Quiz App Code

Html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
 <title>Beginner SQL Quiz</title>
 <link rel="stylesheet" href="style.css" />
</head>
<body>
 <div class="quiz-container">
  <h1>Beginner SQL Quiz</h1>

  <div class="progress-bar">
   <div id="progress-fill"></div>
  </div>

  <div id="timer">Time left: <span id="time">30</span>s</div>

  <div id="question">Question text will appear here</div>

  <ul id="options"></ul>
```

```
    <button id="next-btn">Next</button>

    <div id="result" class="hidden">
      <h2>Your Score: <span id="score"></span> / 25</h2>
      <button onclick="location.reload()">Restart Quiz</button>
    </div>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

- **CSS**

```
* {
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  background: #f4f4f4;
  padding: 10px;
  margin: 0;
}
```

```
.quiz-container {
  width: 100%;
  max-width: 500px;
  margin: 0 auto;
  background: white;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0,0,0,0.2);
}

h1 {
  font-size: 24px;
  margin-bottom: 10px;
}

.progress-bar {
  width: 100%;
  background-color: #ddd;
  border-radius: 5px;
  margin-bottom: 15px;
  height: 12px;
  overflow: hidden;
}

#progress-fill {
  width: 0%;
```

```
  height: 100%;
  background-color: #4caf50;
  transition: width 0.3s ease;
}

#timer {
  font-size: 16px;
  margin-bottom: 15px;
  color: #333;
}

#question {
  font-size: 18px;
  margin-bottom: 20px;
}

#options {
  list-style-type: none;
  padding: 0;
}

#options li {
  margin: 10px 0;
  padding: 12px;
  background: #eee;
  border-radius: 5px;
```

```
  cursor: pointer;
  transition: background 0.3s ease;
}

#options li:hover {
  background: #ddd;
}

#next-btn, #result button {
  width: 100%;
  padding: 12px;
  font-size: 16px;
  margin-top: 20px;
  cursor: pointer;
  border: none;
  background: #4caf50;
  color: white;
  border-radius: 5px;
  transition: background 0.3s;
}

#next-btn:hover, #result button:hover {
  background: #45a049;
}

.hidden {
```

```
  display: none;
}

/* Mobile-Friendly Adjustments */
@media (max-width: 600px) {
  .quiz-container {
    padding: 15px;
  }

  h1 {
    font-size: 20px;
  }

  #question {
    font-size: 16px;
  }

  #options li {
    padding: 10px;
    font-size: 14px;
  }

  #next-btn, #result button {
    font-size: 14px;
    padding: 10px;
  }
```

```
 #timer {
  font-size: 14px;
 }
}
```

Java Script

```
const quizData = [
 {
  question: "What is SQL?",
  options: [
   "Structured Query Language",
   "Simple Query Language",
   "Strong Question Language",
   "Standard Question Logic"
  ],
  answer: "Structured Query Language"
 },
 {
  question: "What is a Database?",
  options: [
   "A program to design apps",
   "A collection of interrelated data",
   "A type of spreadsheet",
   "None of the above"
```

```
  ],
  answer: "A collection of interrelated data"
},
{
  question: "What are the types of SQL commands?",
  options: [
    "DDL, DML, DCL, TCL, DQL",
    "HTML, CSS, JS, SQL",
    "Insert, Delete, Create, Select",
    "None of the above"
  ],
  answer: "DDL, DML, DCL, TCL, DQL"
},
{
  question: "What is Primary Key?",
  options: [
    "A key that can be NULL",
    "A key that uniquely identifies each record",
    "A key that links tables",
    "A duplicate field"
  ],
  answer: "A key that uniquely identifies each record"
},
{
  question: "What is Foreign Key?",
  options: [
```

```
    "A key that stores foreign data",
    "A key that refers to the primary key of another table",
    "A random number",
    "None of the above"
  ],
  answer: "A key that refers to the primary key of another table"
},
{
  question: "What is UNIQUE Key?",
  options: [
    "Allows duplicate values",
    "Ensures all values are unique in the column",
    "Used for encryption",
    "Used for sorting"
  ],
  answer: "Ensures all values are unique in the column"
},
{
  question: "Difference between Primary Key and UNIQUE Key?",
  options: [
    "Primary allows NULL, UNIQUE doesn't",
    "Both allow duplicates",
    "Primary Key cannot be NULL, UNIQUE can be",
    "They are the same"
  ],
  answer: "Primary Key cannot be NULL, UNIQUE can be"
```

```
},
{
  question: "What is NOT NULL constraint?",
  options: [
    "Allows NULL values",
    "Prevents a column from having NULL values",
    "Used to delete columns",
    "Used for constraints"
  ],
  answer: "Prevents a column from having NULL values"
},
{
  question: "What is Default Constraint?",
  options: [
    "Assigns a default value if none is provided",
    "Deletes default values",
    "Sets default keys",
    "None of the above"
  ],
  answer: "Assigns a default value if none is provided"
},
{
  question: "Difference between DELETE, TRUNCATE, and DROP?",
  options: [
    "DELETE removes structure, TRUNCATE keeps it",
    "DROP removes rows, DELETE keeps them",
```

```
    "DELETE removes rows with WHERE, TRUNCATE removes all rows, DROP removes table",
    "They are all the same"
   ],
   answer: "DELETE removes rows with WHERE, TRUNCATE removes all rows, DROP removes table"
  },
  {
   question: "Difference between WHERE and HAVING?",
   options: [
    "WHERE is used with groups, HAVING with columns",
    "HAVING is used with aggregate functions, WHERE is not",
    "Both are same",
    "WHERE is after GROUP BY"
   ],
   answer: "HAVING is used with aggregate functions, WHERE is not"
  },
  {
   question: "What are Joins in SQL?",
   options: [
    "To combine rows from multiple tables",
    "To delete tables",
    "To index data",
    "None of the above"
   ],
   answer: "To combine rows from multiple tables"
  },
```

```
{
  question: "What is INNER JOIN?",
  options: [
    "Returns all rows",
    "Returns rows with matching values in both tables",
    "Returns unmatched rows",
    "Joins columns"
  ],
  answer: "Returns rows with matching values in both tables"
},
{
  question: "What is LEFT JOIN?",
  options: [
    "Returns only unmatched rows",
    "Returns all rows from left and matching from right",
    "Returns all from right",
    "None"
  ],
  answer: "Returns all rows from left and matching from right"
},
{
  question: "What is RIGHT JOIN?",
  options: [
    "Returns all from left",
    "Returns matching rows from right only",
    "Returns all rows from right and matching from left",
```

```
    "None"
  ],
  answer: "Returns all rows from right and matching from left"
},
{
  question: "What is FULL JOIN?",
  options: [
    "Returns only matching rows",
    "Returns all rows when there is a match in one of the tables",
    "Deletes data",
    "Only from left"
  ],
  answer: "Returns all rows when there is a match in one of the tables"
},
{
  question: "What is Self Join?",
  options: [
    "Joining table with itself",
    "Joining two different databases",
    "Deleting same table",
    "None"
  ],
  answer: "Joining table with itself"
},
{
  question: "What is Cross Join?",
```

```
  options: [
    "Returns rows with conditions",
    "Combines each row of the first table with all rows of the second",
    "Returns common rows",
    "Deletes duplicates"
  ],
  answer: "Combines each row of the first table with all rows of the second"
},
{
  question: "What is Union and Union All?",
  options: [
    "Used for subtraction",
    "Combine result sets; UNION removes duplicates, UNION ALL doesn't",
    "Joins tables",
    "Deletes rows"
  ],
  answer: "Combine result sets; UNION removes duplicates, UNION ALL doesn't"
},
{
  question: "Difference between UNION and UNION ALL?",
  options: [
    "UNION keeps duplicates, UNION ALL removes",
    "UNION removes duplicates, UNION ALL keeps all",
    "Both are same",
    "Used for joins"
  ],
```

```
  answer: "UNION removes duplicates, UNION ALL keeps all"
},
{
  question: "What is Normalization?",
  options: [
    "Combining tables",
    "Organizing data to reduce redundancy",
    "Removing data",
    "Encrypting data"
  ],
  answer: "Organizing data to reduce redundancy"
},
{
  question: "What is Denormalization?",
  options: [
    "Splitting tables",
    "Optimizing performance by adding redundancy",
    "Securing data",
    "None"
  ],
  answer: "Optimizing performance by adding redundancy"
},
{
  question: "Difference between CHAR and VARCHAR?",
  options: [
    "CHAR is fixed-length, VARCHAR is variable-length",
```

```
    "Both are same",
    "VARCHAR is slower",
    "CHAR uses less space"
  ],
  answer: "CHAR is fixed-length, VARCHAR is variable-length"
},
{
  question: "Difference between SQL and MySQL?",
  options: [
    "SQL is a database, MySQL is a language",
    "SQL is a query language, MySQL is a database software",
    "Both are databases",
    "MySQL is older"
  ],
  answer: "SQL is a query language, MySQL is a database software"
},
{
  question: "What is Auto Increment in SQL?",
  options: [
    "Automatically deletes records",
    "Generates a unique number when a new record is inserted",
    "Combines fields",
    "Sorts records"
  ],
  answer: "Generates a unique number when a new record is inserted"
}
```

```
];

let currentQuestion = 0;
let score = 0;
let timeLeft = 30;
let timer;

const questionEl = document.getElementById("question");
const optionsEl = document.getElementById("options");
const nextBtn = document.getElementById("next-btn");
const timerEl = document.getElementById("time");
const progressFill = document.getElementById("progress-fill");
const resultEl = document.getElementById("result");
const scoreEl = document.getElementById("score");

function loadQuestion() {
  clearInterval(timer);
  startTimer();
  const currentQuiz = quizData[currentQuestion];
  questionEl.textContent = `${currentQuestion + 1}. ${currentQuiz.question}`;
  optionsEl.innerHTML = "";

  currentQuiz.options.forEach(option => {
    const li = document.createElement("li");
    li.textContent = option;
    li.classList.add("option");
```

```
    li.addEventListener("click", () => selectOption(li, currentQuiz.answer));
    optionsEl.appendChild(li);
  });

  updateProgressBar();
}

function selectOption(selectedLi, correctAnswer) {
  const options = document.querySelectorAll(".option");
  options.forEach(option => {
    option.classList.remove("selected");
    option.style.pointerEvents = "none";
  });

  selectedLi.classList.add("selected");

  if (selectedLi.textContent === correctAnswer) {
    score++;
  }
  clearInterval(timer);
}

function startTimer() {
  timeLeft = 30;
  timerEl.textContent = timeLeft;
  timer = setInterval(() => {
```

```
    timeLeft--;
    timerEl.textContent = timeLeft;
    if (timeLeft === 0) {
      clearInterval(timer);
      nextQuestion();
    }
  }, 1000);
}

function nextQuestion() {
  if (currentQuestion < quizData.length - 1) {
    currentQuestion++;
    loadQuestion();
  } else {
    showResult();
  }
}

function updateProgressBar() {
  const percent = ((currentQuestion + 1) / quizData.length) * 100;
  progressFill.style.width = `${percent}%`;
}

function showResult() {
  document.querySelector(".quiz-container").classList.add("finished");
  questionEl.style.display = "none";
```

```
  optionsEl.style.display = "none";
  nextBtn.style.display = "none";
  timerEl.parentElement.style.display = "none";
  document.querySelector(".progress-bar").style.display = "none";

  resultEl.classList.remove("hidden");
  scoreEl.textContent = score;
}

nextBtn.addEventListener("click", nextQuestion);

window.addEventListener("load", () => {
  loadQuestion();
});
```