21CSC205P Database Management Systems

Project Report

**Register No. : RA2211026050004**

**Name of the Student : Arpan Boas Surin**

**Semester : 4**

**Programme : B.Tech. CSE(AI&ML)-A**

**Academic Year : 2023-2024**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

***TIRUCHIRAPPALLI***

# COLLEGE OF ENGINEERING AND TECHNOLOGY

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

***TIRUCHIRAPPALLI***

# COLLEGE OF ENGINEERING AND TECHNOLOGY



**BONAFIDE CERTIFICATE**

**Certified as the bonafide record of work done by**

**Arpan Boas Surin , Register No.**

**RA2211026050004 of 4/2 (Semester/Year),**

**B.Tech. CSE(AI&ML)-A Programme in the**

**project-based course 21CSC205P, Database Management Systems at SRM Institute of Science and Technology, Tiruchirappalli Campus during the Academic Year 2023-2024.**

**Faculty In-charge Head of the Department**

**Submitted for the End Semester Examination held on 03/05/2024**

**Examiner-1 Examiner-2**

**SRM Institute of Science and Technology Faculty of Engineering and Technology School of Computing**

**Department of Computational Intelligence**

## VISION AND MISSION OF THE DEPARTMENT

**Vision**

**To build a world-renowned academic platform in Computational Intelligence by providing unique learning and research experiences in collaboration with industries and world-class universities.**

## Mission

1. **To envision creating, acquiring, and disseminating engineering knowledge on computational intelligence to elevate a student into a professional by imparting knowledge on mathematics, computing sciences, artificial intelligence, and software engineering along with the skills of cognitive computing.**
2. **To offer a unique learning environment through world class faculty, curriculum, modernized lab facilities, and an interactive classroom environment with real-time experience from industrial experts that leads to a computing career in the latest technologies.**
3. **To uplift the innovative research and development in computational intelligence and its allied fields by collaborating with renowned academic institutions and industries.**
4. **To produce graduates who are global innovators and leaders in the development of computational intelligence-based systems, along with the commitment to ethical responsibilities and lifelong learning.**

## Program Educational Objectives (PEO)

**PEO - 1 Graduates will be able to perform in technical/managerial roles ranging from design, development, problem solving to production support in software industries and R&D sectors**

**PEO - 2 Graduates will be able to successfully pursue higher education in reputed institutions.**

**PEO - 3 Graduates will have the ability to adapt, contribute and innovate new technologies and systems in the key domains of Computer Science and Engineering.**

**PEO - 4 Graduates will be ethically and socially responsible solution providers and entrepreneurs in Computer Science and other engineering disciplines.**

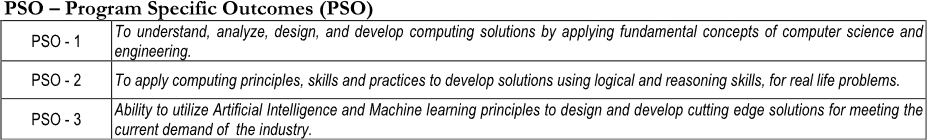
**PEO - 5 Graduates will possess the additional skills in core computer science discipline with knowledge of Hardware, Software, Programming, Logic & Reasoning.**

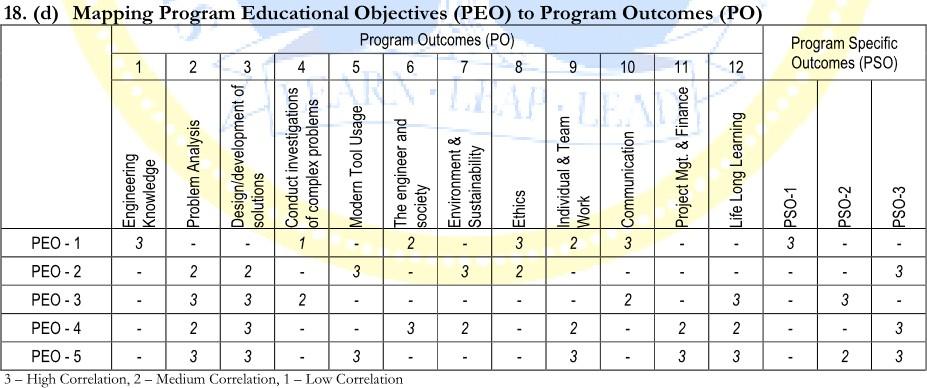
## Mission of the Department to Program Educational Objectives (PEO) Mapping

**Program Outcomes as defined by NBA (PO) Engineering Graduates will be able to:**

1. **Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.**
2. **Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.**
3. **Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.**
4. **Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.**
5. **Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.**
6. **The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.**
7. **Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.**
8. **Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.**
9. **Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.**
10. **Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.**
11. **Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.**
12. **Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.**

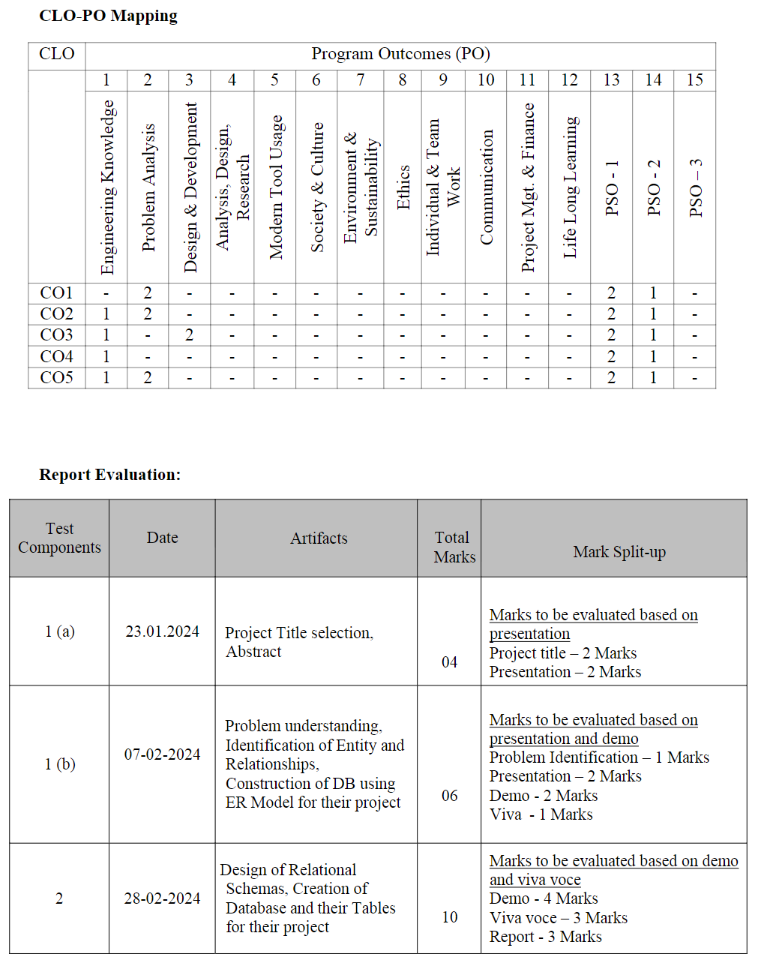
## PSO – Program Specific Outcomes (PSO)



**Mapping Program Educational Objectives (PEO) to Program Learning Outcomes (PLO)**

## COURSE DESIGN

| **Course Learning**  **Outcomes (CLO):** | At the end of this course, learners will be able to: |
| --- | --- |
| **CO – 1** | Acquire knowledge on DBMS architecture and languages |
| **CO – 2** | Acquire knowledge on Relational languages and design a database |
| **CO – 3** | Implement the Database structure with SQL |
| **CO – 4** | Removal of anomalies using Normalization concepts |
| **CO – 5** | Visualizing storage structure, handling concurrency, Failure and recovery principles, NoSQL concept |

****

|  |  | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

| 3 | 27-03-2024 | Writing the complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors | 10 | Marks to be evaluated based on demo and viva voce  Demo - 4 Marks Viva - 3 Marks Report -3 Marks |
| --- | --- | --- | --- | --- |
| 4 | 27-03-2024 | Analyzing the pitfalls, identifying the dependencies and applying normalizations | 10 | Marks to be evaluated based on demo and viva voce  Demo - 4 Marks Viva - 3 Marks Report - 3 Marks |
| 5 | 27-03-2024 | Implementation of concurrency control and recovery mechanisms. | 10 | Marks to be evaluated based on demo and viva voce  Demo - 4 Marks Viva - 3 Marks Report - 3 Marks |
| 6 | 15-04-2024 | Module Description and Functionalities | 10 | Marks to be evaluated based on presentation and demo  Final Presentation - 2 Demo - 2 Marks Viva - 3 Marks Report - 3 Marks |
| 7 | 15-04-2024 | Database Connectivity and Embedded SQL operations | 10 | Marks to be evaluated based on presentation and demo  Final Presentation - 2 Demo - 2 Marks Viva - 3 Marks  Report - 3 Marks |
| 8 | 29-04-2024 | PL/SQL procedures to Operate on the database of your project | 10 | Marks to be evaluated based on presentation and demo  Final Presentation - 2 Demo - 2 Marks  Viva - 3 Marks Report - 3 Marks |
| 9 | 29-04-2024 | Front End Design | 10 | Marks to be evaluated based on presentation and demo  Final Presentation - 2 Demo - 2 Marks Viva - 3 Marks  Report - 3 Marks |
| 10 | 07-05-2024 | Report Generation using Crystal Reports/Report Generation Tool | 10 | Marks to be evaluated based on presentation and demo  Final Presentation - 2 Demo - 2 Marks Viva - 3 Marks  Report - 3 Marks |

**LABORATORY EXPERIMENTS MAPPED WITH CLO, PO, PSO**

| **Sl.**  **No.** | **Title of the Experiments** | **CLO**  **Mapping** | **PO / PSO**  **Mapping** |
| --- | --- | --- | --- |
| 1 | 1. Project Title selection, Abstract 2. Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for your project | 1 | 2 |
| 2 | Design of Relational Schemas, Creation of Database and Tables for your project | 1,2 | 1,2 |
| 3 | SQL Queries based on the concepts of constraints, sets, joins | 3 | 1,3 |
| 4 | Applying Normalization for your project | 4 | 1 |
| 5 | Concurrency control and Recovery mechanisms for your project | 5 | 1,2 |
| 6 | Module Description and Functionalities | 1,2 | 1,2 |
| 7 | Database Connectivity and Embedded SQL operations | 1,2 | 1,2 |
| 8 | PL/SQL procedures to Operate on the database of your project | 2,3 | 1,2,3 |
| 9 | Front End Design, Sample Code for each module, Screen Shots for each module | 5 | 1,2 |
| 10 | Report Generation using Crystal Reports/Report Generation Tool | 5 | 1,2 |

**INDEX**

| **Expt. No.** | **Date of Experiment** | **Name of the Experiment** | **Page No.** | **Marks (50)** | **Signature of Faculty** |
| --- | --- | --- | --- | --- | --- |
| 1 | 07-02-2024 | 1. Project Title selection, Abstract 2. Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for   your project | 01-05 |  |  |
| 2 | 28-02-2024 | Design of Relational Schemas, Creation of Database and Tables for your project | 06-11 |  |  |
| 3 | 27-03-2024 | SQL Queries based on the concepts of constraints, sets, joins | 12-18 |  |  |
| 4 | 27-03-2024 | Applying Normalization for your project | 19-22 |  |  |
| 5 | 27-03-2024 | Concurrency control and Recovery mechanisms for your project | 23-30 |  |  |
| 6 | 15-04-2024 | Module Description and Functionalities | 31-34 |  |  |
| 7 | 15-04-2024 | Database Connectivity and Embedded SQL operations | 35-41 |  |  |
| 8 | 29-04-2024 | PL/SQL procedures to Operate on the database of your project | 42-48 |  |  |
| 9 | 29-04-2024 | Front End Design, Sample Code for each module, Screen Shots for each module | 49-66 |  |  |
| 10 | 07-05-2024 | Report Generation using Crystal Reports/Report Generation Tool | 67 |  |  |
| 11 | 07-05-2024 | Online Course Completion Certificate | 68 |  |  |

**Online Shoes Management System**

The Shoe Management System (SMS) is a comprehensive software solution designed to streamline and automate various aspects of managing a shoe store or retail chain. The system integrates inventory management, sales tracking, customer relationship management (CRM), and reporting functionalities to provide efficient operations and enhance customer satisfaction.

**Problem Understanding :**

**Purpose :**

The Shoe Management System is a software solution created to help manage shoe stores or chains more effectively. It's like having a digital assistant that handles various tasks related to inventory, sales, customer information, and business analysis.

**Features :**

Inventory Management: This feature lets you keep track of all the shoes you have in stock. It records details like product names, sizes, colors, and where they came from (suppliers). Plus, it helps you manage stock movements, like when you get new shoes, sell them, or have returns.

**Sales and POS Integration :**

Imagine having a super-efficient cash register. That's what this feature is like. It helps store staff process sales transactions quickly and accurately. You can pay with cash, cards, or mobile payments, and the system updates the inventory automatically so you always know what's available.

**Customer Relationship Management (CRM) :**

This part of the system helps you keep track of your customers. You can record things like their contact details, what they like to buy, and how often they shop. This information helps you tailor marketing efforts and run loyalty programs.

**Reporting and Analytics :**

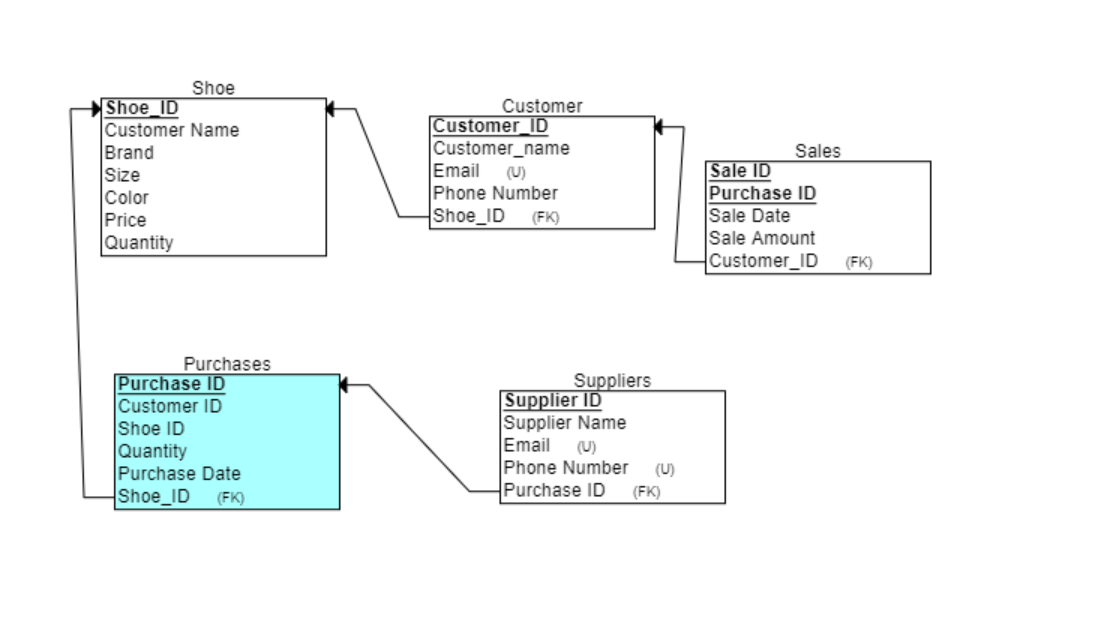
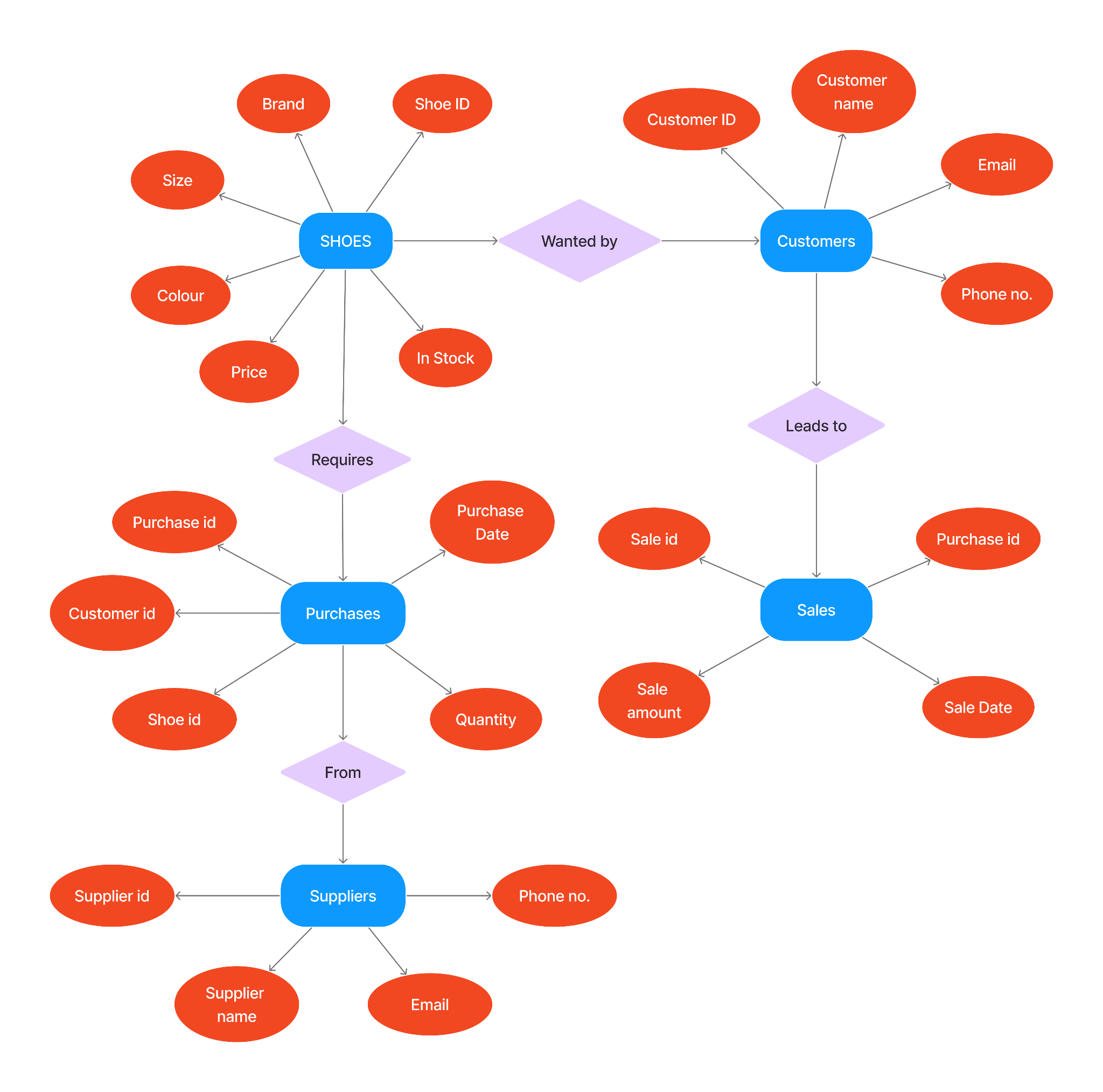
Think of this as your business's dashboard. It gives you reports and graphs that show how well your store is doing. You can see things like how much you're selling, how quickly your shoes are flying off the shelves, and which marketing strategies are working best.

**Multi-Store Management :**

If you have more than one store, this feature helps you manage them all from one place. You can control inventory, check sales, and view reports for each store without having to jump between different systems.

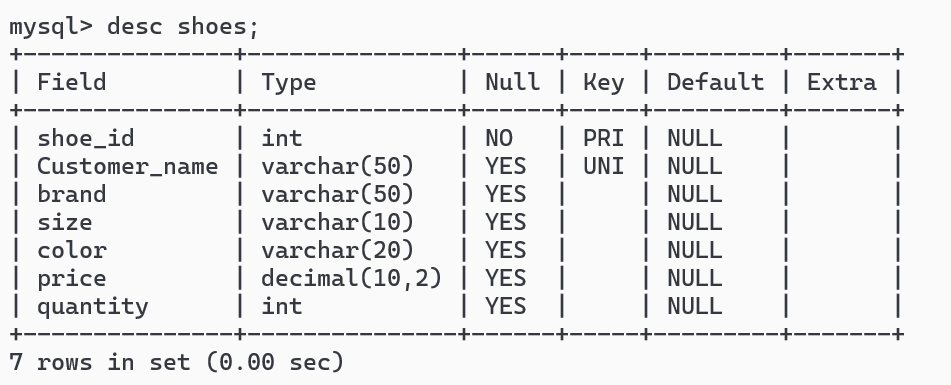
**Identification of Entity & Relationships,**

**Conversion of DB using ER model**

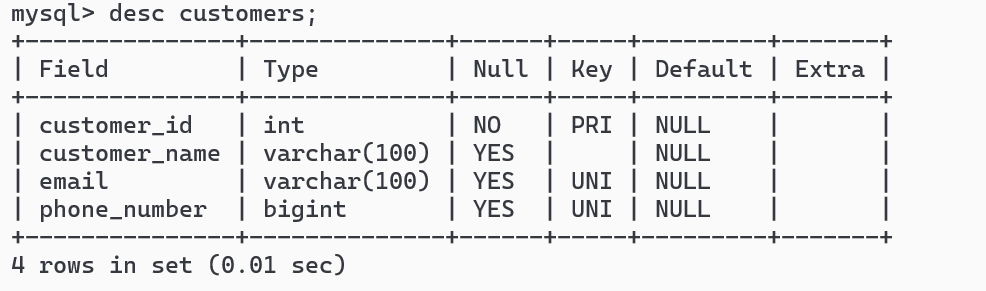
****

**Design of Relational Schemas**

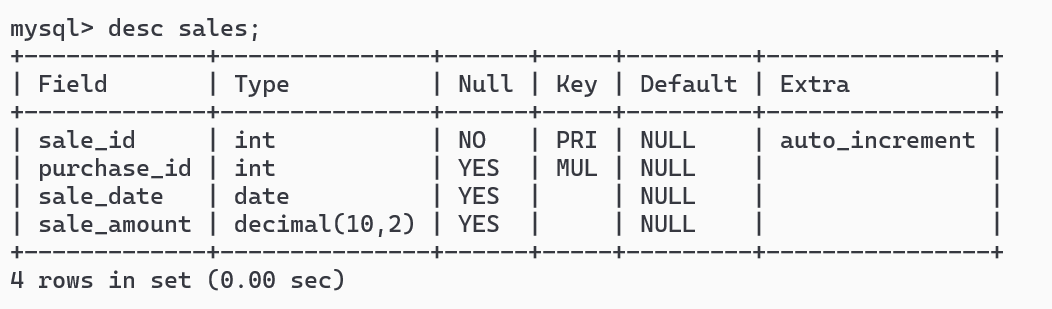
**Shoes Table :**

****

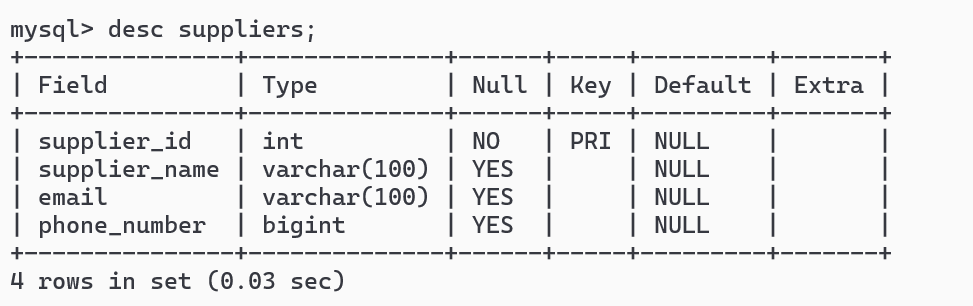
**Customers Table :**

****

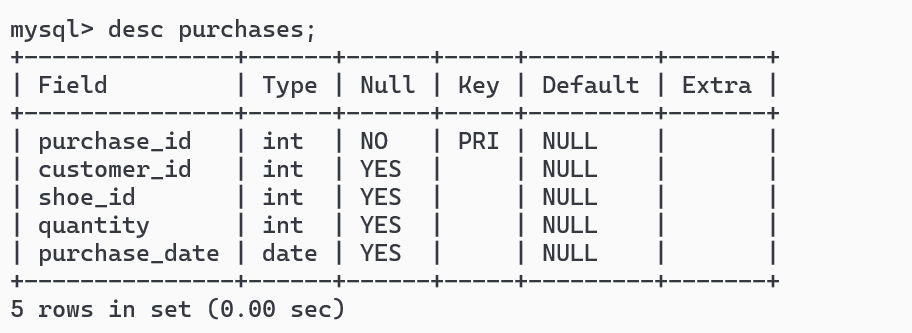
**Sales Table :**

****

**Suppliers Table :**

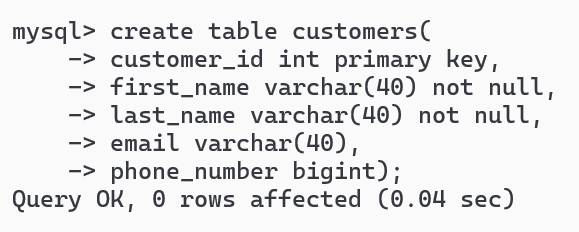
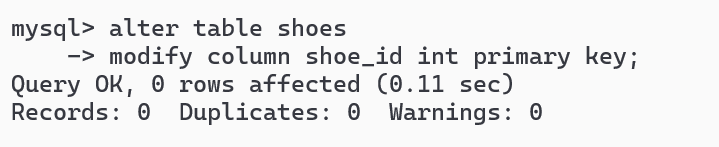
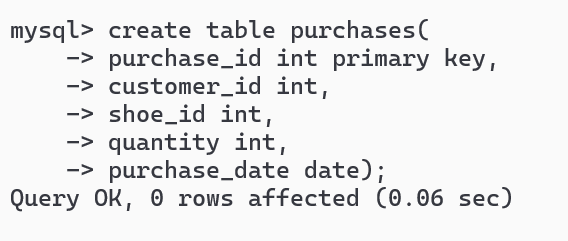
****

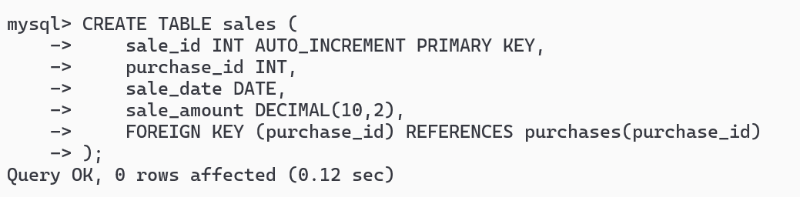
**Purchases Table :**

****

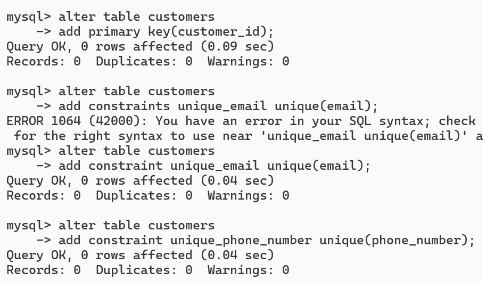
**SQL Queries based on the concepts of constraints, sets, joins**

**Primary key Constraint, :**

****

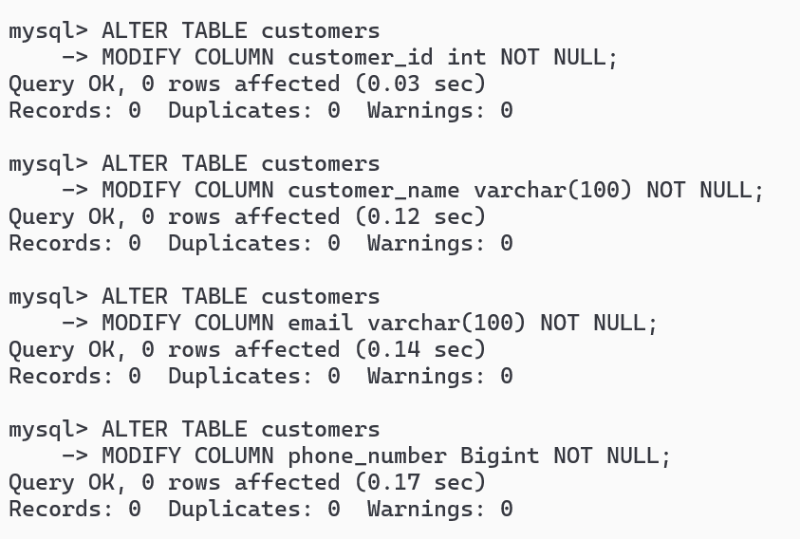
****

**Unique key Constraint :**

****

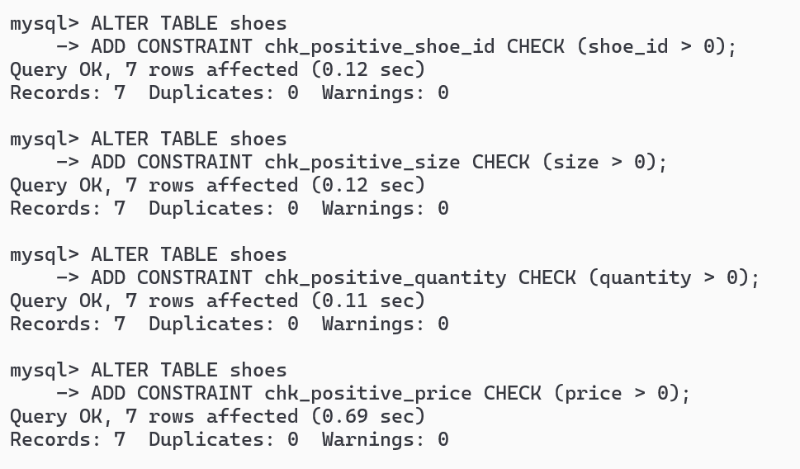
****

**Not Null Constraint :**

****

****

**Check Constraint :**

****

**Normalization Application in the Project**

Normalization in database management refers to the process of organizing the attributes and tables of a relational database to minimize redundancy and dependency. The primary goals of normalization are to reduce data redundancy, eliminate update anomalies, and ensure data integrity.

**First Normal Form (1NF):**

* In 1NF, each attribute (or column) in a table must contain atomic (indivisible) values. This means that each column should hold only a single value, not a set of values or a list.
* 1NF eliminates repeating groups within a table. For example, if you have a table for student records and one student can have multiple phone numbers, instead of having multiple phone number columns, each student should have only one phone number in a single column. If there are multiple phone numbers for a student, you should create a separate table to hold phone numbers and link it back to the student table through a foreign key.

**Second Normal Form (2NF):**

* To be in 2NF, a table must first satisfy the conditions of 1NF.
* Additionally, all non-key attributes (attributes that are not part of the primary key) must be fully functionally dependent on the entire primary key, not on a part of it.
* This means that if a table has a composite primary key (a primary key composed of multiple columns), every non-key attribute should be dependent on the combination of these columns, not just on one of them. If a non-key attribute depends on only part of the primary key, it indicates a partial dependency and violates 2NF.
* To resolve partial dependencies, you may need to split the table into multiple tables, each with its own primary key and the related attributes.

**Third Normal Form (3NF):**

* To achieve 3NF, a table must first satisfy the conditions of 2NF.
* Additionally, all non-key attributes must be directly dependent on the primary key, and there should be no transitive dependencies.
* Transitive dependency occurs when a non-key attribute depends on another non-key attribute, which is itself dependent on the primary key. This kind of dependency can lead to data redundancy and update anomalies.
* To eliminate transitive dependencies, you may need to further normalize the tables by breaking them down into smaller tables, each containing attributes that are directly related to the primary key.

**Boyce-Codd Normal Form (BCNF):**

* BCNF is a stricter form of normalization compared to 3NF.
* In BCNF, every determinant (attribute whose value determines another attribute) must be a candidate key. In other words, no non-prime attribute should be functionally dependent on any proper subset of any candidate key.
* BCNF aims to eliminate all anomalies related to non-trivial functional dependencies while preserving the properties of functional dependency and lossless decomposition.

**Fourth Normal Form (4NF) :**

* 4NF further addresses multivalued dependencies.
* It ensures that there are no non-trivial multivalued dependencies between candidate keys and non-prime attributes.
* 4NF eliminates redundancy and anomalies related to multivalued dependencies by decomposing tables into smaller, more atomic tables.

**Fifth Normal Form (5NF) :**

* The Fifth Normal Form (5NF), also known as Project-Join Normal Form (PJNF), is a level of normalization that deals with the elimination of join dependencies, which are dependencies that involve multiple tables.
* It is an advanced level of normalization beyond BCNF and is rarely encountered in practice, as it typically applies to very complex databases.
* 5NF is achieved when every join dependency in a database schema is a consequence of the candidate keys..
* 5NF ensures that these join dependencies are derived from the candidate keys and not from any other combination of attributes.

**Concurrency control and Recovery mechanisms**

**Concurrency Control Mechanisms :-**

* **Locking :**

Use locking mechanisms to control access to data during read and write operations. For example, implement record-level locking to allow multiple users to read data simultaneously but prevent concurrent writes to the same record.

* **Transactions :**

Implement database transactions to group related database operations into atomic units of work. Use techniques like ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure transactional integrity and isolate concurrent transactions.

* **Isolation Levels :**

Choose appropriate isolation levels (e.g., Read Uncommitted, Read Committed, Repeatable Read, Serializable) to control the visibility and consistency of data accessed by concurrent transactions.

* **Optimistic Concurrency Control :**

Use techniques such as optimistic locking, where conflicts are detected at the time of transaction commit rather than during data access. This can help reduce contention and improve concurrency in scenarios with low update contention.

**Recovery Mechanisms :-**

* **Backup and Restore :**

Regularly backup the database to ensure that data can be restored in the event of data loss or corruption. Implement automated backup processes to create consistent backups at scheduled intervals.

* **Transaction Logging :**

Maintain transaction logs to record changes made to the database. Transaction logs can be used for point-in-time recovery, rollback, and forward recovery in the event of system failures or crashes.

* **Checkpointing :**

Periodically perform checkpoints to flush dirty pages from memory to disk and update the database's metadata. Checkpointing helps minimize recovery time by reducing the amount of redo and undo operations required during system restarts.

* **Redundancy and Failover :**

Implement redundancy and failover mechanisms to ensure high availability and fault tolerance. Use techniques like database replication, clustering, and mirroring to maintain duplicate copies of data and automatically switch to standby servers in case of failures.

By implementing these concurrency control and recovery mechanisms, the online shoe management system can ensure data consistency, integrity, and reliability, even in the face of concurrent access and system failures. Additionally, thorough testing and monitoring are essential to validate the effectiveness of these mechanisms and identify any potential issues or bottlenecks.

**Module Description and their Functionalities**

**Front - End**

**React JS :**

ReactJS is an open-source JavaScript library developed by Facebook for building user interfaces, specifically for single-page applications. It's commonly used for creating interactive and dynamic UI components in web applications.

Key features and concepts of ReactJS include:

* **Component-Based Architecture :**

ReactJS uses a component-based architecture where UIs are composed of reusable components. Components encapsulate their own state and logic, making them modular and easier to manage.

* **Virtual DOM :**

ReactJS uses a virtual DOM (Document Object Model) for efficient updating of the UI. Instead of directly manipulating the browser's DOM, React creates a virtual representation of the DOM in memory and then efficiently updates the actual DOM when needed. This approach improves performance and optimizes rendering.

* **JSX (JavaScript XML ):**

JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within JavaScript. It enables the creation of UI components in a declarative and expressive manner.

* **Unidirectional Data Flow :**

React follows a unidirectional data flow, also known as one-way data binding. Data flows from parent components to child components through props, and child components can communicate with parent components through callbacks.

* **Hooks :**

Introduced in React version 16.8, hooks are functions that enable developers to use state and other React features in functional components. They provide a simpler and more concise way to work with stateful logic in functional components.

* **Component Lifecycle :**

React components have lifecycle methods that allow developers to hook into different stages of a component's lifecycle, such as component mounting, updating, and unmounting. These methods provide opportunities for performing tasks like fetching data, setting up subscriptions, and cleaning up resources.

Overall, ReactJS simplifies the process of building complex user interfaces by providing a modular, declarative, and efficient approach to UI development. It has a large and active community, extensive documentation, and a rich ecosystem of libraries and tools, making it a popular choice for web development**.**

**Back - End**

**Express JS :**

Express.js, commonly referred to as Express, is a minimal and flexible web application framework for Node.js, a runtime environment that allows you to run JavaScript code on the server-side. It provides a robust set of features for building web and mobile applications.

Express.js is designed to make it easier to develop server-side applications by providing a simple and intuitive API for handling HTTP requests, routing, middleware integration, and more. It's often used to build web servers and RESTful APIs, but it can also be used for various other types of server-side applications.

Key features of Express.js include:

* **Routing :**

Express provides a straightforward mechanism for defining routes based on HTTP methods (GET, POST, PUT, DELETE, etc.) and URL patterns. This allows you to create endpoints for handling different types of requests.

* **Middleware :**

Express middleware are functions that have access to the request and response objects, and can modify them or perform additional processing before passing them to the next middleware function in the chain. Middleware can be used for tasks such as parsing request bodies, authenticating users, logging requests, and more.

* **Template engines :**

Although not built-in to Express itself, it supports a wide range of template engines (such as EJS, Pug, Handlebars) that allow you to generate HTML dynamically based on data from your application.

* **Error handling :**

Express provides built-in error handling mechanisms, including error middleware and error-handling routes, to handle errors that occur during request processing.

* **Static file serving :**

Express can serve static files (such as HTML, CSS, and JavaScript files) from a directory on the server, making it easy to host static websites and serve client-side assets.

**Node JS :**

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to build server-side and networking applications. It is built on the V8 JavaScript engine, which is the same engine that powers Google Chrome. Node.js enables developers to use JavaScript for both client-side and server-side scripting, offering a unified development environment.

Here are some key aspects of Node.js:

* **JavaScript Runtime :**

Node.js provides an environment for executing JavaScript code outside of a web browser. It allows developers to run JavaScript on the server, enabling them to build full-stack applications entirely in JavaScript.

* **Event-Driven Architecture :**

Node.js operates on an event-driven, non-blocking I/O model, making it highly efficient and scalable. It uses asynchronous programming, allowing multiple operations to be performed concurrently without blocking the execution of other code.

* **Single-Threaded, Event Loop :**

Node.js applications run on a single thread, but they can handle many concurrent connections efficiently due to its event loop mechanism. The event loop enables Node.js to handle I/O operations asynchronously, making it well-suited for handling large numbers of concurrent requests.

* **NPM (Node Package Manager) :**

NPM is the default package manager for Node.js, providing a vast ecosystem of reusable libraries and modules. Developers can easily install, manage, and share packages using NPM, making it convenient for building complex applications with minimal effort.

* **Cross-Platform :**

Node.js is designed to be cross-platform, meaning it can run on various operating systems such as Windows, macOS, and Linux. This allows developers to write code once and deploy it across different platforms without modification.

* **Use Cases :**

Node.js is commonly used for building web servers, APIs (Application Programming Interfaces), real-time web applications (e.g., chat applications, gaming servers), microservices, command-line tools, and more.

**Database Management**

**MySQL :**

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL). It's one of the most popular database systems in the world and is commonly used for web applications, particularly those built with PHP.

Here are some key points about MySQL:

* **Open Source :**

MySQL is free and open-source software, which means you can use it, modify it, and distribute it under the terms of the GNU General Public License (GPL).

* **Relational Database :**

MySQL is a relational database, meaning it organizes data into tables, which can then be related to each other based on common fields.

* **Structured Query Language (SQL) :**

MySQL uses SQL as its query language. SQL is a standard language for managing and manipulating relational databases. With SQL, you can perform various operations like creating and modifying tables, inserting and updating data, and querying data from the database.

* **Client-Server Architecture :**

MySQL follows a client-server architecture. Clients (such as applications or tools) communicate with the MySQL server over a network connection to perform database operations.

* **Scalability :**

MySQL is designed to be scalable, allowing it to handle large volumes of data and high traffic loads. It supports various replication and clustering features for scalability and high availability.

* **Cross-Platform :**

MySQL runs on many different operating systems, including Linux, Windows, macOS, and various flavors of Unix.

* **Acquisition by Oracle :**

MySQL was originally developed by a Swedish company called MySQL AB. In 2008, Sun Microsystems acquired MySQL AB, and then Oracle Corporation acquired Sun Microsystems in 2010.

**Browser Router :**

In React.js, BrowserRouter is a component provided by the React Router library that enables declarative routing in a React application. It's used to synchronize the UI with the URL of the browser, allowing you to define different views or components to render based on different URLs. here's how BrowserRouter works :

* **Declarative Routing :**

With BrowserRouter, you can define a set of routes, each associated with a specific URL path and corresponding component to render.

* **History API :**

BrowserRouter uses the HTML5 History API under the hood to manipulate the browser's URL without causing a page reload. This allows for a smoother user experience when navigating between different views in a single-page application (SPA).

* **Route Matching :**

When the user navigates to a specific URL, BrowserRouter matches the URL against the defined routes and renders the corresponding component(s). It supports exact matching, partial matching, and parameterized matching for dynamic routing.

* **Nested Routing :**

BrowserRouter supports nested routing, allowing you to define routes within other routes. This enables you to create complex navigation structures with nested views and components.



**Link :**

In React.js, Link is a component provided by the React Router library for declarative navigation within a single-page application (SPA). It's used to create hyperlinks that allow users to navigate between different views or pages without causing a full page reload. here's how Link works:

* **Declarative Navigation :**

With Link, you can define clickable elements (such as text or buttons) that, when clicked, trigger navigation to a specified URL within the application.

* **Preventing Full Page Reloads :**

Unlike traditional HTML <a> tags, which cause a full page reload when clicked, Link uses client-side routing provided by React Router to handle navigation without reloading the entire page. This results in a smoother and faster user experience, similar to navigating within a traditional multi-page website.

* **Automatic URL Updating :**

When a Link is clicked, React Router updates the browser's URL using the History API without causing a full page reload. This ensures that the browser's back and forward buttons work as expected, allowing users to navigate through the application's history.

* **Support for Route Parameters :**

Link supports route parameters, allowing you to create dynamic links that include parameters in the URL. This is useful for creating links to specific resources or pages that depend on dynamic data.



**UseState() :**

In React, useState() is a Hook that allows functional components to manage state. It is one of the built-in Hooks provided by React, introduced in React 16.8. useState() enables you to add stateful logic to functional components without needing to convert them into class components. here's how useState() works :

* **Initializing State :**

You call useState() within a functional component to declare a piece of state and its initial value. The useState() function returns an array containing two elements: the current state value and a function to update the state.

* **Accessing State :**

You can access the current state value by destructuring the first element of the array returned by useState(). This value represents the current state of the component.

* **Updating State :**

To update the state, you call the function returned by useState() (the second element of the array) and pass in the new state value. React will re-render the component with the updated state.

**UseEffect() :**

In React, the useEffect() hook is a built-in function that allows you to perform side effects in function components. Side effects include things like fetching data, subscribing to events, or manually changing the DOM.

The useEffect() hook is called after every render of the component (including the initial render), and it runs the specified callback function. You can think of it as a way to perform tasks that need to happen after the component has rendered or when certain conditions have changed. here's how useEffect() works :

* **Effect function :**

This is the callback function that contains the side effect code you want to run. It can contain any code that needs to be executed after the component renders or when certain dependencies change.

* **Dependencies array :**

This is an optional array that specifies the values (props or state variables) that the effect depends on. If any of the values in the dependencies array change between renders, the effect function will be re-executed. If you omit the dependencies array, the effect function will run after every render.

* **Cleanup :**

If the effect callback returns a function, React will run this cleanup function before running the effect again or when the component unmounts. This cleanup function can be used to perform any necessary cleanup or teardown tasks, such as unsubscribing from a subscription or removing event listeners.





**Axios :**

Axios is a popular JavaScript library used for making HTTP requests from web browsers and Node.js environments. In React applications, Axios is commonly used to fetch data from APIs and perform CRUD (Create, Read, Update, Delete) operations with a server.

Here's an overview of Axios and its features:

* **HTTP Requests :**

Axios provides a simple and intuitive API for making HTTP requests, including GET, POST, PUT, DELETE, and more. It supports various request methods and provides flexibility for customizing request headers, parameters, and data payloads.

* **Promise-based :**

Axios is built on top of JavaScript promises, allowing you to write asynchronous code using async/await or .then() syntax. This makes it easy to handle asynchronous operations and manage response data in a clean and concise manner.

* **Interceptors :**

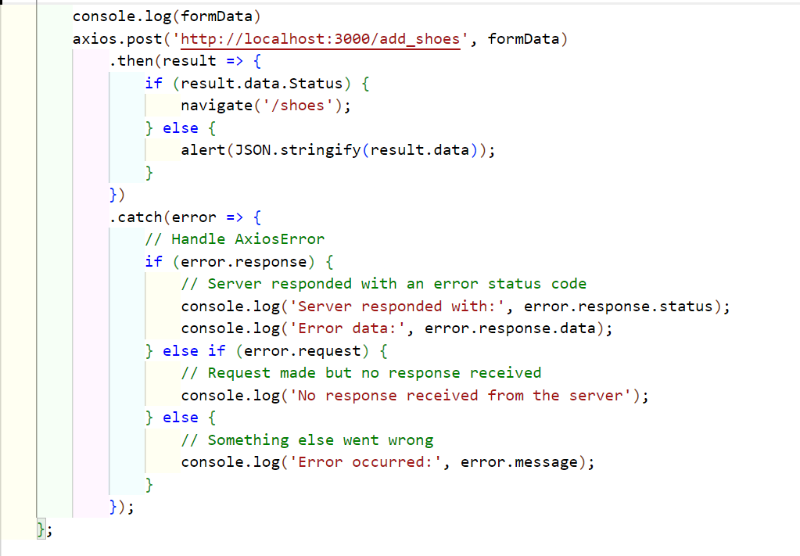
Axios allows you to intercept and modify HTTP requests and responses globally or on a per-request basis. Interceptors can be used for tasks such as adding authentication tokens, logging request details, or handling errors.

* **Request Cancelation :**

Axios supports canceling pending HTTP requests using cancel tokens. This feature is useful for scenarios like canceling in-flight requests when a user navigates away from a page or cancels an operation.

* **Browser Compatibility :**

Axios is compatible with all modern web browsers and Node.js environments. It includes built-in support for handling cross-origin resource sharing (CORS) and automatically converts response data to JSON format.



**UseNavigate() :**

useNavigate() is a hook provided by React Router v6, which enables programmatic navigation in React components. It's an alternative to useHistory() from React Router v5.

Here's an explanation of useNavigate():

* **Import the Hook :**

First, you need to import the useNavigate() hook from the react-router-dom package.

* **Access the Navigate Function:**

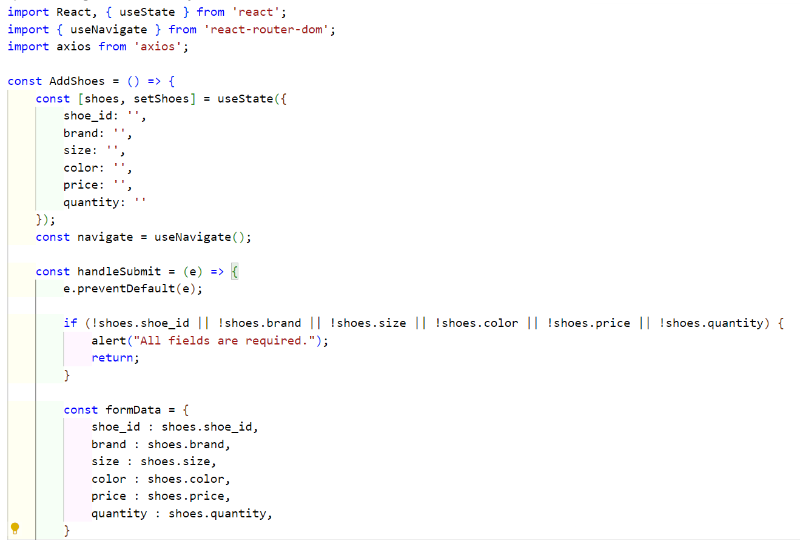
Call the useNavigate() hook within your component to access the navigate function, which is used to navigate to different routes.

* **Use the Navigate Function :**

You can use the navigate function to programmatically navigate to a specific route by passing the route path as an argument.

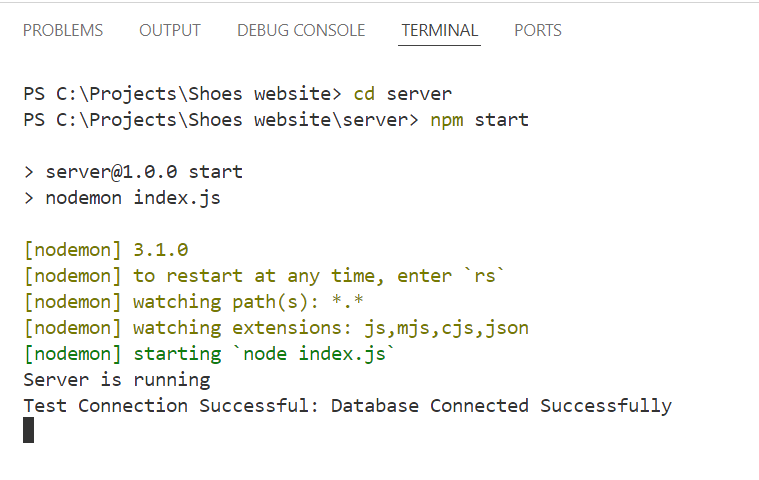
* **Handle Navigation Programmatically :**

You can call the navigate function within event handlers, lifecycle methods, or any other logic to trigger navigation based on certain conditions or user interactions.

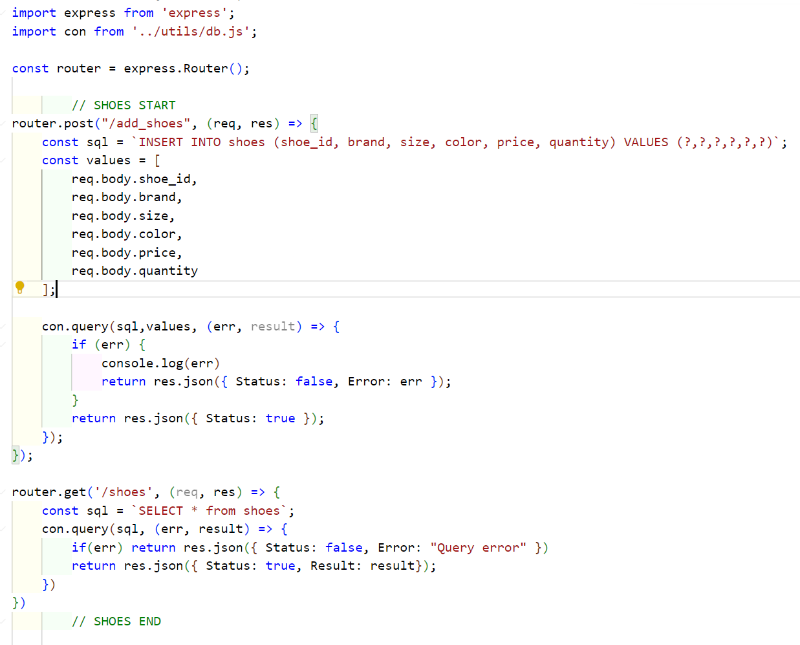


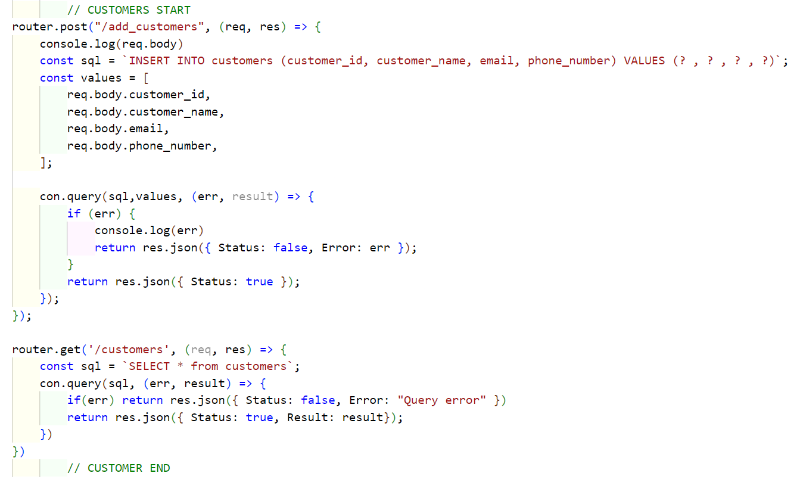
**Database Connectivity**

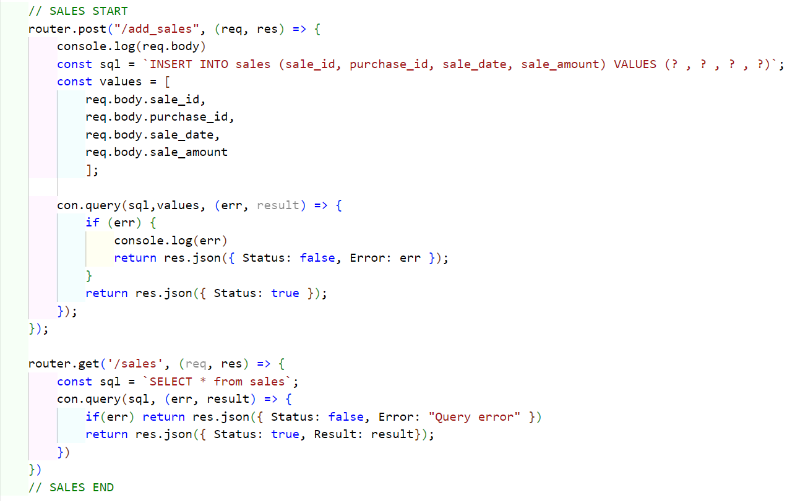


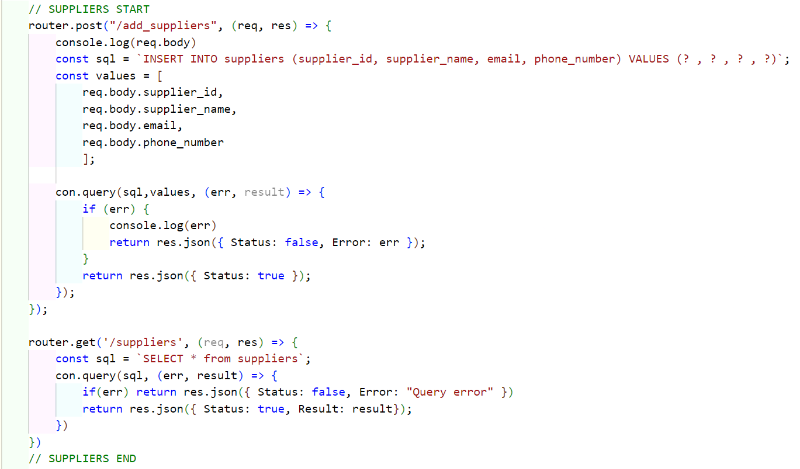


**Embedded SQL operations**



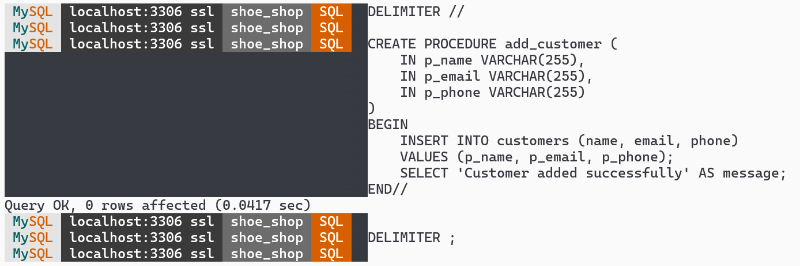


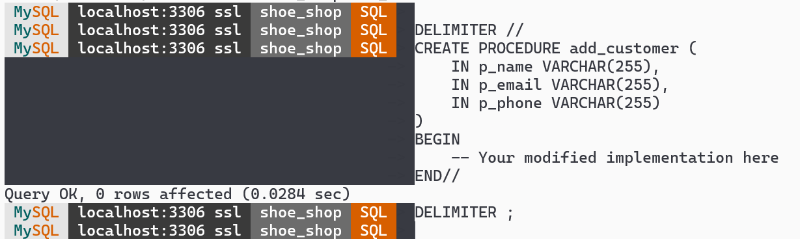


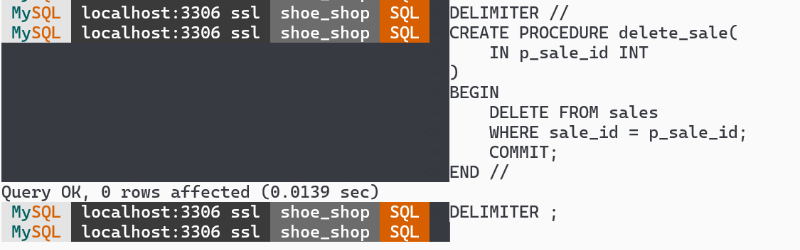
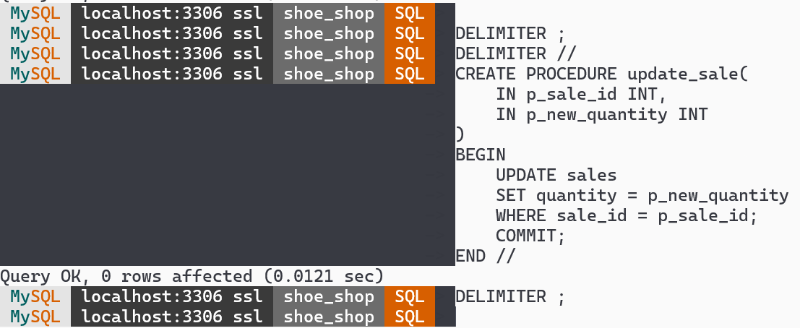
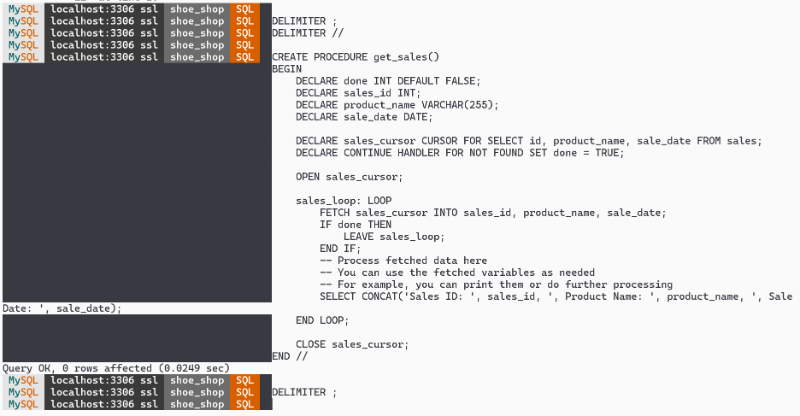
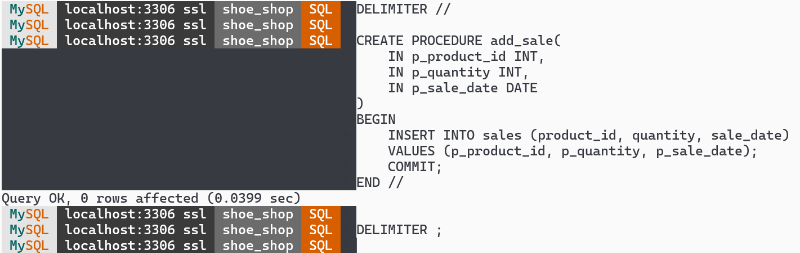


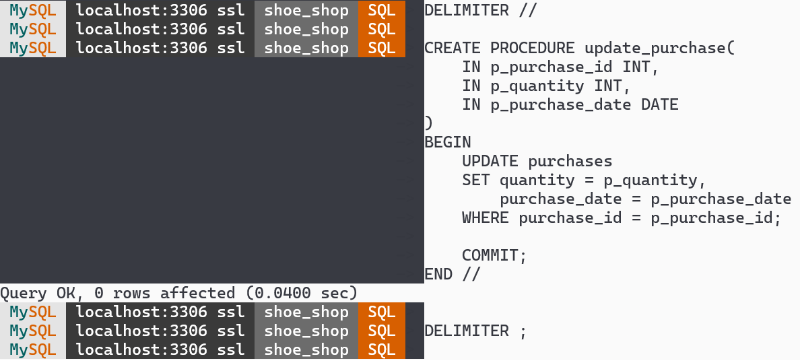
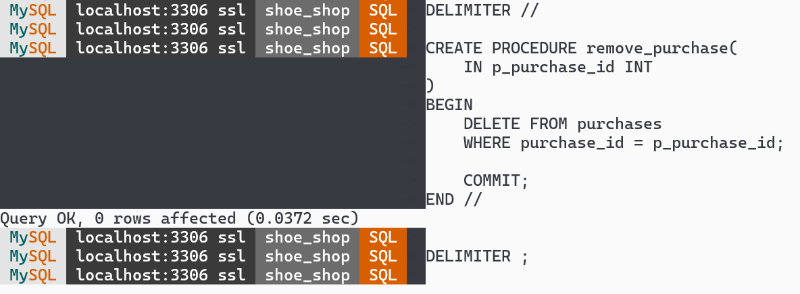
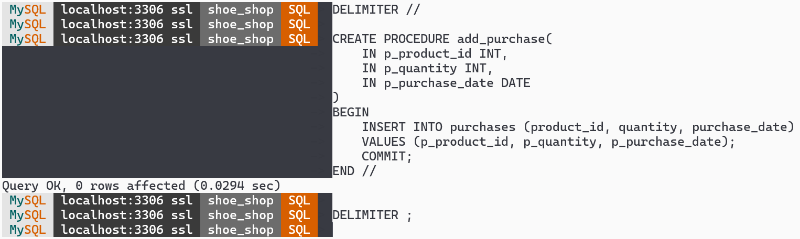


**PL/SQL procedures to Operate on the database of your project**

****

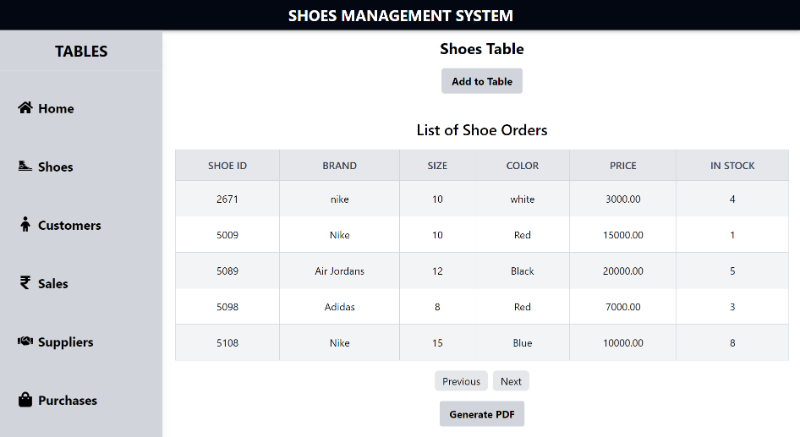
****

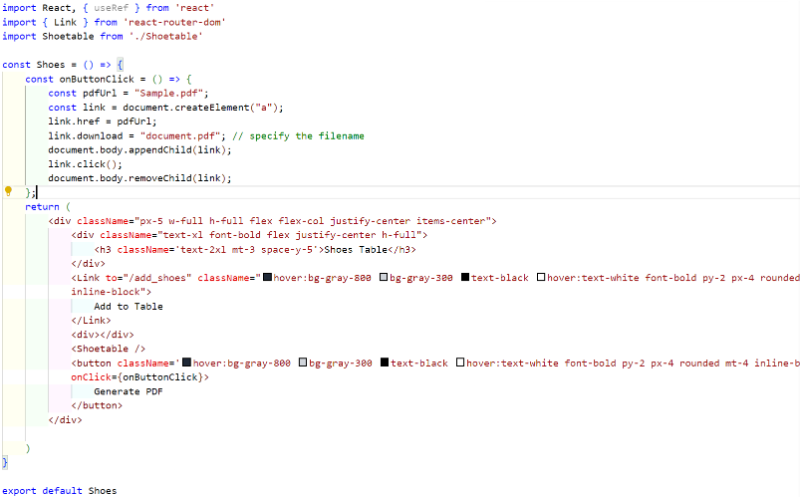
****

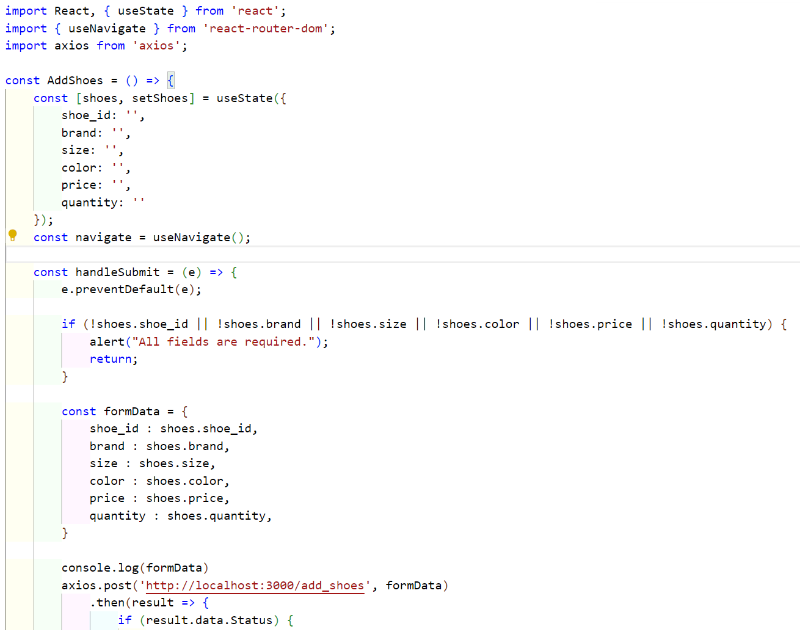
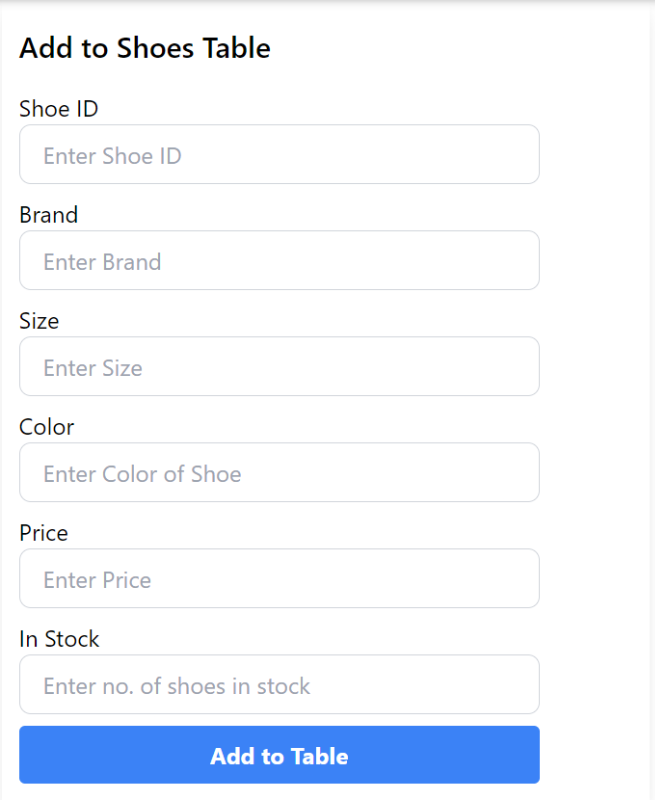
****

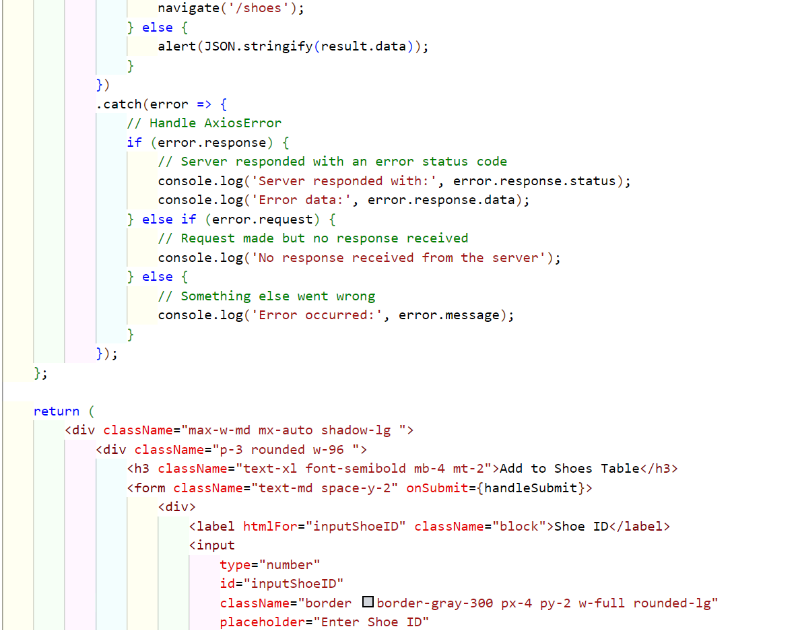
**Front End Design, Screen Shots & Sample Code for each module**

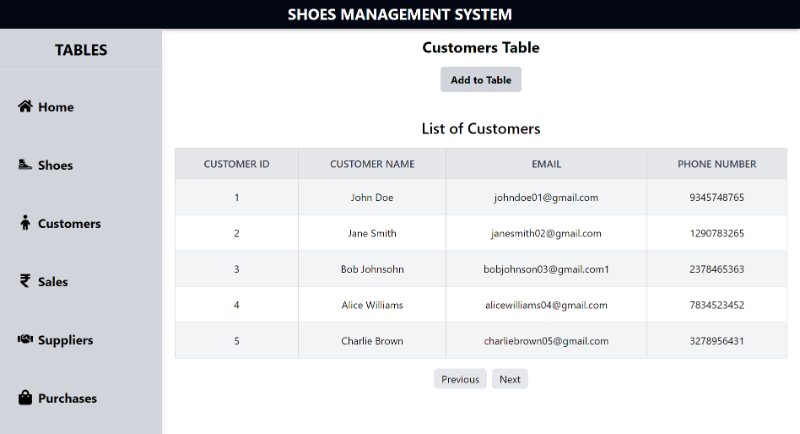
****

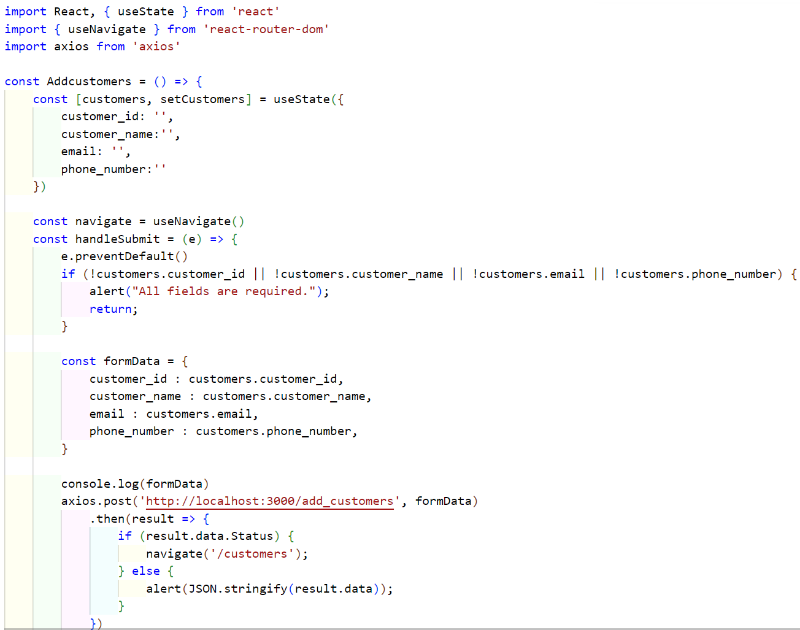
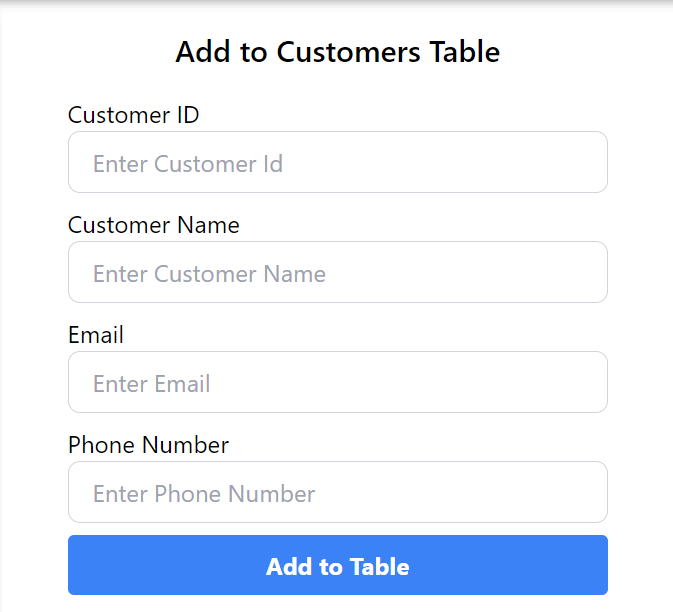
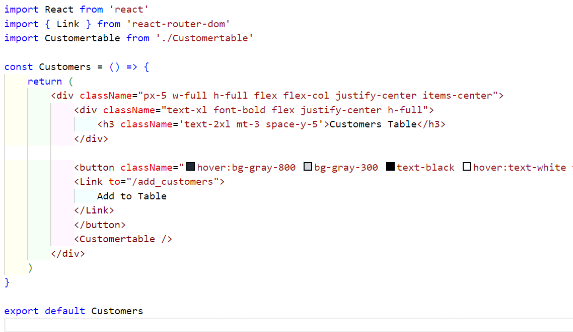
****

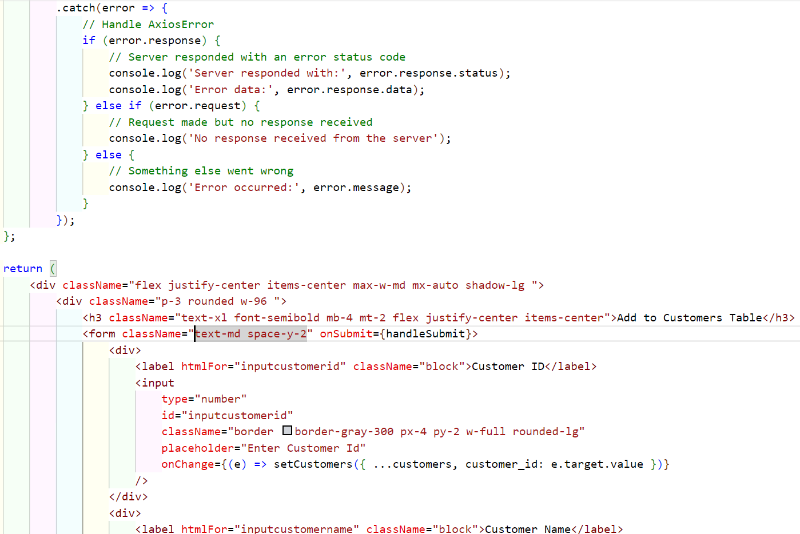
****

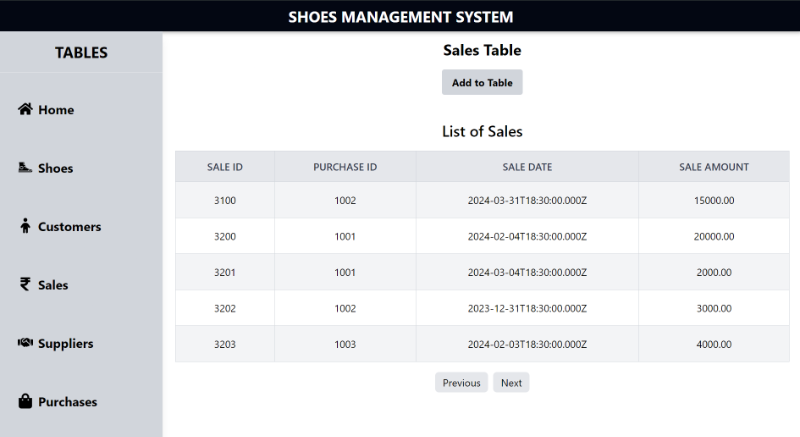
****

****

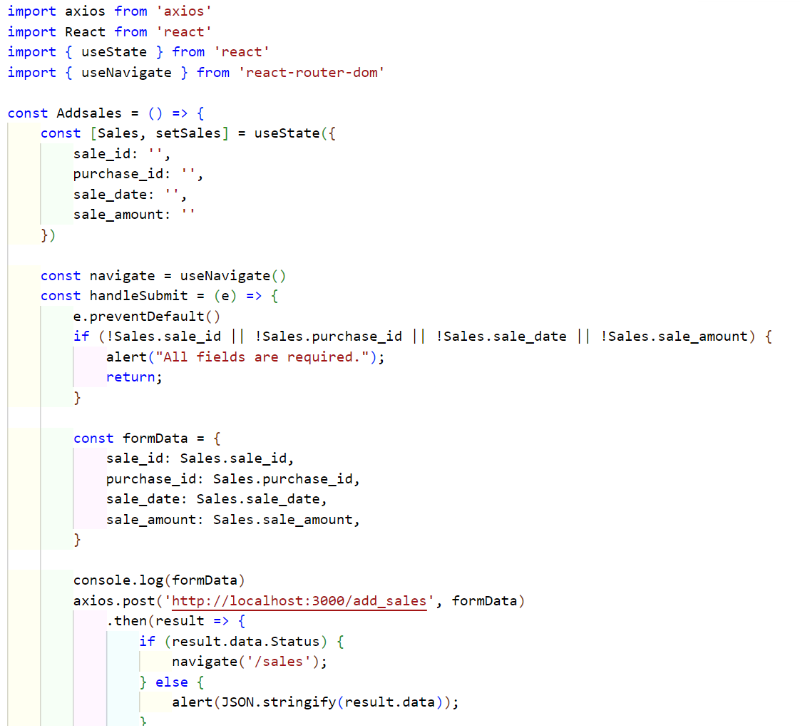
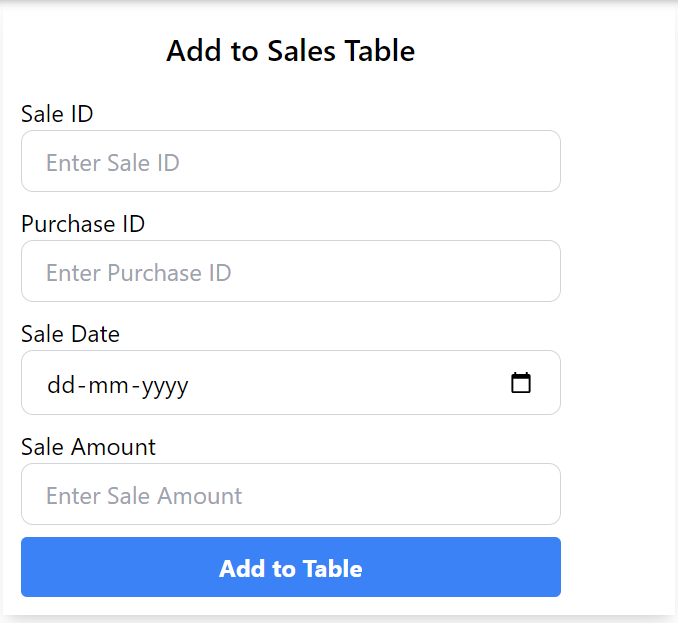
****

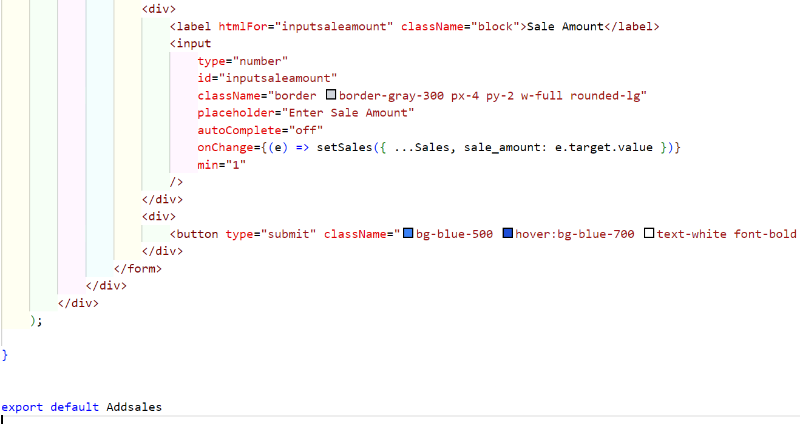
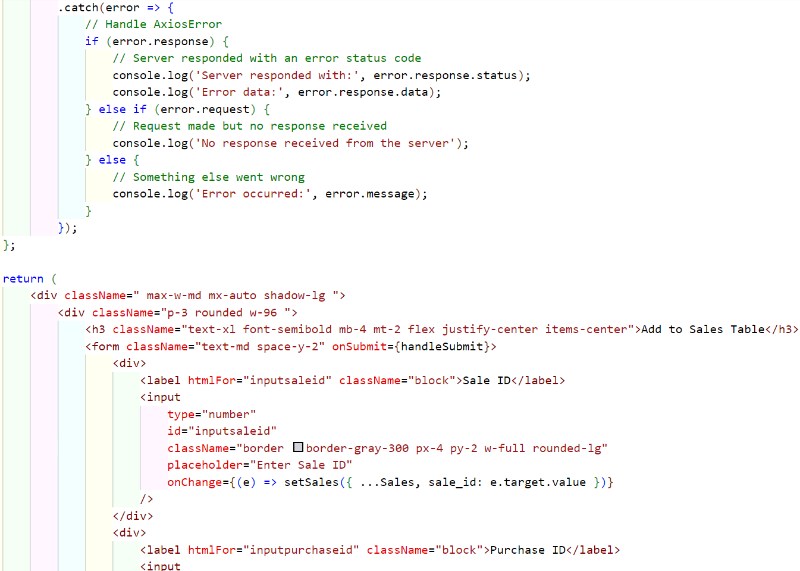
****

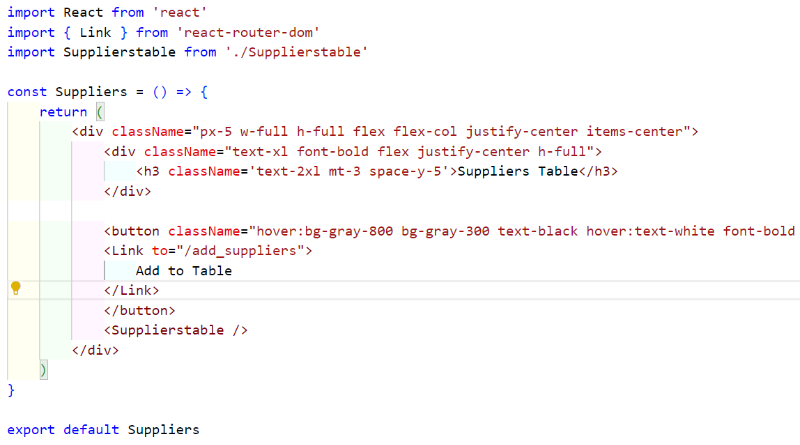
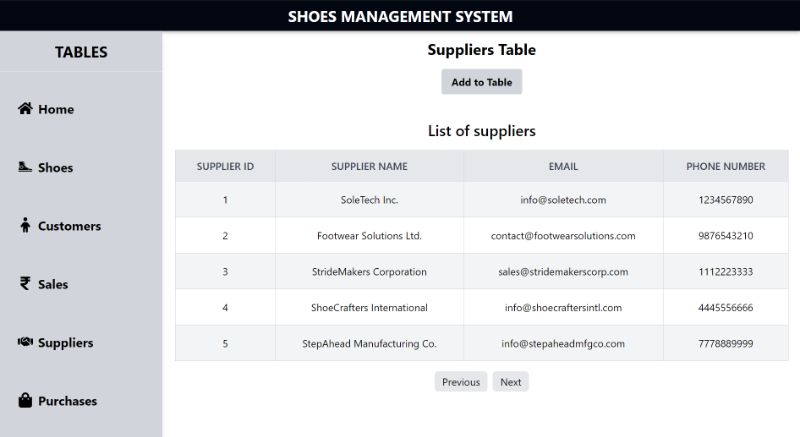
****

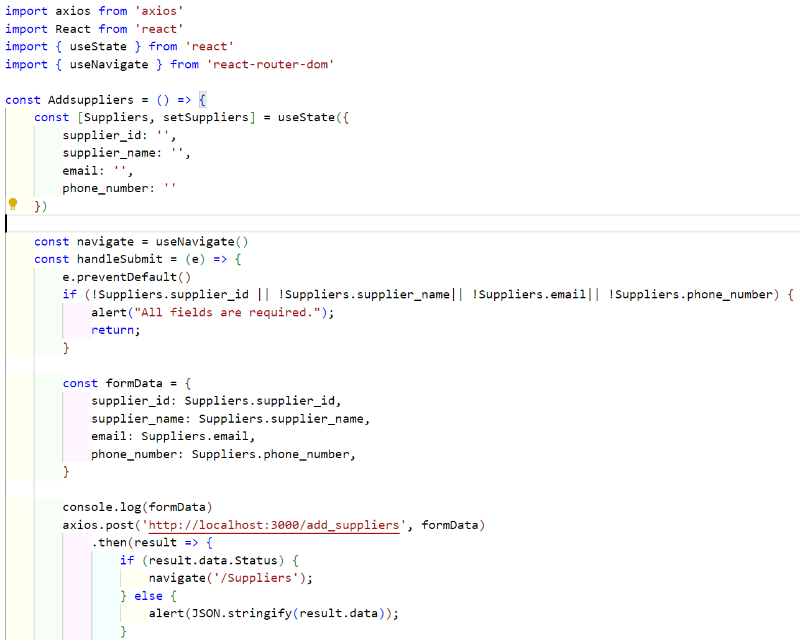
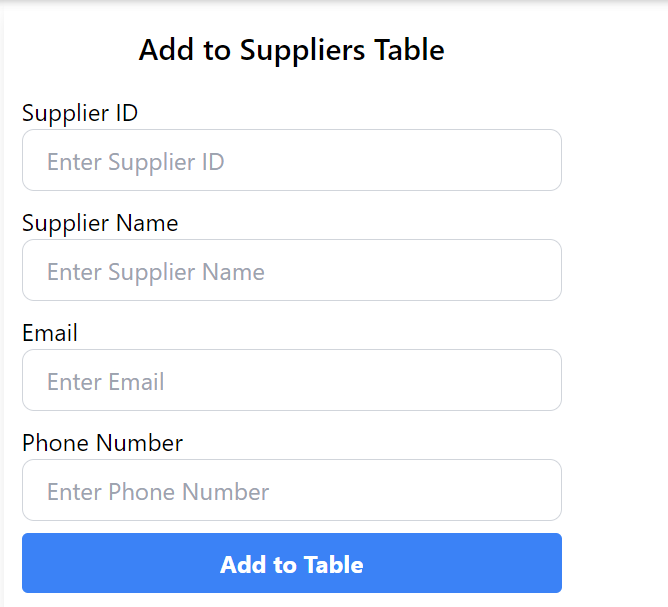
****

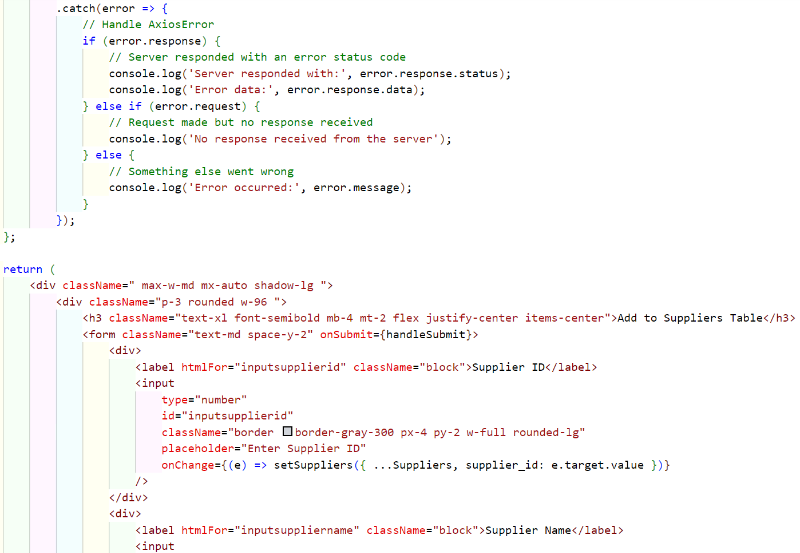
****

****

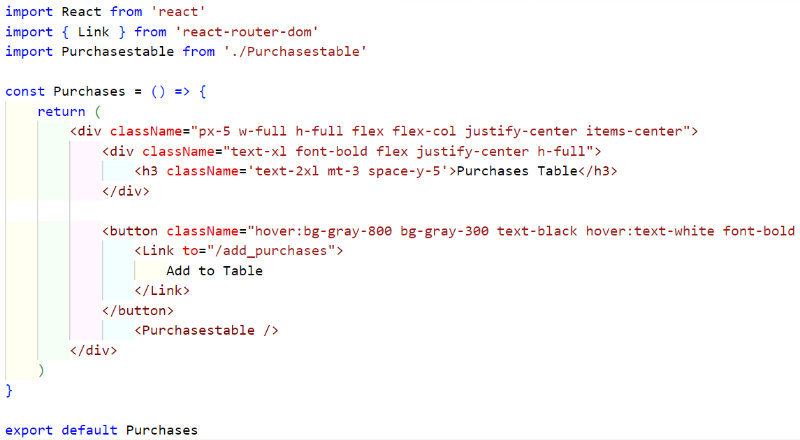
****

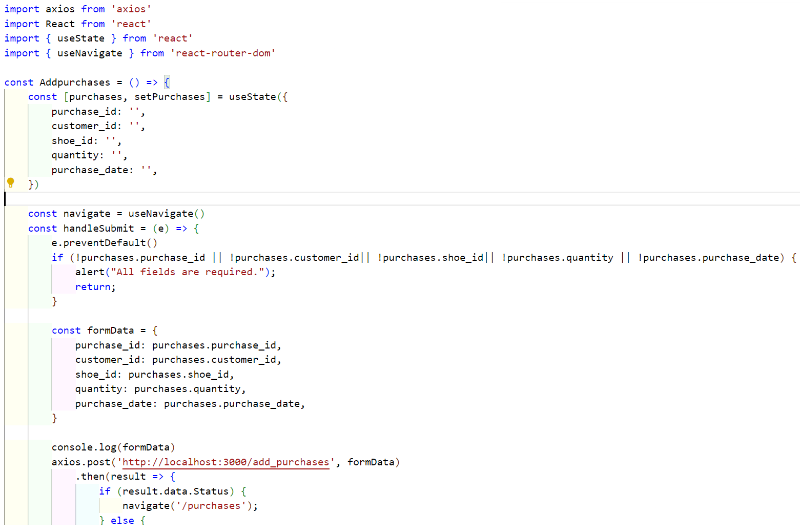
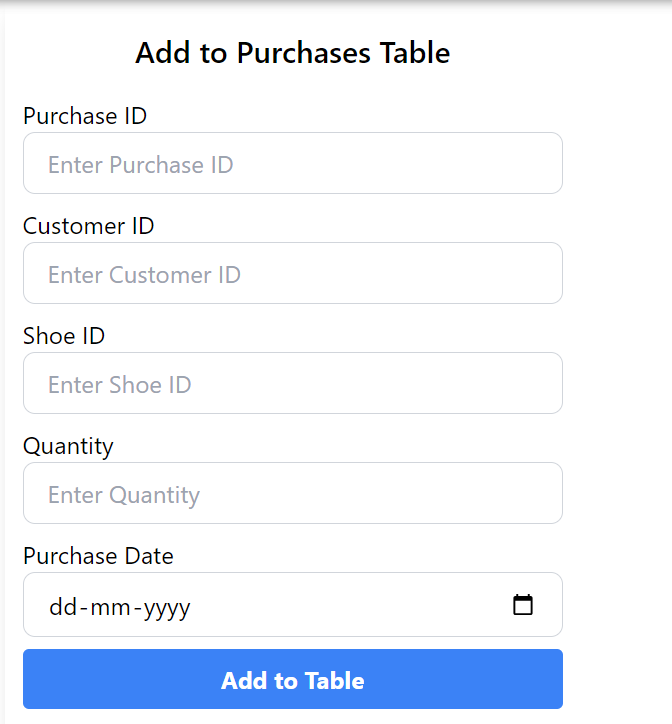
****

****

****

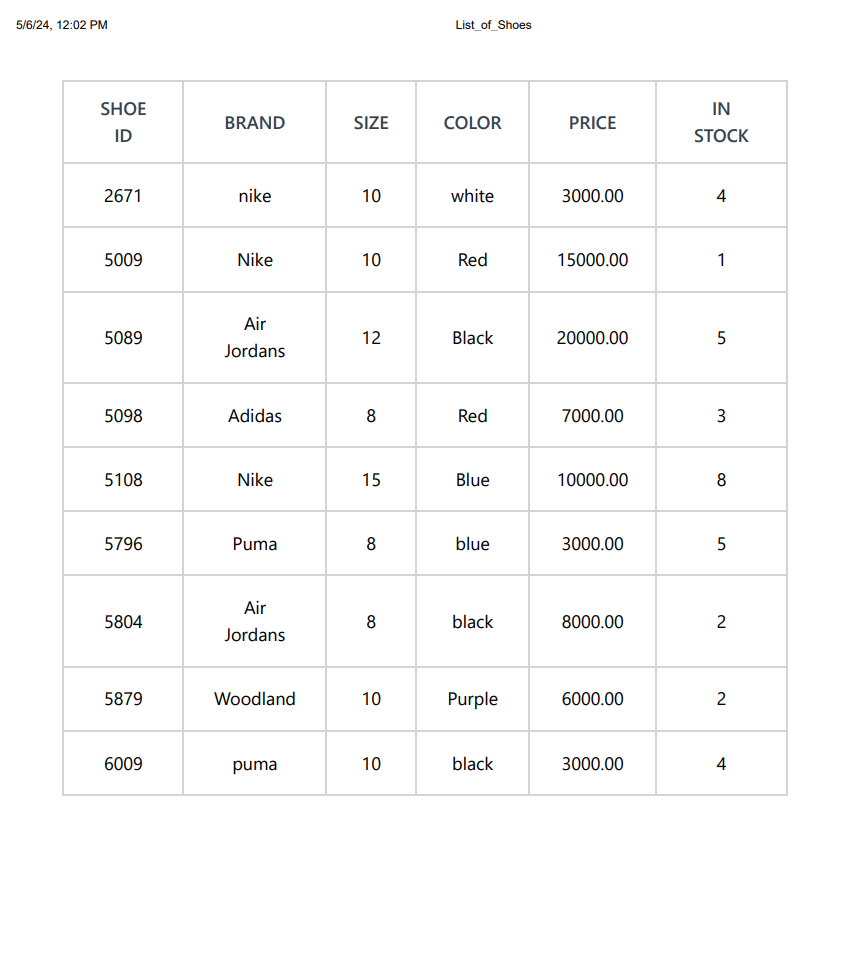
****

****

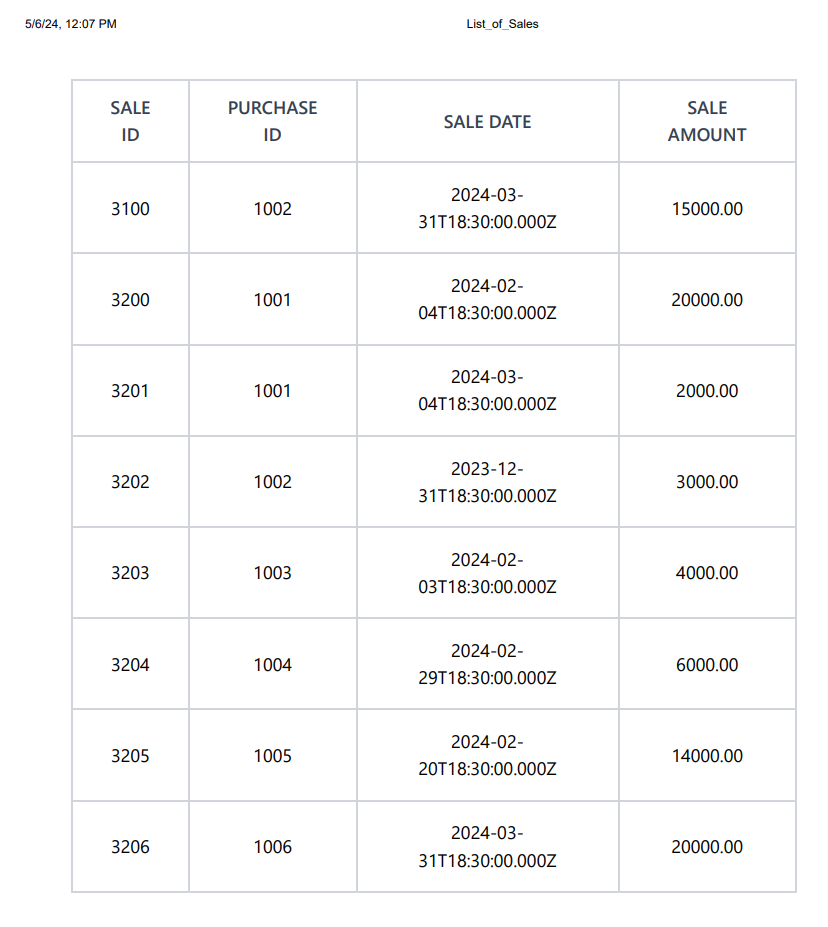
****

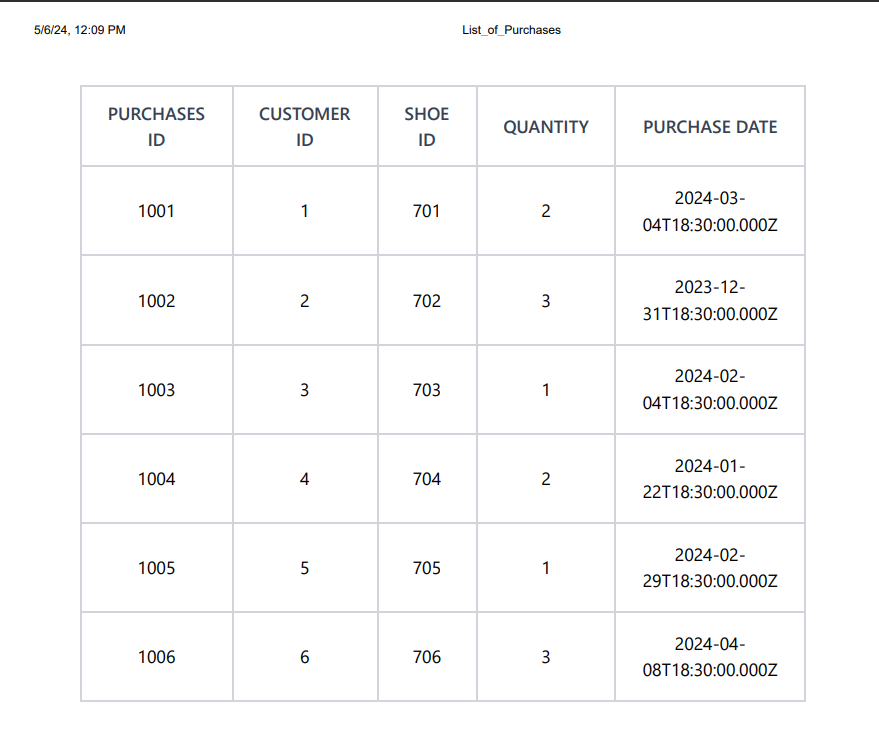
****

**Report Generation using Crystal Reports/Report Generation Tool**

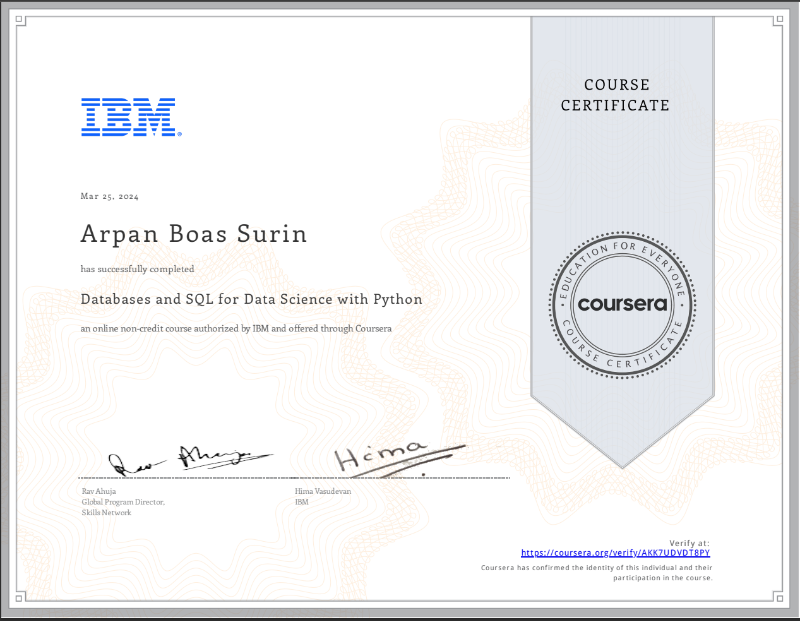
****

****

****

****

**Online Course Completion Certificate**

****