MINI PROJECT REPORT

ON

" Interactive Visualization of Database Join Operations "

Submitted By :-

Name :- Moirangthem Satyabrata

UID :- 24MCA20343

Branch / Semester :- MCA 3rd

Subject :- Business Analytics

Subject Code :- 24CAH - 703

Date of Performance :- 24/10/2025

Submitted To :-

Ms. Reeti Jaswal

Assistant Professor

Date of Submission: 01/11/2025

University Institute of Computing

Chandigarh University, Gharuan, Mohali

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

DECLARATION

We hereby declare that this submission is our own work and that, to the best of mine knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award for any other degree or diploma of the university or other institute of higher learning, except where acknowledgement has been made in the text.

Name :- Moirangthem Satyabrata

UID :- 24MCA20343

Session 2024 – 25

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

CERTIFICATE

It is to certify that Moirangthem Satyabrata of class MCA 2nd Year under University Roll Number :- 24MCA20343 has completed the project titled " Interactive Visualization of Database Join Operations " in which the language used is React + TypeScript, Vite, Tailwind CSS, PostCSS, ESLint, and Node.js + npm for the Masters of Computer Application 3rd Semester under my supervision. The work done in project is a result of the candidate's own efforts and report maintains is satisfied as per requirement.

Project Guide :-

Ms. Reeti Jaswal

INDEX

Abstract

The project **"Interactive Visualization of Database Join Operations"** aims to simplify the understanding of SQL join concepts through dynamic and engaging visual representation. SQL joins are fundamental in relational databases, allowing users to combine data from multiple tables based on related columns. However, understanding how each join (such as Inner, Left, Right, Full Outer, and Cross Join) behaves can often be confusing for beginners. This project tackles that challenge by using interactive visual tools to show how records from different tables merge under various join operations, making the learning process more intuitive and engaging.

Furthermore, the system visually represents the matched and unmatched data from two or more tables, allowing users to clearly identify how SQL queries affect the output. By enabling users to interact with visual elements, the project enhances both theoretical understanding and practical knowledge of SQL joins. It serves as a valuable learning tool for students, educators, and professionals in database management, helping them strengthen their foundation in data manipulation and relational concepts.

Objective

The main objective of this project is to design and develop an interactive educational tool that visually explains how different SQL joins operate within relational databases. It aims to bridge the gap between theoretical understanding and practical application by demonstrating how data is combined, filtered, and represented in various join types. This visualization will help users clearly identify how records are selected or excluded in each case, which is often difficult to comprehend through text-based SQL queries alone.

Additionally, the project seeks to make database learning more engaging and accessible by using intuitive visuals and animations. It focuses on improving conceptual clarity, supporting self-paced learning, and providing hands-on experience with join operations. Another goal is to create a tool that can be integrated into academic environments or used independently by learners to practice and test their understanding. Overall, the project aims to strengthen the foundation of SQL knowledge, making it easier to work with complex datasets in real-world applications.

The final objective is to develop a scalable and adaptable registration form that can be integrated into a variety of web platforms and applications. Whether it's a small business website, a large-scale e-commerce platform, or an educational portal, the registration form should be easily customizable and scalable to meet the needs of different users. Additionally, the form will be designed with security best practices in mind, ensuring compliance with data protection standards and creating a seamless and trustworthy experience for all users.

Technology Used

React + TypeScript, Vite, Tailwind CSS, PostCSS, ESLint, and Node.js + npm **:-**

React :- A JavaScript library for building fast, interactive user interfaces using components.

TypeScript :- A superset of JavaScript that adds static typing for better error checking and cleaner code.

Vite :- A modern, ultra-fast build tool and development server for web projects. It speeds up development with instant updates and optimized builds.

Tailwind CSS :- A utility-first CSS framework that lets you style websites quickly using pre-defined classes directly in HTML or JSX.

PostCSS :- A CSS processing tool that transforms modern CSS into browser-compatible styles and adds extra features like autoprefixing.

ESLint :- A tool that analyzes JavaScript/TypeScript code to find and fix problems, ensuring consistent coding standards and fewer bugs.

Node.js :- A runtime that lets you run JavaScript outside the browser.

npm :- A package manager used to install and manage project dependencies.

Code

HTML :-

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>SQL Join Venn Diagram Visualizer</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

React :- App.tsx

```
import { useState } from 'react';
import JoinVisualizer from './components/JoinVisualizer';
import JoinSelector from './components/JoinSelector';
import DataPreview from './components/DataPreview';
import { Database, Sparkles } from 'lucide-react';

export type JoinType = 'inner' | 'left' | 'right' | 'full' | 'cross';

function App() {
  const [selectedJoin, setSelectedJoin] = useState<JoinType>('inner');
```

```
 return (
   <div className="min-h-screen bg-gradient-to-br from-slate-900 via-blue-900 to-slate-900">
     <div className="absolute inset-0 bg-[url('')] opacity-40"></div>

     <div className="relative z-10">
       <header className="text-center pt-12 pb-8 px-4">
         <div className="flex items-center justify-center gap-3 mb-4">
           <Database className="w-12 h-12 text-cyan-400 animate-pulse" />
           <h1 className="text-5xl font-bold text-white tracking-tight">
             SQL Joins
           </h1>
           <Sparkles className="w-12 h-12 text-yellow-400 animate-pulse" />
         </div>
         <p className="text-xl text-cyan-300 font-light">
           Interactive Visualization of Database Join Operations
         </p>
       </header>

       <div className="max-w-7xl mx-auto px-4 pb-16">
         <JoinSelector selectedJoin={selectedJoin} onSelectJoin={setSelectedJoin} />
         <JoinVisualizer joinType={selectedJoin} />
         <DataPreview joinType={selectedJoin} />
```

```
        </div>
      </div>
    </div>
  );
}

export default App;
```

main.tsx :-

```
import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import App from './App.tsx';
import './index.css';

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

DataPreview.tsx :-

```
import { JoinType } from '../App';
import { getJoinData } from '../utils/joinData';

interface DataPreviewProps {
  joinType: JoinType;
}
```

```
export default function DataPreview({ joinType }: DataPreviewProps) {
  const data = getJoinData(joinType);

  return (
    <div className="bg-white/10 backdrop-blur-lg rounded-3xl p-8 border border-white/20 shadow-2xl">
      <h3 className="text-2xl font-bold text-white mb-6 text-center">Sample Data Result</h3>

      <div className="grid md:grid-cols-3 gap-6 mb-8">
        <div className="bg-blue-900/30 rounded-xl p-4 border border-blue-400/50">
          <h4 className="text-blue-300 font-bold mb-3 text-center">Table A: Employees</h4>
          <div className="space-y-2">
            {data.tableA.map((item, idx) => (
              <div key={idx} className="bg-blue-800/40 rounded-lg p-2 text-sm">
                <div className="text-white font-semibold">{item.name}</div>
                <div className="text-blue-200">Dept: {item.deptId}</div>
              </div>
            ))}
          </div>
        </div>

        <div className="bg-green-900/30 rounded-xl p-4 border border-green-400/50">
          <h4 className="text-green-300 font-bold mb-3 text-center">Table B: Departments</h4>
          <div className="space-y-2">
            {data.tableB.map((item, idx) => (
              <div key={idx} className="bg-green-800/40 rounded-lg p-2 text-sm">
                <div className="text-white font-semibold">{item.name}</div>
                <div className="text-green-200">ID: {item.id}</div>
```

```
                </div>
              ))}
            </div>
          </div>

          <div className="bg-purple-900/30 rounded-xl p-4 border border-purple-400/50">
            <h4 className="text-purple-300 font-bold mb-3 text-center">Result ({data.result.length} rows)</h4>
            <div className="space-y-2 max-h-64 overflow-y-auto">
              {data.result.map((item, idx) => (
                <div key={idx} className="bg-purple-800/40 rounded-lg p-2 text-sm">
                  <div className="text-white font-semibold">{item.employee || '—'}</div>
                  <div className="text-purple-200">{item.department || '—'}</div>
                </div>
              ))}
            </div>
          </div>

          <div className="bg-gradient-to-r from-cyan-500/20 to-blue-500/20 rounded-xl p-4 border border-cyan-400/30">
            <p className="text-cyan-100 text-center">
              <span className="font-bold text-cyan-300">Result Count:</span> {data.result.length} row{data.result.length !== 1 ? 's' : ''} returned
            </p>
          </div>
        </div>
      );
    }
```

JoinSelector.tsx :-

```
import { JoinType } from '../App';

interface JoinSelectorProps {
  selectedJoin: JoinType;
  onSelectJoin: (join: JoinType) => void;
}

const joins = [
  { type: 'inner' as JoinType, label: 'INNER JOIN', color: 'from-purple-500 to-pink-500' },
  { type: 'left' as JoinType, label: 'LEFT JOIN', color: 'from-blue-500 to-cyan-500' },
  { type: 'right' as JoinType, label: 'RIGHT JOIN', color: 'from-green-500 to-emerald-500' },
  { type: 'full' as JoinType, label: 'FULL OUTER JOIN', color: 'from-orange-500 to-red-500' },
  { type: 'cross' as JoinType, label: 'CROSS JOIN', color: 'from-yellow-500 to-amber-500' },
];

export default function JoinSelector({ selectedJoin, onSelectJoin }: JoinSelectorProps) {
  return (
    <div className="flex flex-wrap justify-center gap-4 mb-12">
      {joins.map(({ type, label, color }) => (
        <button
          key={type}
          onClick={() => onSelectJoin(type)}
          className={`
            relative px-8 py-4 rounded-xl font-bold text-white text-lg
```

```
            transform transition-all duration-300 hover:scale-110 hover:shadow-2xl
            ${selectedJoin === type ? 'scale-110 shadow-2xl' : 'scale-100 opacity-70'}
          `}
        >
          <div className={`absolute inset-0 bg-gradient-to-r ${color} rounded-xl`}></div>
          <div className="relative z-10">{label}</div>
          {selectedJoin === type && (
            <div className="absolute -inset-1 bg-gradient-to-r from-white to-transparent opacity-30 rounded-xl animate-pulse"></div>
          )}
        </button>
      ))}
    </div>
  );
}
```

Join Visualizer.tsx :-

```
import { JoinType } from '../App';
import VennDiagram from './VennDiagram';
import { getJoinDescription } from '../utils/joinDescriptions';

interface JoinVisualizerProps {
  joinType: JoinType;
}

export default function JoinVisualizer({ joinType }: JoinVisualizerProps) {
  const description = getJoinDescription(joinType);

  return (
```

```
          <div className="bg-white/10 backdrop-blur-lg rounded-3xl p-8 mb-8 border
border-white/20 shadow-2xl">
            <div className="grid md:grid-cols-2 gap-8 items-center">
              <div className="order-2 md:order-1">
                <h2 className="text-3xl font-bold text-white mb-
4">{description.title}</h2>
                <p className="text-cyan-200 text-lg mb-6 leading-relaxed">
                  {description.description}
                </p>
                <div className="bg-slate-800/50 rounded-xl p-4 border border-cyan-
500/30">
                  <code className="text-cyan-300 font-mono text-sm block">
                    {description.sqlExample}
                  </code>
                </div>
                <div className="mt-6 bg-gradient-to-r from-blue-500/20 to-purple-500/20
rounded-xl p-4 border border-blue-400/30">
                  <p className="text-blue-200 font-semibold">💡 Use Case:</p>
                  <p className="text-blue-100 mt-2">{description.useCase}</p>
                </div>
              </div>
              <div className="order-1 md:order-2">
                <VennDiagram joinType={joinType} />
              </div>
            </div>
          </div>
        );
      }
```

VennDiagram.tsx :-

```
import { useEffect, useState } from 'react';
import { JoinType } from '../App';

interface VennDiagramProps {
  joinType: JoinType;
}

export default function VennDiagram({ joinType }: VennDiagramProps) {
  const [animate, setAnimate] = useState(false);

  useEffect(() => {
    setAnimate(false);
    const timer = setTimeout(() => setAnimate(true), 50);
    return () => clearTimeout(timer);
  }, [joinType]);

  const getHighlightedAreas = () => {
    switch (joinType) {
      case 'inner':
        return { leftOnly: false, intersection: true, rightOnly: false };
      case 'left':
        return { leftOnly: true, intersection: true, rightOnly: false };
      case 'right':
        return { leftOnly: false, intersection: true, rightOnly: true };
      case 'full':
        return { leftOnly: true, intersection: true, rightOnly: true };
      case 'cross':
        return { leftOnly: true, intersection: true, rightOnly: true };
    }
```

```
};

const highlighted = getHighlightedAreas();

return (
  <div className="relative w-full aspect-square max-w-md mx-auto">
    <svg viewBox="0 0 400 300" className="w-full h-full">
      <defs>
        <linearGradient id="leftGradient" x1="0%" y1="0%" x2="100%"
y2="100%">
          <stop offset="0%" style={{ stopColor: '#3b82f6', stopOpacity: 0.7 }} />
          <stop offset="100%" style={{ stopColor: '#06b6d4', stopOpacity: 0.7 }} />
        </linearGradient>
        <linearGradient id="rightGradient" x1="0%" y1="0%" x2="100%"
y2="100%">
          <stop offset="0%" style={{ stopColor: '#10b981', stopOpacity: 0.7 }} />
          <stop offset="100%" style={{ stopColor: '#34d399', stopOpacity: 0.7 }} />
        </linearGradient>
        <linearGradient id="intersectionGradient" x1="0%" y1="0%" x2="100%"
y2="100%">
          <stop offset="0%" style={{ stopColor: '#a855f7', stopOpacity: 0.9 }} />
          <stop offset="100%" style={{ stopColor: '#ec4899', stopOpacity: 0.9 }} />
        </linearGradient>
      </defs>

      {joinType === 'cross' ? (
        <g className={`transition-opacity duration-500 ${animate ? 'opacity-100' :
'opacity-0'}`}>
          <rect
            x="50"
            y="50"
            width="120"
```

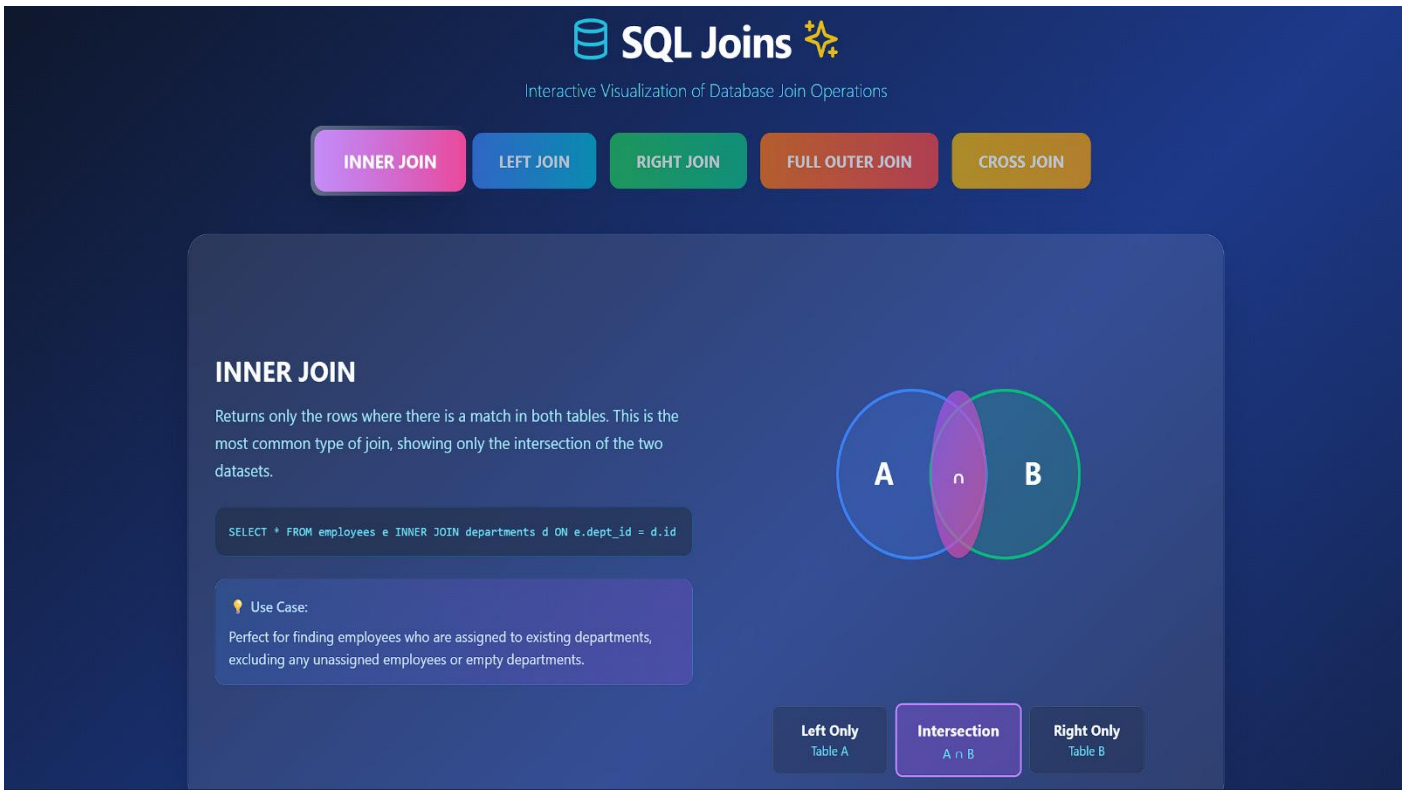
```
                height="200"
                fill="url(#leftGradient)"
                stroke="#3b82f6"
                strokeWidth="3"
                rx="10"
                className="animate-pulse"
              />
              <rect
                x="230"
                y="50"
                width="120"
                height="200"
                fill="url(#rightGradient)"
                stroke="#10b981"
                strokeWidth="3"
                rx="10"
                className="animate-pulse"
                style={{ animationDelay: '0.2s' }}
              />
              <text x="110" y="160" textAnchor="middle" className="fill-white font-bold text-2xl">
                A
              </text>
              <text x="290" y="160" textAnchor="middle" className="fill-white font-bold text-2xl">
                B
              </text>
              <text x="200" y="280" textAnchor="middle" className="fill-cyan-300 font-bold text-lg">
                Every row × Every row
              </text>
            </g>
```

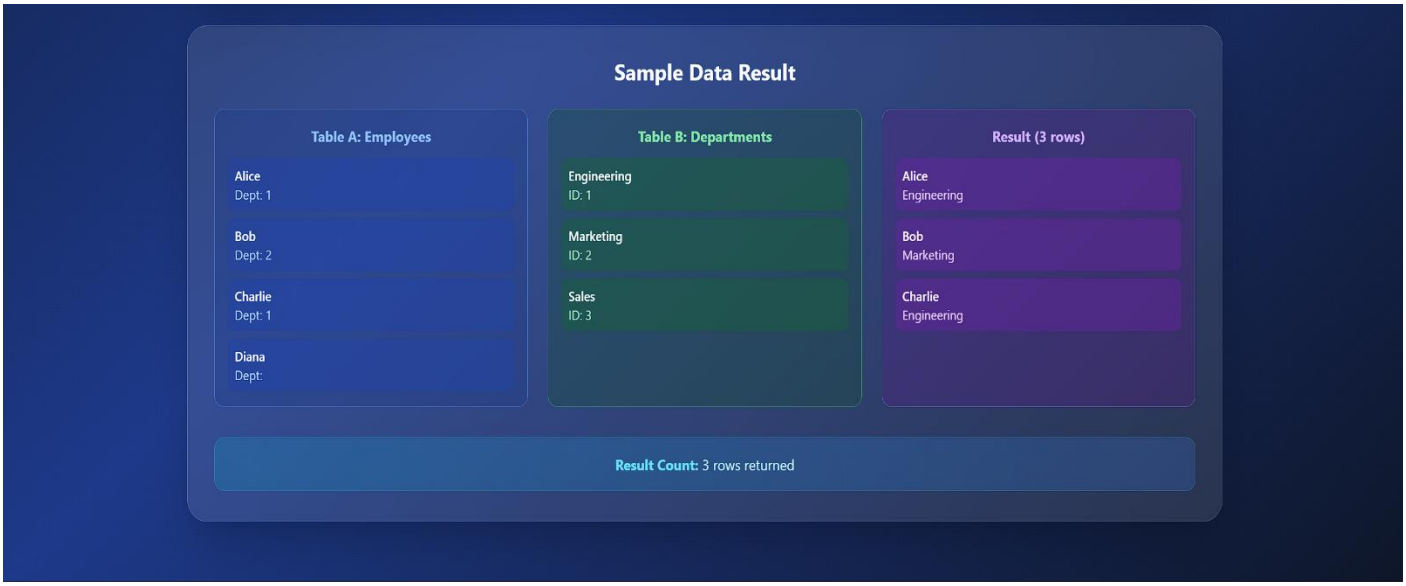
```
          ) : (
            <g>
              <circle
                cx="150"
                cy="150"
                r="80"
                fill={highlighted.leftOnly ? 'url(#leftGradient)' : 'rgba(59, 130, 246, 0.2)'}
                stroke="#3b82f6"
                strokeWidth="3"
                className={`transition-all duration-700 ${animate ? 'scale-100' : 'scale-0'}
${highlighted.leftOnly ? 'animate-pulse' : ''}`}
                style={{ transformOrigin: '150px 150px' }}
              />
              <circle
                cx="250"
                cy="150"
                r="80"
                fill={highlighted.rightOnly ? 'url(#rightGradient)' : 'rgba(16, 185, 129,
0.2)'}
                stroke="#10b981"
                strokeWidth="3"
                className={`transition-all duration-700 ${animate ? 'scale-100' : 'scale-0'}
${highlighted.rightOnly ? 'animate-pulse' : ''}`}
                style={{ transformOrigin: '250px 150px', animationDelay: '0.1s' }}
              />
              <ellipse
                cx="200"
                cy="150"
                rx="30"
                ry="80"
                fill={highlighted.intersection ? 'url(#intersectionGradient)' : 'rgba(168, 85,
247, 0.2)'}
```

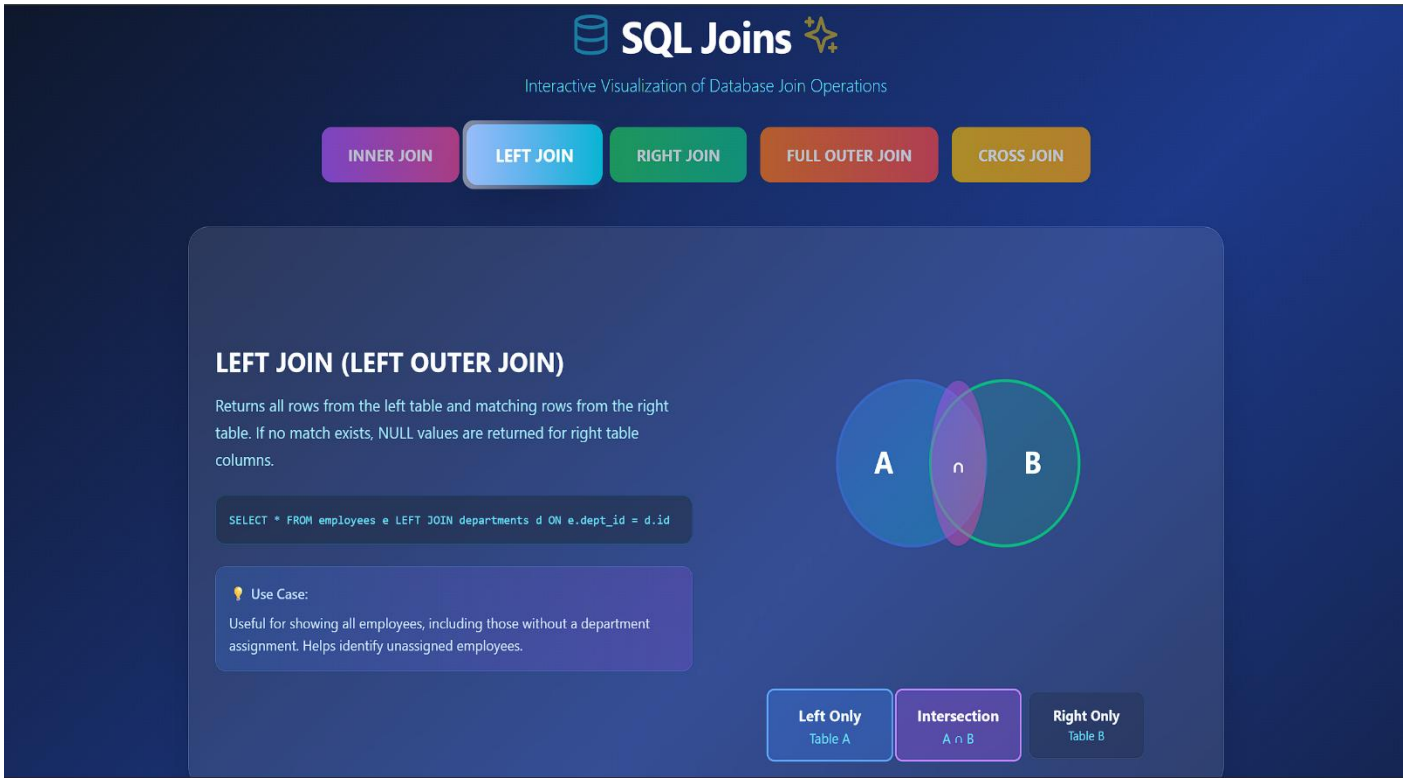
```
                  className={`transition-all duration-700 ${animate ? 'scale-100' : 'scale-0'}
${highlighted.intersection ? 'animate-pulse' : ''}`}
                  style={{ transformOrigin: '200px 150px', animationDelay: '0.2s' }}
                />
                <text
                  x="120"
                  y="160"
                  textAnchor="middle"
                  className="fill-white font-bold text-3xl pointer-events-none"
                >
                  A
                </text>
                <text
                  x="280"
                  y="160"
                  textAnchor="middle"
                  className="fill-white font-bold text-3xl pointer-events-none"
                >
                  B
                </text>
                <text
                  x="200"
                  y="160"
                  textAnchor="middle"
                  className="fill-white font-bold text-xl pointer-events-none"
                >
                  ∩
                </text>
              </g>
            )}
          </svg>
```

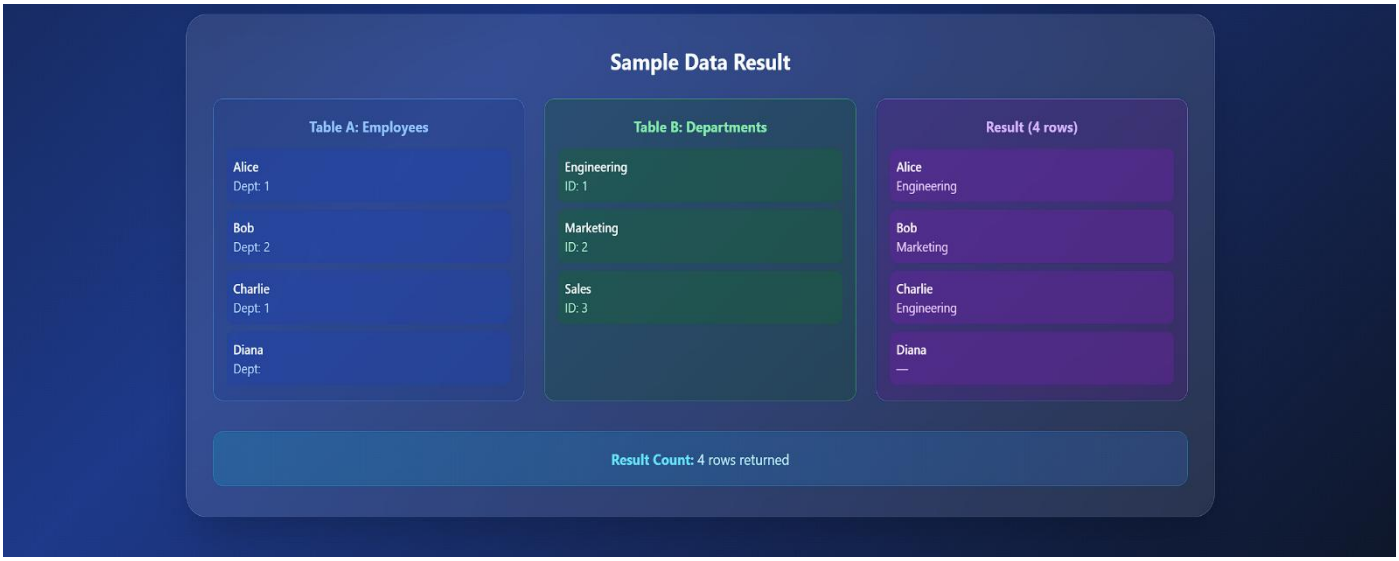
```
            <div className="mt-6 grid grid-cols-3 gap-4 text-center">
              <div className={`p-3 rounded-lg transition-all duration-500 ${highlighted.leftOnly ? 'bg-blue-500/30 border-2 border-blue-400 scale-110' : 'bg-slate-800/30 border border-slate-600'}`}>
                <div className="text-white font-bold">Left Only</div>
                <div className="text-sm text-cyan-300">Table A</div>
              </div>
              <div className={`p-3 rounded-lg transition-all duration-500 ${highlighted.intersection ? 'bg-purple-500/30 border-2 border-purple-400 scale-110' : 'bg-slate-800/30 border border-slate-600'}`}>
                <div className="text-white font-bold">Intersection</div>
                <div className="text-sm text-cyan-300">A ∩ B</div>
              </div>
              <div className={`p-3 rounded-lg transition-all duration-500 ${highlighted.rightOnly ? 'bg-green-500/30 border-2 border-green-400 scale-110' : 'bg-slate-800/30 border border-slate-600'}`}>
                <div className="text-white font-bold">Right Only</div>
                <div className="text-sm text-cyan-300">Table B</div>
              </div>
            </div>
          </div>
        );
      }
```

Output of Project

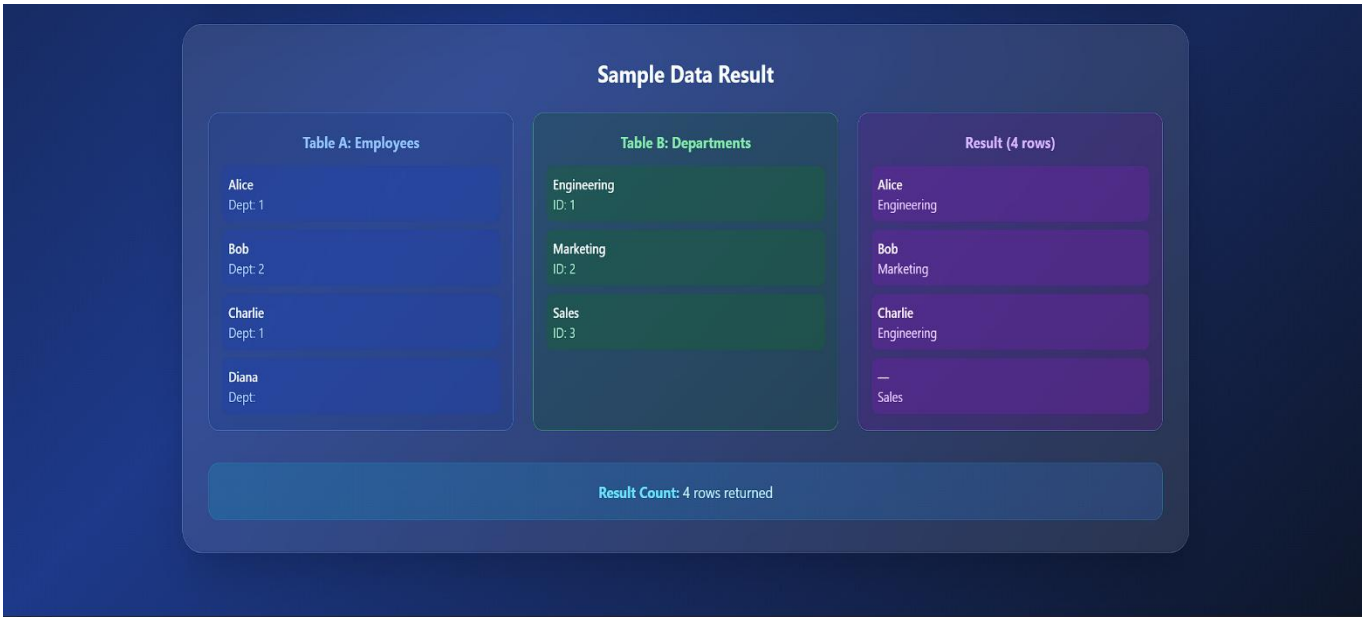


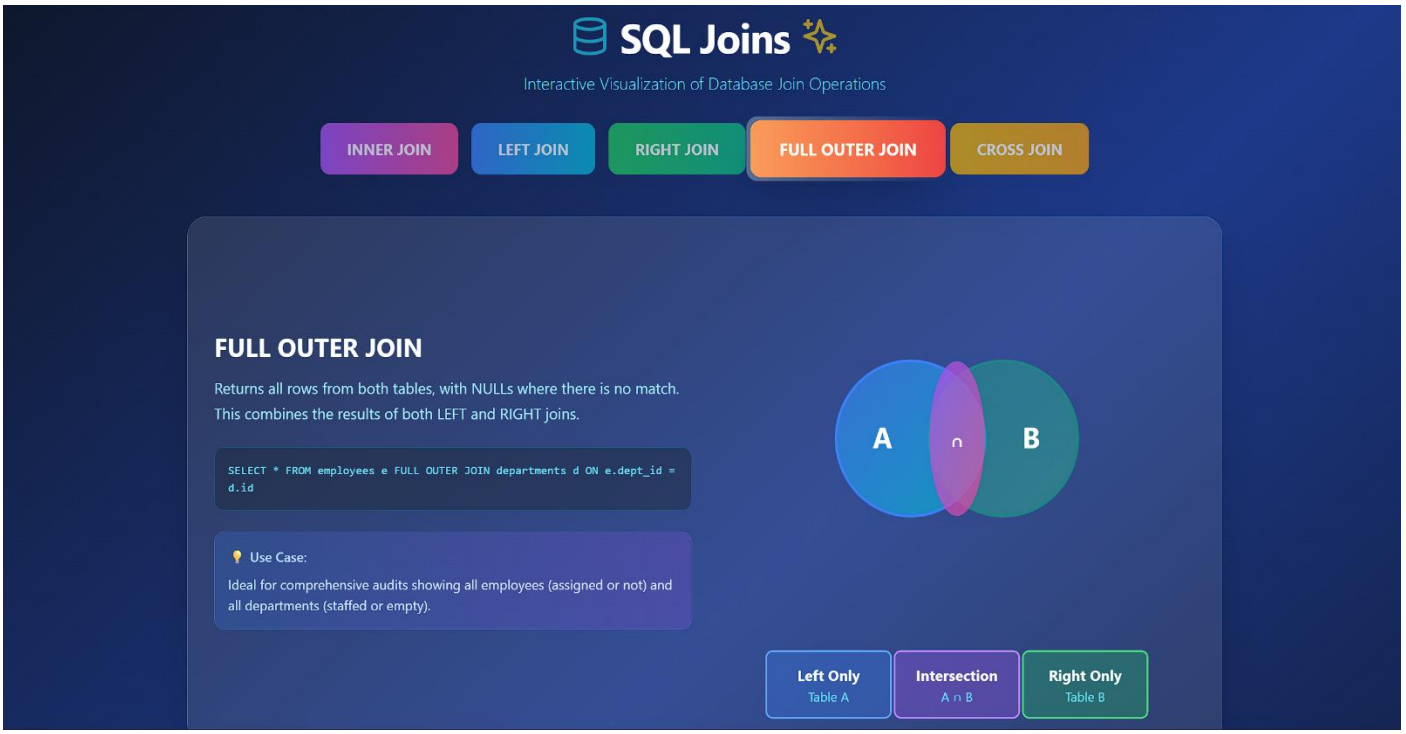


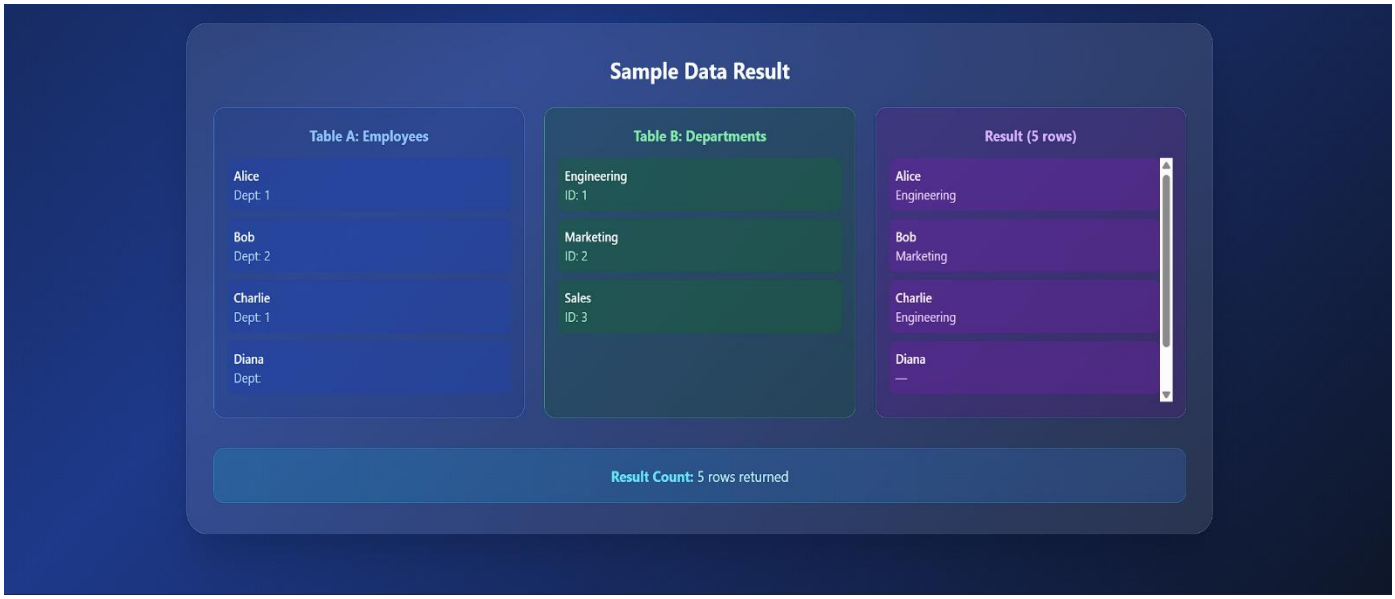

SQL Joins
Interactive Visualization of Database Join Operations
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN
CROSS JOIN
LEFT JOIN (LEFT OUTER JOIN)
Returns all rows from the left table and matching rows from the right table. If no match exists, NULL values are returned for right table columns.
SELECT * FROM employees e LEFT JOIN departments d ON e.dept_id = d.id
Use Case:
Useful for showing all employees, including those without a department assignment. Helps identify unassigned employees.
A
∩
B
Left Only
Table A
Intersection
A ∩ B
Right Only
Table B

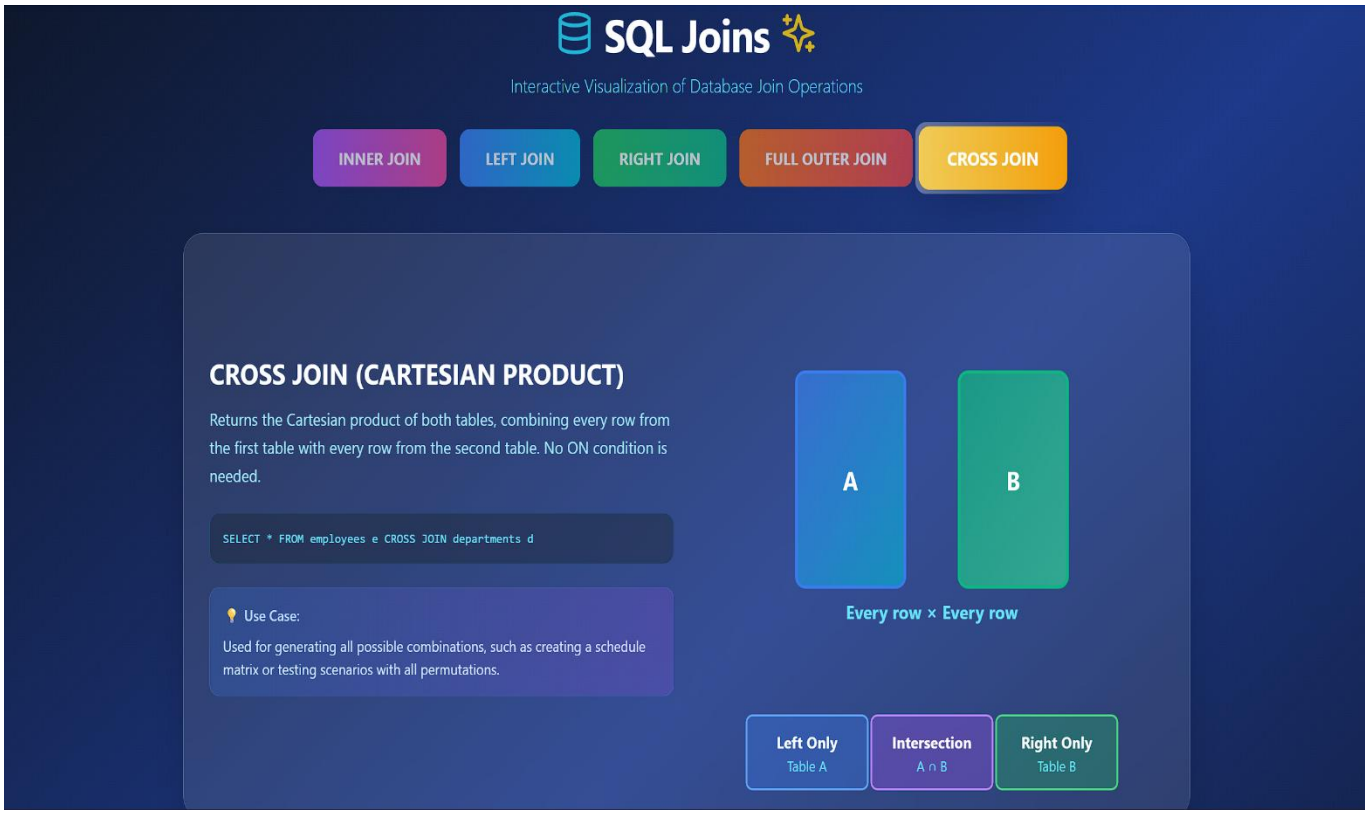

Sample Data Result
Table A: Employees
Alice
Dept: 1
Bob
Dept: 2
Charlie
Dept: 1
Diana
Dept:
Table B: Departments
Engineering
ID: 1
Marketing
ID: 2
Sales
ID: 3
Result (4 rows)
Alice
Engineering
Bob
Marketing
Charlie
Engineering
Diana
—
Result Count: 4 rows returned

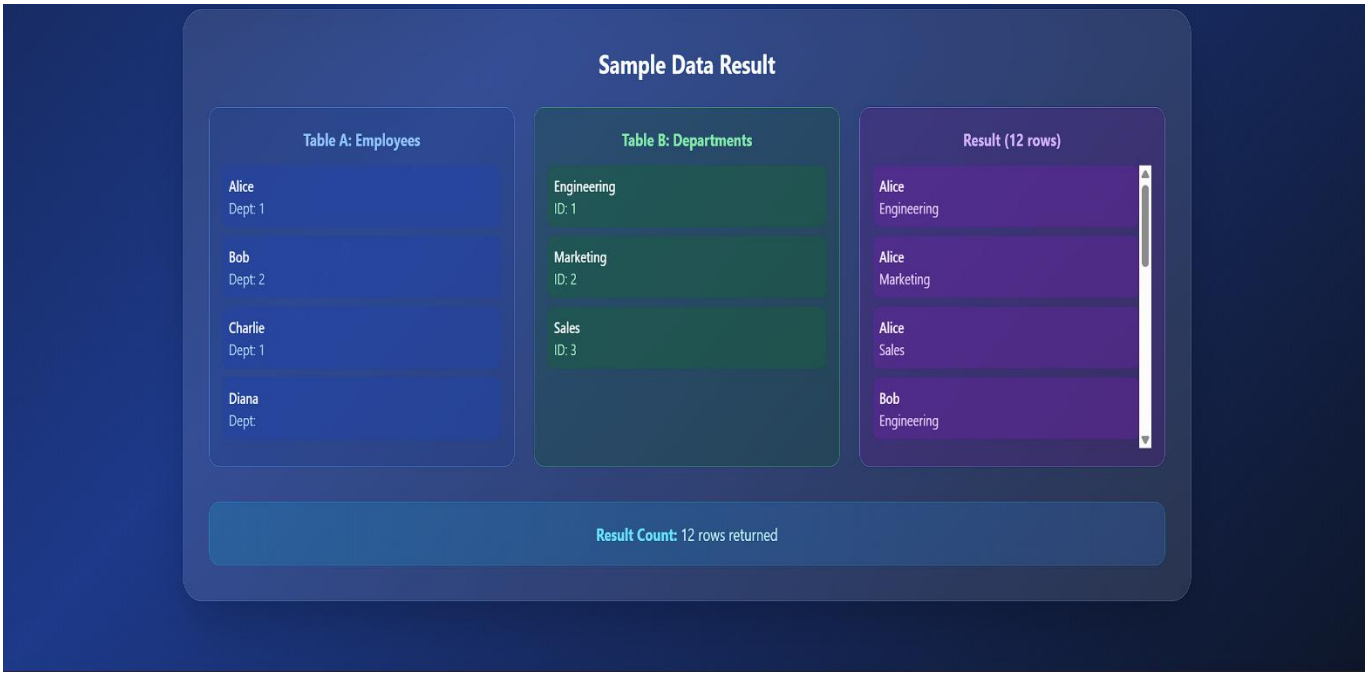

SQL Joins
Interactive Visualization of Database Join Operations
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN
CROSS JOIN
RIGHT JOIN (RIGHT OUTER JOIN)
Returns all rows from the right table and matching rows from the left table. If no match exists, NULL values are returned for left table columns.
SELECT * FROM employees e RIGHT JOIN departments d ON e.dept_id = d.id
Use Case:
Great for finding all departments, including empty ones with no employees. Helps identify underutilized departments.
A
∩
B
Left Only
Table A
Intersection
A ∩ B
Right Only
Table B


Sample Data Result
Table A: Employees
Alice
Dept: 1
Bob
Dept: 2
Charlie
Dept: 1
Diana
Dept:
Table B: Departments
Engineering
ID: 1
Marketing
ID: 2
Sales
ID: 3
Result (4 rows)
Alice
Engineering
Bob
Marketing
Charlie
Engineering
—
Sales
Result Count: 4 rows returned


SQL Joins
Interactive Visualization of Database Join Operations
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN
CROSS JOIN
FULL OUTER JOIN
Returns all rows from both tables, with NULLs where there is no match.
This combines the results of both LEFT and RIGHT joins.
SELECT * FROM employees e FULL OUTER JOIN departments d ON e.dept_id = d.id
Use Case:
Ideal for comprehensive audits showing all employees (assigned or not) and all departments (staffed or empty).
A
∩
B
Left Only
Table A
Intersection
A ∩ B
Right Only
Table B


Sample Data Result
Table A: Employees
Alice
Dept: 1
Bob
Dept: 2
Charlie
Dept: 1
Diana
Dept:
Table B: Departments
Engineering
ID: 1
Marketing
ID: 2
Sales
ID: 3
Result (5 rows)
Alice
Engineering
Bob
Marketing
Charlie
Engineering
Diana
—
Result Count: 5 rows returned


SQL Joins
Interactive Visualization of Database Join Operations
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN
CROSS JOIN
CROSS JOIN (CARTESIAN PRODUCT)
Returns the Cartesian product of both tables, combining every row from the first table with every row from the second table. No ON condition is needed.
SELECT * FROM employees e CROSS JOIN departments d
Use Case:
Used for generating all possible combinations, such as creating a schedule matrix or testing scenarios with all permutations.
A
B
Every row × Every row
Left Only
Table A
Intersection
A ∩ B
Right Only
Table B


Sample Data Result
Table A: Employees
Alice
Dept: 1
Bob
Dept: 2
Charlie
Dept: 1
Diana
Dept:
Table B: Departments
Engineering
ID: 1
Marketing
ID: 2
Sales
ID: 3
Result (12 rows)
Alice
Engineering
Alice
Marketing
Alice
Sales
Bob
Engineering
Result Count: 12 rows returned

Conclusion

The project **"Interactive Visualization of Database Join Operations"** successfully demonstrates how visual learning can simplify complex database concepts. By transforming SQL join operations into interactive graphical representations, it helps users clearly understand how data from multiple tables combine using different join types such as Inner, Left, Right, Full Outer, and Cross Join. This visual approach makes it easier to grasp how matching and non-matching records are handled in each case, reducing the confusion often faced when studying SQL through traditional text-based methods.

Through this project, learners gain both theoretical understanding and practical insight into relational database management. The tool enhances engagement, promotes self-learning, and bridges the gap between query writing and data interpretation. Overall, it serves as an effective educational resource for students, educators, and professionals, contributing to better comprehension of SQL joins and their real-world applications in data analytics and database design.