 SELECT DISTINCT
2     Преподаватели.ФИО AS Имя_преподавателя,
3     Курсы.название AS Название_курса
4 FROM
5     Преподаватели
6 JOIN
7     Курсы ON Преподаватели.язык = Курсы.язык
8 WHERE
9     Курсы.язык = 'Английский'";
```

3.2 - Запрос преподавателей и курсов на иностранном языке

Запрос 2.

Студенты по выбранному языку.

Вывести имена студентов, названия курсов, которые они изучают, и имена преподавателей, если курс ведется на выбранном языке.

SQL:

```
1 SELECT
2     Студенты.ФИО AS Имя_студента,
3     Курсы.название AS Название_курса,
4     Преподаватели.ФИО AS Имя_преподавателя
5 FROM
6     Студенты
7 JOIN
8     Записи_на_курсы ON Студенты.id = Записи_на_курсы.студент_id
9 JOIN
10    Курсы ON Курсы.id = Записи_на_курсы.курс_id
11 JOIN
12    Преподаватели ON Преподаватели.id = Записи_на_курсы.преподаватель_id
13 WHERE
14    Курсы.язык = :language;
```

3.3 - Запрос студентов по выбранному языку

Запрос 3.

Студенты, записанные на курсы до определенной даты.

Вывести имена студентов, названия курсов, на которые они записаны, и даты начала курсов, если курс начинается до указанной даты.

SQL:

```
1 SELECT
2     Студенты.ФИО AS Имя_студента,
3     Курсы.название AS Название_курса,
4     Записи_на_курсы.дата_начала AS Дата_начала
5 FROM
6     Студенты
7 JOIN
8     Записи_на_курсы ON Студенты.id = Записи_на_курсы.студент_id
9 JOIN
10    Курсы ON Курсы.id = Записи_на_курсы.курс_id
11 WHERE
12    Записи_на_курсы.дата_начала <= :start_date;
```

3.4 - Запрос студентов на выбранном курсе до определенной даты

Результат работы 3 запроса:

Запрос 4: Студенты записанные на курс до определенной даты
Выберите дату начала курса:

29 . 11 . 2024

Выполнить

Имя студента	Название курса	Дата начала
Алексеева Наталья Ивановна	Немецкий для начинающих	2024-11-25
Борисов Дмитрий Владимирович	Продвинутый французский	2024-11-13
Алексеева Наталья Ивановна	Продвинутый французский	2024-11-11
Дмитриев Павел Константинович	Итальянский для путешественников	2024-11-11
Егорова Ирина Павловна	Усложненный английский	2024-11-10

[Вернуться на главную](#)

3.5 - Запрос студентов на записанные курсы

4.Листининг

4.1.SQL запросы для создания таблиц с учетом всех ключей и ограничений, вставки данных.

Создание таблицы Курсы

```
CREATE TABLE IF NOT EXISTS Курсы (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    название TEXT NOT NULL,  
    язык TEXT NOT NULL,  
    стоимость REAL  
)
```

Заполнение таблицы(можно вручную на web-сайте,можно запросами):

```
INSERT INTO Курсы (id, название, язык, стоимость) VALUES  
(1, 'Продвинутый французский', 'Французский', 2000),  
(2, 'Немецкий для начинающих', 'Немецкий', 1800),  
(3, 'Итальянский для путешественников', 'Итальянский', 1200),  
(4, 'Испанский для начинающих', 'Испанский', 1600);
```

Создание таблицы **Преподаватели**

```
CREATE TABLE IF NOT EXISTS Преподаватели (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    ФИО TEXT NOT NULL,  
    специализация TEXT NOT NULL,  
    язык TEXT NOT NULL,  
    опыт_работы INTEGER CHECK(опыт_работы >= 0)  
)
```

Заполнение таблицы(можно вручную на web-сайте,можно запросами):

```
INSERT INTO Преподаватели (id, ФИО, специализация, язык, опыт_работы)  
VALUES  
('Иванов Иван Иванович', 'Преподаватель английского языка', 'Английский', 5),  
('Петрова Мария Александровна', 'Преподаватель французского языка',  
'Французский', 7),  
('Смирнов Сергей Валерьевич', 'Преподаватель немецкого языка', 'Немецкий',  
3),  
('Кузнецова Ольга Дмитриевна', 'Преподаватель итальянского языка',  
'Итальянский', 10),  
('Лебедев Алексей Юрьевич', 'Преподаватель испанского языка', 'Испанский', 6),  
('Петрова Мария Александровна', 'Преподаватель английского языка',  
'Английский', 3),  
('Иванов Иван Иванович', 'Преподаватель итальянского языка', 'Итальянский',  
1),  
('Лебедев Алексей Юрьевич', 'Преподаватель английского языка', 'Английский',  
1);
```

Создание таблицы **Студенты**

```
CREATE TABLE IF NOT EXISTS Студенты (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    ФИО TEXT NOT NULL,  
    возраст INTEGER CHECK(возраст >= 16 AND возраст <= 100),  
    телефон TEXT NOT NULL  
)
```

Заполнение таблицы(можно вручную на web-сайте,можно запросами):

```
INSERT INTO Студенты (id, ФИО, возраст, телефон) VALUES  
('Алексеева Наталья Ивановна', 25, '+7 987 654 3210'),  
('Борисов Дмитрий Владимирович', 30, '+7 916 234 5678'),  
('Григорьева Екатерина Юрьевна', 22, '+7 903 876 5432'),  
('Дмитриев Павел Константинович', 28, '+7 905 123 4567'),  
('Егорова Ирина Павловна', 27, '+7 911 321 7654');
```

Создание таблицы **Записи на курсы**

```
CREATE TABLE IF NOT EXISTS Записи_на_курсы (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    студент_id INTEGER NOT NULL,  
    курс_id INTEGER NOT NULL,  
    преподаватель_id INTEGER NOT NULL,  
    дата_начала TEXT NOT NULL,  
    FOREIGN KEY(студент_id) REFERENCES Студенты(id),  
    FOREIGN KEY(курс_id) REFERENCES Курсы(id),  
    FOREIGN KEY(преподаватель_id) REFERENCES Преподаватели(id)  
)
```

Заполнение таблицы(можно вручную на web-сайте,можно запросами):

```
INSERT INTO Записи_на_курсы (id, студент_id, курс_id, преподаватель_id,  
дата_начала) VALUES  
(1, 2, 3, 3, '2024-11-12'),  
(2, 1, 4, 4, '2024-11-19'),  
(3, 1, 1, 1, '2024-11-15'),  
(4, 1, 2, 2, '2024-11-05'),  
(5, 1, 1, 6, '2024-10-28');
```


4.2.PHP-скрипты

index.php:

```
<?php
require_once 'db.php';
require_once 'functions.php';

$db = require 'db.php';

$action = $_GET['action'] ?? 'home';
$table = $_GET['table'] ?? '';
$validTables = ['Курсы', 'Преподаватели', 'Студенты', 'Записи_на_курсы'];

?>
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Центр изучения иностранных языков</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
<div class="container">
    <header>
        <h1>Центр изучения иностранных языков</h1>
        <nav><ul>
            <li><a href="?action=home" <?php if ($action === 'home') echo
'class="active"'; ?>>Главная</a></li>
            <?php foreach ($validTables as $validTable): ?>
                <li><a href="?action=view&table=<?php echo $validTable; ?>" <?php if
($action === 'view' && $table === $validTable) echo 'class="active"'; ?>><?php echo
$validTable; ?></a></li>
            <?php endforeach; ?>
            <li><a href="?action=queries" <?php if ($action === 'queries') echo
'class="active"'; ?>>Запросы</a></li>
        </ul></nav>
    </header>
```

```

<main>
    <?php
        if ($action === 'view' && in_array($table, $validTables)) {
            $records = $db->query("SELECT * FROM
$table")->fetchAll(PDO::FETCH_ASSOC);
            echo "<h1>Таблица: $table</h1>";
            renderTable($records);
            echo "<h2>Добавить новую запись в таблицу $table</h2>";
            echo "<form method='post' action='?action=add&table=$table'>";

            if ($table === 'Записи_на_курсы') {
                echo "<label>Выберите студента:</label>";
                echo "<select name='студент_id' required>";
                $students = $db->query("SELECT id, ФИО FROM
Студенты")->fetchAll(PDO::FETCH_ASSOC);
                foreach ($students as $student) {
                    echo "<option value='{ $student['id']}'>{ $student['ФИО']} (ID:
{ $student['id']})</option>";
                }
                echo "</select>";

                echo "<label>Выберите курс:</label>";
                echo "<select id='курс_id' name='курс_id' required>";
                echo "<option value=''>Выберите курс</option>";
                $courses = $db->query("SELECT id, название, язык FROM
Курсы")->fetchAll(PDO::FETCH_ASSOC);
                foreach ($courses as $course) {
                    echo "<option value='{ $course['id']}'
data-language='{ $course['язык']}'>{ $course['название']}
({ $course['язык']})</option>";
                }
                echo "</select>";

                echo "<label>Выберите преподавателя:</label>";
                echo "<select id='преподаватель_id' name='преподаватель_id'
required>";
                echo "<option value=''>Выберите преподавателя</option>";
                $teachers = $db->query("SELECT id, ФИО, специализация, язык FROM
Преподаватели")->fetchAll(PDO::FETCH_ASSOC);
                foreach ($teachers as $teacher) {

```

```

        echo "<option value='{${teacher['id']}'
data-language='{${teacher['язык']}'>${teacher['ФИО']} (${teacher['язык']}) (ID:
${teacher['id']})</option>";
    }
    echo "</select>";

```

```

    echo "<label>Дата начала курса:</label>";
    echo "<input type='date' name='дата_начала' required>";
} elseif ($table === 'Курсы') {
    $columns = getColumns($table);
    foreach ($columns as $col) {
        echo "<label>$col:</label>";
        if ($col === 'стоимость') {
            echo "<input type='number' name='$col' min='0' step='any'
required>";
        } else {
            echo "<input type='text' name='$col' required>";
        }
    }
} elseif ($table === 'Студенты') {
    $columns = getColumns($table);
    foreach ($columns as $col) {
        echo "<label>$col:</label>";
        if ($col === 'возраст') {
            echo "<input type='number' name='$col' min='16' max='100'
required>";
        } elseif ($col === 'телефон') {
            echo "<input type='tel' name='$col' pattern='^[0-9\s\-\+\(\)]*$'
placeholder='Пример: +1 (234) 567-8901' required>";
        } else {
            echo "<input type='text' name='$col' required>";
        }
    }
} elseif ($table === 'Преподаватели') {
    $columns = getColumns($table);
    foreach ($columns as $col) {
        echo "<label>$col:</label>";
        if ($col === 'опыт_работы') {
            echo "<input type='number' name='$col' min='0' required>";
        } else {
            echo "<input type='text' name='$col' required>";
        }
    }
}

```

```
}
```

```
echo "<input type='submit' value='Добавить запись'>";  
echo "</form>";
```

```
echo "<a href='?'>Вернуться на главную</a>";
```

```
} elseif ($action === 'add' && in_array($table, $validTables)) {  
    try {  
        $columns = getColumns($table);  
        $values = array_map(fn($col) => $_POST[$col], $columns);  
  
        if ($table === 'Записи_на_курсы') {  
            $студент_id = $_POST['студент_id'];  
            $курс_id = $_POST['курс_id'];  
            $преподаватель_id = $_POST['преподаватель_id'];  
            $дата_начала = $_POST['дата_начала'];  
  
            $checkDuplicate = $db->prepare("SELECT COUNT(*) FROM  
Записи_на_курсы WHERE студент_id = ? AND курс_id = ? AND  
преподаватель_id = ? AND дата_начала = ?");  
            $checkDuplicate->execute([$студент_id, $курс_id, $преподаватель_id,  
$дата_начала]);  
            $exists = $checkDuplicate->fetchColumn();  
  
            if ($exists) {  
                echo "<p class='error'>Ошибка: Эта запись уже существует.</p>";  
                echo "<meta http-equiv='refresh'  
content='3;url=?action=view&table=Записи_на_курсы'>";  
                exit();  
            }  
  
            $courseLanguage = $db->query("SELECT язык FROM Курсы WHERE  
id = $курс_id")->fetchColumn();  
  
            $teacherQuery = $db->prepare("SELECT COUNT(*) FROM  
Преподаватели WHERE id = ? AND язык = ?");  
            $teacherQuery->execute([$преподаватель_id, $courseLanguage]);  
            $languageMatch = $teacherQuery->fetchColumn();
```

```

        if (!$languageMatch) {
            echo "<p class='error'>Ошибка: Преподаватель не владеет языком
выбранного курса или указанный язык не совпадает с его
специализацией.</p>";
            echo "<meta http-equiv='refresh'
content='3;url=?action=view&table=Записи_на_курсы'>";
            exit();
        }
    } else if ($table === 'Преподаватели') {
        $fio = $_POST['ФИО'];
        $language = $_POST['язык'];
        $existingTeacher = checkTeacherExists($fio, $language, $db);
        if ($existingTeacher) {
            echo "<p class='error'>Ошибка: Преподаватель с таким ФИО и
языком уже существует.</p>";
            echo "<meta http-equiv='refresh'
content='3;url=?action=view&table=Преподаватели'>";
            exit();
        }
    } elseif ($table === 'Курсы') {
        $название = $_POST['название'];
        $язык = $_POST['язык'];
        $стоимость = $_POST['стоимость'];

        $query = $db->prepare("SELECT COUNT(*) FROM Курсы WHERE
название = ? AND язык = ? AND стоимость = ?");
        $query->execute([$название, $язык, $стоимость]);
        $exists = $query->fetchColumn();

        if ($exists) {
            echo "<p class='error'>Ошибка: Курс с таким названием, языком и
стоимостью уже существует.</p>";
            echo "<meta http-equiv='refresh'
content='3;url=?action=view&table=Курсы'>";
            exit();
        }
    }
}

$columnsList = implode(" ", $columns);
$placeholders = implode(" ", array_fill(0, count($columns), "?"));

```

```
        $stmt = $db->prepare("INSERT INTO $table ($columnsList) VALUES  
($placeholders)");  
        $stmt->execute($values);
```

```
header("Location: ?action=view&table=$table");  
exit();
```

```
    } catch (PDOException $e) {  
        echo "<p class='error'>Ошибка при добавлении записи: " .  
$e->getMessage() . "</p>";  
        echo "<a href='?action=view&table=$table'>Вернуться к таблице</a>";  
    }  
    } elseif ($action === 'queries') {  
        include('queries.php');  
    } else {  
        echo "<h2>Добро пожаловать!</h2>";  
        echo "<p>Выберите таблицу для просмотра или редактирования данных,  
или перейдите к разделу запросов.</p>";  
    }  
    ?>  
<?php  
if ($table === 'Студенты') {  
    echo "<a href='edit_students.php'>Редактировать студента</a>";  
}  
?>
```

```
</main>  
</div>  
</body>  
</html>
```

queries.php:

```
<?php
require_once 'db.php';
require_once 'functions.php';
```

```
$db = require 'db.php';
```

```
function renderQueryResult($result, ...$headers)
{
    if ($result) {
        echo "<table border='1'>";
        echo "<thead><tr>";
        foreach ($headers as $header) {
            echo "<th>$header</th>";
        }
        echo "</tr></thead>";
        echo "<tbody>";
        foreach ($result as $row) {
            echo "<tr>";
            foreach ($row as $col) {
                echo "<td>$col</td>";
            }
            echo "</tr>";
        }
        echo "</tbody></table>";
    } else {
        echo "<p>Нет данных для отображения.</p>";
    }
}
```

```
function getLanguages($db)
{
    $sql = "SELECT DISTINCT язык FROM Курсы";
    $stmt = $db->query($sql);
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}
```

```
echo "<h2>Запросы</h2>";
```

```
echo "<form method='post'>";  
echo "<h3>Запрос 1: Студенты и курсы</h3>";  
echo "<p>Вывести имена студентов и названия курсов, на которые они  
записаны, вместе с датой начала курса.</p>";  
echo "<button type='submit' name='query' value='query1'>Выполнить</button>";
```

```
echo "<h3>Запрос 2: Преподаватели и курсы на английском языке</h3>";  
echo "<p>Вывести имена преподавателей, которые ведут курсы на английском  
языке, и названия этих курсов.</p>";  
echo "<button type='submit' name='query' value='query2'>Выполнить</button>";
```

```
echo "<h3>Запрос 3: Студенты по выбранному языку</h3>";  
echo "<label for='language'>Выберите язык:</label>";  
echo "<select id='language' name='language'>";  
foreach (getLanguages($db) as $language) {  
    echo "<option value='$language'>$language</option>";  
}  
echo "</select>";  
echo "<button type='submit' name='query' value='query3'>Выполнить</button>";
```

```
echo "<h3>Запрос 4: Студенты записанные на курс до определенной  
даты</h3>";  
echo "<label for='start_date'>Выберите дату начала курса:</label>";  
echo "<input type='date' id='start_date' name='start_date'>";  
echo "<button type='submit' name='query' value='query4'>Выполнить</button>";  
echo "</form>";
```

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['query'])) {  
    $query = $_POST['query'];
```

```
    switch ($query) {  
        case 'query1':  
            $sql = "  
            SELECT
```



```

        Студенты.ФИО AS Имя_студента,
        Курсы.название AS Название_курса,
        Записи_на_курсы.дата_начала AS Дата_начала
FROM
    Студенты
JOIN
    Записи_на_курсы ON Студенты.id = Записи_на_курсы.студент_id
JOIN
    Курсы ON Курсы.id = Записи_на_курсы.курс_id";
$stmt = $db->query($sql);
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);
renderQueryResult($result, "Имя студента", "Название курса", "Дата
начала");
break;

```

```

case 'query2':
    $sql = "
        SELECT DISTINCT
        Преподаватели.ФИО AS Имя_преподавателя,
        Курсы.название AS Название_курса
FROM
    Преподаватели
JOIN
    Курсы ON Преподаватели.язык = Курсы.язык
WHERE
    Курсы.язык = 'Английский';
$stmt = $db->query($sql);
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);
renderQueryResult($result, "Имя преподавателя", "Название курса");
break;

```

```

case 'query3':
    if (!empty($_POST['language'])) {
        $language = trim($_POST['language']);
        $sql = "
            SELECT
            Студенты.ФИО AS Имя_студента,
            Курсы.название AS Название_курса,
            Преподаватели.ФИО AS Имя_преподавателя
FROM
    Студенты
JOIN

```

```

        Записи_на_курсы ON Студенты.id =
Записи_на_курсы.студент_id
        JOIN
        Курсы ON Курсы.id = Записи_на_курсы.курс_id
        JOIN
        Преподаватели ON Преподаватели.id =
Записи_на_курсы.преподаватель_id
        WHERE
        Курсы.язык = :language;
";
$stmt = $db->prepare($sql);
$stmt->execute([':language' => $language]);
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);
if ($result) {
    renderQueryResult($result, "Имя студента", "Название курса", "Имя
преподавателя");
} else {
    echo "<p>Нет студентов, изучающих курсы на языке:
$language.</p>";
}
} else {
    echo "<p>Выберите язык для выполнения запроса.</p>";
}
break;

```

```

case 'query4':
    if (!empty($_POST['start_date'])) {
        $startDate = trim($_POST['start_date']);
        $sql = "
        SELECT
        Студенты.ФИО AS Имя_студента,
        Курсы.название AS Название_курса,
        Записи_на_курсы.дата_начала AS Дата_начала
        FROM
        Студенты
        JOIN
        Записи_на_курсы ON Студенты.id = Записи_на_курсы.студент_id
        JOIN
        Курсы ON Курсы.id = Записи_на_курсы.курс_id
        WHERE
        Записи_на_курсы.дата_начала <= :start_date";
        $stmt = $db->prepare($sql);
        $stmt->execute([':start_date' => $startDate]);
    }

```

```

        $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
        if ($result) {
            renderQueryResult($result, "Имя студента", "Название курса", "Дата
начала");
        } else {
            echo "<p>Нет студентов, записанных на курсы до даты:
$startDate.</p>";
        }
        } else {
            echo "<p>Введите корректную дату для выполнения запроса.</p>";
        }
        break;

        default:
            echo "<p>Неизвестный запрос.</p>";
    }
}

```

```

echo "<a href='?'>Вернуться на главную</a>";
?>

```

db.php:

```

<?php
$path = 'D:/7_semak/BD/RGR/language.db';
try {
    $db = new PDO("sqlite:$path");
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $db->exec("CREATE TABLE IF NOT EXISTS Курсы (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        название TEXT NOT NULL,
        язык TEXT NOT NULL,
        стоимость REAL
    )");

    $db->exec("CREATE TABLE IF NOT EXISTS Преподаватели (
        id INTEGER PRIMARY KEY AUTOINCREMENT,

```

```

        ФИО TEXT NOT NULL,
        специализация TEXT NOT NULL,
        язык TEXT NOT NULL,
        опыт_работы INTEGER CHECK(опыт_работы >= 0)
    );

```

```

$db->exec("CREATE TABLE IF NOT EXISTS Студенты (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ФИО TEXT NOT NULL,
    возраст INTEGER CHECK(возраст >= 16 AND возраст <= 100),
    телефон TEXT NOT NULL
)");

```

```

$db->exec("CREATE TABLE IF NOT EXISTS Записи_на_курсы (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    студент_id INTEGER NOT NULL,
    курс_id INTEGER NOT NULL,
    преподаватель_id INTEGER NOT NULL,
    дата_начала TEXT NOT NULL,
    FOREIGN KEY(студент_id) REFERENCES Студенты(id),
    FOREIGN KEY(курс_id) REFERENCES Курсы(id),
    FOREIGN KEY(преподаватель_id) REFERENCES Преподаватели(id)
)");

```

```

    return $db;
} catch (PDOException $e) {
    die("Ошибка базы данных: " . $e->getMessage());
}
?>

```

functions.php:

```

<?php
function renderTable($records) {
    echo "<table>";
    if (!empty($records)) {
        echo "<tr>";
        foreach (array_keys($records[0]) as $col) {
            echo "<th>$col</th>";

```

```

    }
    echo "</tr>";
    foreach ($records as $row) {
        echo "<tr>";
        foreach ($row as $col) {
            echo "<td>$col</td>";
        }
        echo "</tr>";
    }
} else {
    echo "<tr><td colspan='100%'>Нет данных для отображения</td></tr>";
}
echo "</table>";
}

```

```

function getColumn($table) {
    switch ($table) {
        case 'Курсы':
            return ['название', 'язык', 'стоимость'];
        case 'Преподаватели':
            return ['ФИО', 'специализация', 'язык', 'опыт_работы'];
        case 'Студенты':
            return ['ФИО', 'возраст', 'телефон'];
        case 'Записи_на_курсы':
            return ['студент_id', 'курс_id', 'преподаватель_id', 'дата_начала'];
        default:
            return [];
    }
}

```

```

function checkTeacherExists($fio, $language, $db) {
    $query = $db->prepare("SELECT id FROM Преподаватели WHERE ФИО = ?
AND язык = ?");
    $query->execute([$fio, $language]);
    return $query->fetch(PDO::FETCH_ASSOC);
}
?>

```

edit_students:

```
<?php
require_once 'db.php';
require_once 'functions.php';

$db = require 'db.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $id = $_POST['student_id'];
    $name = $_POST['name'];
    $age = $_POST['age'];
    $phone = $_POST['phone'];

    try {
        $stmt = $db->prepare("UPDATE Студенты SET ФИО = ?, возраст = ?,
        телефон = ? WHERE id = ?");
        $stmt->execute([$name, $age, $phone, $id]);

        header("Location: ?action=view&table=Студенты");
        exit();
    } catch (PDOException $e) {
        die("<p class='error'>Ошибка: {$e->getMessage()}</p>");
    }
}

if ($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['student_id'])) {
    $id = $_GET['student_id'];
    $stmt = $db->prepare("SELECT ФИО, возраст, телефон FROM Студенты
    WHERE id = ?");
    $stmt->execute([$id]);
    $student = $stmt->fetch(PDO::FETCH_ASSOC);
    echo json_encode($student);
    exit();
}

$students = $db->query("SELECT id, ФИО FROM
Студенты")->fetchAll(PDO::FETCH_ASSOC);
```

```

?>
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Редактировать Студента</title>
  <link rel="stylesheet" href="style.css">
  <script>
    async function fetchStudentData(studentId) {
      if (!studentId) return;

      const response = await fetch(`edit_students.php?student_id=${studentId}`);
      const data = await response.json();

      document.getElementById('name').value = data.ФИО || "";
      document.getElementById('age').value = data.возраст || "";
      document.getElementById('phone').value = data.телефон || "";
    }
  </script>
</head>
<body>
<div class="container">
  <h1>Редактировать студента</h1>
  <form method="post">
    <label for="student_id">Выберите студента:</label>
    <select id="student_id" name="student_id" required
onchange="fetchStudentData(this.value)">
      <option value="">Выберите студента</option>
      <?php foreach ($students as $student): ?>
        <option value="<?php echo $student['id']; ?>"><?php echo
$student['ФИО']; ?></option>
      <?php endforeach; ?>
    </select>

    <label for="name">ФИО:</label>
    <input type="text" id="name" name="name" required>

    <label for="age">Возраст:</label>
    <input type="number" id="age" name="age" min="16" max="100" required>

```

```
<label for="phone">Телефон:</label>
<input type="tel" id="phone" name="phone" pattern="^[0-9\s\-\+\(\)]*$"
placeholder="Пример: +1 (234) 567-8901" required>

<button type="submit">Сохранить</button>
</form>

<a href="index.php?action=view&table=Студенты">Вернуться к таблице</a>

</div>
</body>
</html>
```

style.css:

```
:root {
  --primary: #4b4b4b;
  --secondary: #8e8e8e;
  --background: #2d2d2d;
  --text: #000000;
  --white: #ffffff;
  --border: #e0e0e0;
  --button-hover: #485049;
  --error: #f44336;
  --success: #4CAF50;
  --link: #e42323;
  --link-hover: #e42323;
}
```

```
*,
*::before,
*::after {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
```



```
}
```

```
body {  
  font-family: 'Helvetica Neue', sans-serif;  
  background-color: var(--background);  
  color: var(--text);  
}
```

```
.container {  
  max-width: 960px;  
  margin: 30px auto;  
  background: var(--white);  
  padding: 40px;  
  border-radius: 8px;  
  box-shadow: 0 2px 12px rgba(0, 0, 0, 0.1);  
}
```

```
header {  
  background-color: var(--primary);  
  color: var(--white);  
  padding: 20px 30px;  
  margin-bottom: 30px;  
  border-radius: 10px;  
  box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1);  
  display: flex;  
  flex-direction: column;  
  align-items: flex-start;  
}
```

```
header h1 {  
  font-size: 2rem;  
  font-weight: 700;  
}
```

```
nav ul {  
  list-style: none;  
  padding: 0;  
  margin-top: 20px;  
  display: flex;
```

```
    gap: 20px;
    flex-wrap: wrap;
}
```

```
nav a {
    text-decoration: none;
    color: var(--white);
    font-weight: 500;
    padding: 10px 15px;
    border-radius: 6px;
    transition: background-color 0.3s, color 0.3s;
}
```

```
nav a:hover,
nav a.active {
    background-color: var(--link-hover);
    color: var(--white);
}
```

```
h1, h2, h3 {
    color: var(--text);
}
```

```
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
    border-radius: 8px;
    overflow: hidden;
}
```

```
th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid var(--border);
}
```

```
th {
```

```
background-color: var(--primary);
color: var(--white);
font-weight: bold;
}
```

```
td {
  background-color: var(--white);
  color: var(--text);
}
```

```
input[type="text"],
input[type="submit"],
input[type="date"],
select,
input[type="number"],
input[type="tel"] {
  width: 100%;
  padding: 12px;
  margin-top: 10px;
  border: 1px solid var(--border);
  border-radius: 6px;
  background-color: #fafafa;
  font-size: 1rem;
  color: var(--text);
}
```

```
input[type="submit"] {
  background-color: var(--primary);
  color: var(--white);
  cursor: pointer;
  border: none;
  padding: 12px 20px;
  font-size: 1.1rem;
  font-weight: 500;
  border-radius: 6px;
  transition: background-color 0.3s, transform 0.2s;
}
```

```
input[type="submit"]:hover {
  background-color: var(--button-hover);
}
```

```
    transform: scale(1.05);  
}
```

```
a {  
    display: inline-block;  
    margin-top: 20px;  
    text-decoration: none;  
    color: var(--link);  
    font-weight: 500;  
}
```

```
a:hover {  
    text-decoration: underline;  
    color: var(--link-hover);  
}
```

```
.error {  
    color: var(--error);  
    font-size: 1rem;  
    margin-top: 10px;  
}
```

```
.success {  
    color: var(--success);  
    font-size: 1rem;  
    margin-top: 10px;  
}
```

```
select option {  
    padding: 12px;  
    font-size: 1rem;  
}
```

```
tr:hover {  
    background-color: #f9f9f9;  
}
```

```
form label {  
  display: block;  
  margin-bottom: 6px;  
  font-weight: 600;  
  font-size: 1rem;  
  color: var(--text);  
}
```

```
form {  
  margin-top: 20px;  
}
```

```
form h3 {  
  font-size: 1.2rem;  
  margin-bottom: 10px;  
  color: var(--text);  
  font-weight: 600;  
}
```

```
form p {  
  font-size: 1rem;  
  margin-bottom: 20px;  
  color: var(--text);  
}
```

```
button[type="submit"] {  
  background-color: var(--primary);  
  color: var(--white);  
  padding: 10px 15px;  
  border-radius: 6px;  
  border: none;  
  cursor: pointer;  
  transition: background-color 0.3s, transform 0.2s;  
  font-size: 1rem;  
  margin-right: 10px;  
}
```

```
button[type="submit"]:hover {  
  background-color: var(--button-hover);  
}
```

```
    transform: scale(1.05);  
}
```

```
button[type="submit"]:focus {  
    outline: none;  
}
```

```
input[type="text"] {  
    width: 100%;  
    padding: 12px;  
    margin-top: 10px;  
    border-radius: 6px;  
    border: 1px solid var(--border);  
    background-color: #fafafa;  
    font-size: 1rem;  
}
```

```
input[type="date"] {  
    width: 100%;  
    padding: 12px;  
    margin-top: 10px;  
    border-radius: 6px;  
    border: 1px solid var(--border);  
    background-color: #fafafa;  
    font-size: 1rem;  
}
```

```
table.query-result {  
    width: 100%;  
    border-collapse: collapse;  
    margin-top: 30px;  
    border-radius: 8px;  
    overflow: hidden;  
    box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1);  
}
```

```
table.query-result th, table.query-result td {  
    padding: 12px;  
    text-align: left;
```

```
border-bottom: 1px solid var(--border);  
}
```

```
table.query-result th {  
  background-color: var(--primary);  
  color: var(--white);  
  font-weight: bold;  
}
```

```
table.query-result td {  
  background-color: var(--white);  
  color: var(--text);  
}
```

```
table.query-result tr:nth-child(even) {  
  background-color: #f9f9f9;  
}
```

```
table.query-result tr:hover {  
  background-color: #f1f1f1;  
}
```

```
a.back-link {  
  display: inline-block;  
  margin-top: 20px;  
  text-decoration: none;  
  color: var(--link);  
  font-weight: 500;  
  transition: color 0.3s;  
}
```

```
a.back-link:hover {  
  color: var(--link-hover);  
  text-decoration: underline;  
}
```

```
a.back-link:focus {
```

```
    outline: none;
}
```

```
h2.query-title {
    font-size: 1.6rem;
    margin-top: 40px;
    color: var(--text);
    font-weight: 700;
}
```

```
@media (max-width: 768px) {
    .container {
        max-width: 90%;
        padding: 20px;
    }
    header {
        padding: 15px 20px;
    }
    header h1 {
        font-size: 1.5rem;
    }
    nav ul {
        flex-direction: column;
        align-items: center;
    }
    nav a {
        font-size: 1rem;
    }
    table {
        font-size: 0.9rem;
    }
}
```

```
@media (max-width: 480px) {
    header h1 {
        font-size: 1.3rem;
    }

    nav a {
        font-size: 0.9rem;
    }
}
```