VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI - 590 018

A MINI PROJECT REPORT

on

"PETROL PUMP MANAGEMENT SYSTEM"

submitted by

NIKHIL KUMAR	4SF22CD401
KAVAN K T	4SF21CD011

In partial fulfillment of the requirements for the V semester

DBMS LABORATORY WITH MINI PROJECT

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

under the Guidance of

Mr. Vasudeva Rao P V

Assistant Professor

Department of ISE

at

SAHYADRI

College of Engineering & Management

An Autonomous Institution

Adyar, Mangaluru - 575 007

AY: 2023 - 24

SAHYADRI

College of Engineering & Management

An Autonomous Institution

Adyar, Mangaluru - 575 007

Department of Computer Science & Engineering (Data Science)

CERTIFICATE

This is to certify that the **Mini Project** entitled **"Petrol Pump Management System"** has been carried out by **Nikhil Kumar (4SF22CS401)** and **Kavan K T (4SF21CD011)**, the bonafide students of Sahyadri College of Engineering & Management in partial fulfillment of the requirements for the V semester **DBMS Laboratory with Mini Project (21CSL55)** of **Computer Science & Engineering (Data Science)** of Visvesvaraya Technological University, Belagavi during the Academic Year 2023 - 24. It is certified that all corrections/suggestions indicated for Continuous Internal Assessment have been incorporated in the report deposited in the departmental library of Information Science & Engineering. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work.

Mr. Vasudeva Rao P V	**Dr. Rithesh Pakkala P.**
Assistant Professor	Assoc.Professor & Head
Dept. of ISE, SCEM	Dept. of ISE & CSE(DS), SCEM

External Practical Examination:

Examiner's Name	Signature with Date
1.	....
2.	....

SAHYADRI

College of Engineering & Management

An Autonomous Institution

Adyar, Mangaluru - 575 007

Department of Computer Science & Engineering (Data Science)

DECLARATION

We hereby declare that the entire work embodied in this Mini Project Report titled **"Petrol Pump Management System"** has been carried out by us at Sahyadri College of Engineering & Management, Mangaluru under the supervision of **Mr. Vasudeva Rao P V** as the part of the V semester **DBMS Laboratory with Mini Project (21CSL55)** of **Bachelor of Engineering** in **Computer Science & Engineering (Data Science)**. This report has not been submitted to this or any other university.

Nikhil Kumar (4SF22CD401)

Kavan K T (4SF21CD011)

SCEM, Mangaluru

Abstract

Fuel consumption is increasing due to the growth of automobiles in the market. Unfortunately, if the vehicle stalls due to lack of fuel, for some reason people need to push a car or get help to reach the nearest gas station. In the above steps, the vehicle owner spends time and manual work.

This project uses MYSQL to store data and perform CRUD operations and Some of the famous libraries such as pandas and streamlit library for frontend to make User Interface interactive. This project will maintain data about Petrol Pumps in an area, their owners, Employees details working in that petrol, Customer details, Tanker details as well as Sales of a particular Petrol Pump.

Acknowledgement

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on **"Petrol Pump Management System"**. We have completed it as a part of the V semester **DBMS Laboratory with Mini Project (21CSL55)** of **Bachelor of Engineering** in **Computer Science & Engineering (Data Science)** of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guide, **Mr. Vasudeva Rao P V**, Assistant Professor, ce & Engineering (ISE) for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Rithesh Pakkala P.**, Assoc. Professor & Head, Department of ISE & CSE(DS) for his invaluable support and guidance.

We sincerely thank **Dr. S S Injaganeri**, Principal, Sahyadri College of Engineering & Management who have always been a great source of inspiration.

We further thank the non-teaching staff members of the Department of ISE, who have provided necessary support during the course of the work.

Finally, yet importantly, We express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

Nikhil Kumar
4SF22CD401
V Sem, B.E., CSE(DS)
SCEM, Mangaluru

Kavan K T
4SF21CD011
V Sem, B.E., CSE(DS)
SCEM, Mangaluru

Table of Contents

Abstract **i**

Acknowledgement **ii**

Table of Contents **iv**

List of Figures **v**

1 Introduction **1**

1.1 Database 1

1.2 Database Management System(DBMS) 1

1.3 Characteristics of the Database Approach 2

1.4 Advantages of using the DBMS Approach 2

1.5 Schemas 3

1.6 DBMS Languages 3

1.7 Motivational Challenges for the Project Work 3

1.8 Application of the Project 4

2 Conceptual Data Modeling **5**

2.1 Entity Relationship (ER) Model 6

2.2 Entities 6

2.3 Attributes 7

2.4 Relationships 7

2.5 Notation for ER Diagrams 8

3 Relational Data Model **11**

3.1 Relational Model Concepts 11

3.2 Relational Model Constraints 12

4 Structured Query Language(SQL) Programming **13**

4.1 SQL Data Definition and Data Types 13

4.2 Assertions 15

4.3 Triggers 15

4.4 Database Programming 15
4.5 Database Connection 16
4.6 Stored Procedures 16

5 Requirements Specification **17**
5.1 Hardware Requirements 17
5.2 Software Requirements 17

6 Relational Database Design **18**
6.1 Entity Relation(ER) Diagram 18
6.2 Mapping From ER Diagram to Relational Schema Diagram 19
6.3 Relational Schema Diagram 20

7 Implementation **22**
7.1 Table Structure 22
7.2 Codes Used For Modules: 24

8 Results and Discussion **28**

9 Conclusion and Future work **31**

References **31**

List of Figures

2.1 Summary of the Notation for ER Diagrams 10

6.1 ER Diagram of Petrol Pump Management System 18
6.2 Schema Diagram of Petrol Pump Management System 21

7.1 Customer Table 22
7.2 PetrolPump Table 22
7.3 Owner Table 23
7.4 Employee Table 23
7.5 Invoice Table 23
7.6 Tanker Table 24
7.7 Snippets for Insert 24
7.8 Snippets for Delete 25
7.9 Snippets for Update 25
7.10 Snippets for Update 26
7.11 Snippets for Trigger 26
7.12 Snippets for Stored Procedure 27

8.1 Invoice Details 28
8.2 Deleting Invoice 29
8.3 Updating Owner Details 29
8.4 Searching Customer ID 30

Chapter 1

Introduction

1.1 Database

A database is a structured repository that stores data in an organized manner, enabling efficient data management, retrieval, and manipulation. It serves as a central hub for storing large volumes of information, allowing users to interact with the data through queries and transactions. Structured storage formats, such as tables, rows, and columns, provide a framework for organizing diverse types of data, ensuring consistency and ease of access.

1.2 Database Management System(DBMS)

A Database Management System (DBMS) is a software application that facilitates the management of databases. It acts as an intermediary between users and the underlying database, providing a systematic and organized way to store, retrieve, and manipulate data. At its core, a DBMS is designed to efficiently handle large volumes of data while ensuring data integrity, security, and accessibility. One of the primary functions of a DBMS is data definition, where users can define the structure of the database schema. This includes specifying tables, their columns, data types, constraints, and relationships between tables. By providing tools for defining the database schema, DBMS enables users to organize and structure their data in a logical manner, making it easier to manage and query.

1.3 Characteristics of the Database Approach

A number of characteristics distinguish the database approach from the much older approach of writing customized programs to access data stored in files. In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application. For example, one user, the grade reporting office, may keep files on students and their grades. Programs to print a student's transcript and to enter new grades are implemented as part of the application. A second user, the accounting office, may keep track of students' fees and their payments. Although both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files.

The main characteristics of the database approach versus the file-processing approach are the following:

- Self-describing nature of a database system.
- Insulation between programs and data, and data abstraction.
- Support of multiple views of the data.
- Sharing of data and multiuser transaction processing.

1.4 Advantages of using the DBMS Approach

Advantages of using a DBMS and the capabilities that a good DBMS should possess. These capabilities are in addition to the four main characteristics discussed in Section 1.3. The DB must utilize these capabilities to accomplish a variety of objectives related to the design, administration, and use of a large multiuser database. The main advantages of using database are:

- Controlling Redundancy.
- Restricting Unauthorized Access.
- Providing Persistent Storage for Program Objects.
- Providing Storage Structures and Search.

1.5 Schemas

In a Database Management System (DBMS), schemas define the structure of the database. There are mainly three types:

1.Physical Schema: Defines how data is stored on the storage media.

2.Logical Schema: Defines the logical structure, including data organization and relationships.

3.External Schema: Defines user views of the database, providing tailored access to specific data.

1.6 DBMS Languages

In a Database Management System (DBMS), there are mainly four types of languages:

1.Data Definition Language (DDL):Used for creating, modifying, and deleting the structure of the database (like tables).

2.Data Manipulation Language (DML):Used for retrieving, adding, modifying, and deleting data from the database.

3.Data Control Language (DCL):Used for managing access permissions and security settings in the database.

4.Transaction Control Language (TCL):Used for managing transactions, which are groups of database operations that need to be executed together.

1.7 Motivational Challenges for the Project Work

Motivational challenges for the Petrol Pump Management System project involve understanding and addressing the complexities inherent in managing a petrol pump operation. One significant challenge lies in comprehensively mapping out the diverse array of tasks and processes involved, from inventory management to employee scheduling and customer service.Moreover, the need to develop a user-friendly interface that caters to the diverse needs of both petrol pump owners and customers poses another challenge. Additionally, integrating security measures to safeguard sensitive data such as financial transactions and customer information presents a critical consideration. Moreover, ensuring scalability and adaptability to accommodate future technological advancements and evolving industry standards adds another dimension to the project's complexity.

Addressing these motivational challenges requires a deep understanding of the operational dynamics of petrol pump management, coupled with robust database management strategies and efficient software development practices.

1.8 Application of the Project

This project enables flexible working practice,reduces manpower.This project involves managing information about the various products,staffs,customers etc.The user can consume less time in calculation and sales activity will be completed in fraction of seconds. The application of the Petrol Pump Management System offers a transformative solution for addressing the multifaceted challenges encountered in the day-to-day operations of petrol pump management. At its core, this system revolutionizes the traditional approach to petrol pump operations by digitizing and automating critical processes, thereby unlocking a myriad of benefits for both petrol pump owners and customers.
First and foremost, the system empowers petrol pump owners with unprecedented control and visibility over their operations. By centralizing essential functions such as inventory management, transaction handling, and employee scheduling, owners can streamline workflows, minimize operational inefficiencies, and optimize resource allocation. Real-time monitoring of fuel stock levels, coupled with automated alerts for low inventory, ensures uninterrupted fuel supply, mitigating the risk of revenue loss due to stock outs and improving overall business continuity.

Chapter 2

Conceptual Data Modeling

Conceptual modeling is a very important phase in designing a successful database application. Generally, the term database application refers to a particular database and the associated programs that implement the database queries and updates. For example, a BANK database application that keeps track of customer accounts would include programs that implement database updates corresponding to customer deposits and withdrawals. These programs would provide user-friendly graphical user interfaces (GUIs) utilizing forms and menus for the end users of the application—the bank customers or bank tellers in this example. In addition, it is now common to provide interfaces to these programs to BANK customers via mobile devices using mobile apps. Hence, a major part of the database application will require the design, implementation, and testing of these application programs. Traditionally, the design and testing of application programs has been considered to be part of software engineering rather than database design. In many software design tools, the database design methodologies and software engineering methodologies are intertwined since these activities are strongly related. In a Conceptual Data Model:

- Entities: These are the main things the organization cares about, like customers or products.
- Attributes: These are details about each thing, like a customer's name or a product's price.
- Relationships: These show how things are connected, like how customers buy products.

2.1 Entity Relationship (ER) Model

The Entity-Relationship (ER) Model is a conceptual framework used to describe the relationships between different entities in a database. It's a graphical representation that helps to visualize the relationships between entities and their attributes. In the ER Model:

1. Entities: Entities represent real-world objects or concepts, such as people, places, or things. Each entity is depicted as a rectangle in the diagram and has attributes that describe its properties.

2. Attributes: Attributes are the properties or characteristics of entities. They provide more detail about each entity. For example, a "Person" entity might have attributes like "Name," "Age," and "Address."

3. Relationships: Relationships represent how entities are connected or associated with each other. They show how data is related and can be one-to-one, one-to-many, or manyto-many. Relationships are depicted as lines connecting entities in the ER diagram.

The ER Model helps database designers and developers understand the structure of the data and how different entities relate to each other. It serves as a blueprint for designing the logical structure of a database, which can then be translated into a physical database schema. Overall, the ER Model provides a clear and visual way to represent the relationships between entities in a database, making it easier to design, develop, and maintain databases.

2.2 Entities

Entities, in the context of the Entity-Relationship (ER) Model used in database design, represent the basic building blocks of the data model. An entity is a real-world object or concept that is identifiable and distinguishable from other objects. In simpler terms, entities are things about which the organization wants to store information in a database.

Entities can be concrete, such as a person, place, or thing, or they can be abstract, such as an event or a concept. For example, in a university database, entities could include students, courses, instructors, and departments.

Each entity in the ER Model is depicted as a rectangle in diagrams, with the name of the entity written inside the rectangle. Additionally, entities have attributes that describe their properties or characteristics. For instance, a student entity might have attributes like student ID, name, date of birth, and email address.

Entities are essential components of the database design process as they help define the structure and organization of the data. They serve as the foundation upon which relationships are established and data is stored, managed, and retrieved within the database system.

2.3 Attributes

Attributes, within the context of the Entity-Relationship (ER) Model used in database design, represent the characteristics or properties of entities. They describe the details or qualities associated with each entity in a database.

In simpler terms, attributes are the specific pieces of information that we want to store about each entity. For example, if we have an entity representing a "Person," some attributes might include:

- Name - Age - Gender - Address - Email Each attribute provides additional detail about the entity and helps to define its characteristics. In a database diagram, attributes are typically depicted as ovals connected to the respective entity rectangle. Attributes can have different data types, such as text, numbers, dates, or binary data, depending on the nature of the information they represent.

Additionally, attributes can have constraints or rules associated with them, such as requiring a unique value or restricting the range of acceptable values. In summary, attributes are essential components of the database design process as they define the specific details or properties of each entity, enabling the storage, manipulation, and retrieval of data within the database system.

2.4 Relationships

In the context of the Entity-Relationship (ER) Model used in database design, relationships define how entities are connected or associated with each other. Relationships represent the connections between entities and describe the interactions or dependencies between them.

In simpler terms, relationships show how data in one entity is related to data in another entity. They help to illustrate the associations and interactions between different entities in a database. There are three main types of relationships in the ER Model:

• One-to-One (1:1): In a one-to-one relationship, each record in one entity is associated with exactly one record in another entity, and vice versa. For example, a person may have only one social security number, and each social security number is associated with only one person.

• One-to-Many (1:N): In a one-to-many relationship, each record in one entity can be associated with multiple records in another entity, but each record in the second entity is associated with only one record in the first entity. For example, a department can have multiple employees, but each employee works in only one department.

• Many-to-Many (M:N): In a many-to-many relationship, each record in one entity can be associated with multiple records in another entity, and vice versa. For example, students can enroll in multiple courses, and each course can have multiple students.

Relationships are typically depicted as lines connecting the entities in a database diagram. The lines indicate the direction of the relationship and may include labels to describe the nature of the association, such as ”works for” or ”belongs to.”

Overall, relationships play a crucial role in database design as they define the structure and connectivity between entities, enabling the representation of complex data interactions and dependencies within the database system.

2.5 Notation for ER Diagrams

In Entity-Relationship (ER) diagrams, several notations are used to visually represent entities, attributes, and relationships. The main components of ER diagrams and their notations are as follows:

Entity: Represented by a rectangle, an entity in an ER diagram signifies a real-world object, concept, or thing about which data is stored in the database. Each entity typically corresponds to a table in the relational database model.

Attribute: Attributes describe the properties or characteristics of an entity and are depicted as ovals connected to their respective entities by lines. They represent the data elements or fields within the entity and provide detailed information about the entity.

Primary Key: Denoted by underlining an attribute within an entity, the primary key uniquely identifies each record or row within the entity. It ensures the uniqueness and integrity of the data stored in the database table.

Composite Attribute: When an attribute can be further divided into smaller sub parts, it is known as a composite attribute. It is represented by an oval containing smaller ovals to indicate its constituent parts.

Multivalued Attribute: If an attribute can hold multiple values for a single entity occurrence, it is termed a multivalued attribute. It is represented by double ovals attached to the entity by a dashed line.

Derived Attribute: A derived attribute is one whose value is derived or computed from other attributes rather than being directly stored in the database. It is depicted by a dashed oval connected to its derived attributes.

Key Attributes: Key attributes, which uniquely identify each instance of an entity, are usually underlined within the attribute oval.

Primary Key and Foreign Key: Primary keys (PK) and foreign keys (FK) are often indicated within the entity rectangles. Primary keys are underlined attributes representing unique identifiers for each entity instance, while foreign keys are attributes that reference the primary key of another entity.

Overall, ER diagrams provide a visual representation of the database structure, making it easier to understand the relationships between entities and the attributes associated with them. They serve as important tools in database design and modeling.

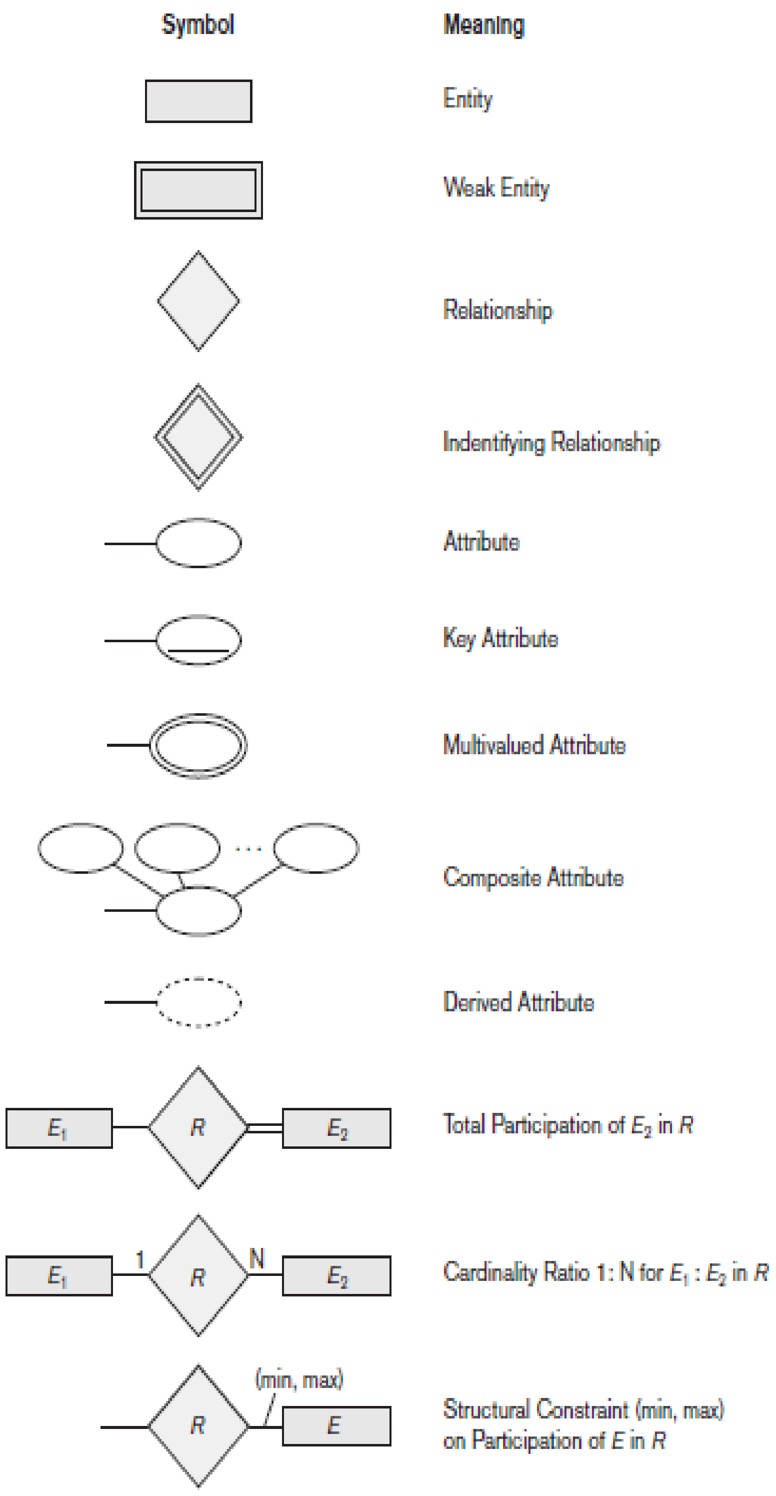

Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E1
R
E2
Total Participation of E2 in R
1
N
Cardinality Ratio 1: N for E1 : E2 in R
(min, max)
E
Structural Constraint (min, max) on Participation of E in R

Figure 2.1: Summary of the Notation for ER Diagrams

Chapter 3

Relational Data Model

3.1 Relational Model Concepts

The Relational Data Model is a conceptual framework for organizing and representing data in a relational database management system (RDBMS). Proposed by Edgar F. Codd in 1970, the relational model is based on mathematical set theory and predicate logic.

• Tables (Relations): Data is organized into tables, also known as relations. Each table represents an entity type, and each row in the table represents a specific instance of that entity. For example, a "Customers" table might store information about individual customers, with each row representing a different customer.

• Attributes (Columns): Each table consists of one or more columns, also known as attributes or fields. Columns represent the different properties or characteristics of the entities being modeled. For example, in a "Customers" table, attributes might include "Customer ID," "Name," "Address," and "Phone Number."

• Tuples (Rows): Rows, also known as tuples, represent individual records or instances within a table. Each row contains values for each attribute defined in the table's schema. For example, a row in the "Customers" table might contain the specific details (values) for a particular customer, such as their ID, name, address, and phone number.

• Keys: The relational model defines various types of keys to uniquely identify rows within a table. The primary key uniquely identifies each row in a table, while foreign keys establish relationships between tables by referencing the primary key of another table.

• Relationships: Relationships between tables are established using foreign keys. By referencing the primary key of one table in another table, relationships can be established to represent associations or dependencies between different entities.

• Normalization: The relational model supports normalization, a process used to organize data efficiently by reducing redundancy and dependency. Normalization involves breaking down large tables into smaller, more manageable tables and ensuring that each table represents a single entity type.

The Relational Data Model provides a flexible and standardized way to represent data, making it easier to store, retrieve, and manipulate information within a database system. It forms the basis for most modern relational database management systems, such as MySQL, PostgreSQL, Oracle, SQL Server, and SQLite.

3.2 Relational Model Constraints

Constraints on databases can generally be divided into three main categories:

• Constraints that are inherent in the data model. We call these inherent model-based constraints or implicit constraints.

• Constraints that can be directly expressed in the schemas of the data model, typically by specifying them in the DDL (data definition language). We call these schema-based constraints or explicit constraints.

• Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs or in some other way. We call these application-based or semantic constraints or business rules.

The constraint that a relation cannot have duplicate tuples is an inherent constraint. The constraints we discuss in this section are of the second category, namely, constraints that can be expressed in the schema of the relational model via the DDL. Constraints in the third category are more general, relate to the meaning as well as behavior of attributes, and are difficult to express and enforce within the data model, so they are usually checked within the application programs that perform database updates. In some cases, these constraints can be specified as assertions in SQL.

Chapter 4

Structured Query Language(SQL) Programming

4.1 SQL Data Definition and Data Types

SQL uses the terms table, row, and column for the formal relational model terms relation, tuple, and attribute, respectively. We will use the corresponding terms interchangeably. The main SQL command for data definition is the CREATE statement, which can be used to create schemas, tables (relations), types, and domains, as well as other constructs such as views, assertions, and triggers. Here are some commonly used SQL data types:

1. Numeric Data Types:

- INTEGER: Represents whole numbers.
- DECIMAL or NUMERIC: Represents fixed-point numbers with a specified precision and scale.
- FLOAT or REAL: Represents floating-point numbers with optional precision.
- DOUBLE or DOUBLE PRECISION: Represents double-precision floating-point numbers.

2. Character String Data Types:

- CHAR(n): Represents fixed-length character strings with a specified maximum length (n).
- VARCHAR(n): Represents variable-length character strings with a maximum length of (n).

- TEXT: Represents variable-length character strings with no specified maximum length.

3. Date and Time Data Types:

- DATE: Represents a date in YYYY-MM-DD format.
- TIME: Represents a time of day in HH:MM:SS format.
- TIMESTAMP: Represents a date and time combination in YYYY-MM-DD HH:MM:SS format.
- DATETIME: Similar to TIMESTAMP, represents a date and time combination.

4. Boolean Data Type:

- BOOLEAN or BOOL: Represents boolean values (TRUE, FALSE, or NULL).

5. Binary Data Types:

- BINARY(n): Represents fixed-length binary strings with a specified maximum length (n).
- VARBINARY(n): Represents variable-length binary strings with a maximum length of (n).
- BLOB: Represents variable-length binary data with no specified maximum length.

6. Other Data Types:

- ARRAY: Represents a collection of values of the same data type.
- JSON: Represents JSON (JavaScript Object Notation) data.
- XML: Represents XML (eXtensible Markup Language) data.

These are just a few examples of SQL data types. The specific data types available may vary depending on the SQL database management system (DBMS) being used, as different DBMSs may support additional data types or have variations in syntax and functionality.

4.2 Assertions

Assertions in database management systems serve as logical conditions or predicates designed to enforce data integrity rules that extend beyond the capabilities of traditional constraints like primary keys and foreign keys.

Unlike constraints that are applied at the table level, assertions allow for the specification of complex integrity checks that may involve multiple tables and rows within the database schema.

These declarative statements are automatically evaluated by the database system whenever triggering events, such as data modification operations, occur, ensuring that the defined conditions remain valid.

Despite their effectiveness in enhancing data quality and consistency, assertions may introduce overhead in terms of evaluation time and complexity, necessitating careful consideration of performance implications.

4.3 Triggers

Triggers, a fundamental feature of database systems, serve as automated procedures that execute in response to predefined events, facilitating dynamic data management and enforcing business rules without manual intervention.
These specialized stored procedures are designed to automatically respond to actions such as data insertion, updates, or deletions within specified database tables, thereby enhancing data integrity and enforcing consistency across the database environment.

A trigger's functionality is structured around its key components: the triggering event, trigger type (e.g., BEFORE or AFTER), triggering statement (e.g., INSERT, UPDATE, DELETE), and trigger body comprising SQL statements defining the actions to be executed upon event occurrence.

4.4 Database Programming

Database programming in Database Management Systems (DBMS) is like using a special language to talk to and work with the data stored in databases. Here's a simpler breakdown:

1. Getting Connected: You start by connecting your program to the database, providing details like where it's located and how to access it.

2. Asking Questions: Once connected, you can ask the database questions using a language called SQL. You can ask for specific data, like "give me all the customers," or "how many products do we have?"

3. Making Changes: You can also tell the database to make changes, like adding new data, updating existing data, or deleting data. For example, you might add a new employee or update a customer's address.

4. Dealing with Errors: Sometimes things don't go as planned, so you need to handle errors that might occur, like if the database is down or if there's a problem with your query.

4.5 Database Connection

Database connection is the process of establishing a communication link between a software application and a database management system (DBMS). It involves providing the necessary credentials such as username, password, and database URL to authenticate and connect to the database server. Once connected, the application can execute SQL queries to retrieve, manipulate, or update data stored in the database. The connection process typically involves using programming language-specific APIs or libraries, such as JDBC in Java, to handle the connection establishment, error handling, and resource management. After executing the necessary database operations, the connection should be properly closed to release resources and maintain system efficiency. 8

4.6 Stored Procedures

Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.
- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.

In general, many commercial DBMSs allow stored procedures and functions to be written in a general-purpose programming language.

Chapter 5

Requirements Specification

5.1 Hardware Requirements

- Processor : Any Processor more than 500 MHz
- RAM : 2GB
- Hard Disk : 500GB
- Input Device : Standard Keyboard and Mouse
- Output Device : Monitor

5.2 Software Requirements

- Database : MySQL
- Programming Language : Python
- Frontend : Streamlit
- IDE : Visual Studio
- Operating System : Windows 11

Chapter 6

Relational Database Design

6.1 Entity Relation(ER) Diagram

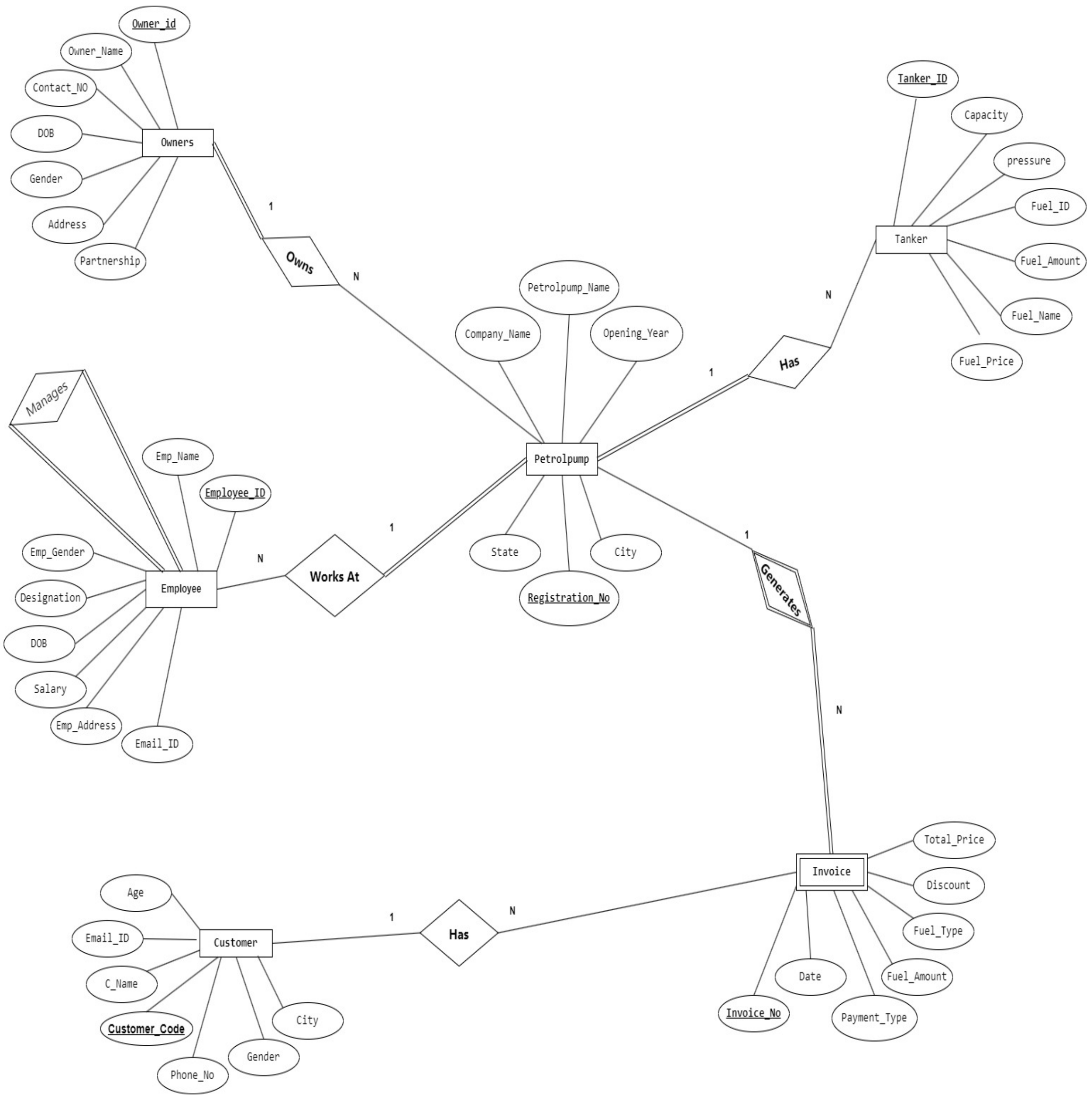


Figure 6.1: ER Diagram of Petrol Pump Management System

6.2 Mapping From ER Diagram to Relational Schema Diagram

1.Mapping of Regular Entities:This step involves mapping all the regular entity types to tabular format by identifying their primary keys.
2.Mapping of 1:1 Relation: In this step foreign keys are assigned using foreign key approach.The primary key of the participating relation R or S is added as primary key to second entity types by looking at the participating constraints.
3.Mapping of 1:N Relation: Foreign key approach is used to add one sided primary key to the n sided entity at foreign key.
4.Mapping of M:N Relation: Here we use the cross reference approach where the relationship is converted to a new relation within attributes on primary keys of both participating relation.
5.Mapping of Weak Entity: When mapping weak entity types along with other attributes the partial key and primary key of parent entity together will form their primary key of the new relation.
6.Mapping of N-ary Relation: For mapping N array relationship we create a new relation with a relationship name in its attribute and primary keys of all participating entity types.
7.Mapping of Multivalued Relation: For multivalued attributes a separate relation has to be created along with primary key of parent relation.

In our database we have the following mappings:
Step – 1 : Mapping of Regular Entities.
From the ER diagram we identify all the strong entities E and create a relation R that includes all it's simple attributes and primary keys.
The following are the strong entities from our schema diagram :
1.OWNERS(Ownerid)
2.PETROLPUMP(Registrion No,Ownerid)
3.EMPLOYEE(Employee ID,ManagerId,EppNo)
4.TANKER(Tanker Id,TppNo)
5.CUSTOMER(Customer Code)
6.INVOICE(Invoice No,CustomerCode,IppNo)

Step – 2 : Mapping of binary 1:N Relation Types.

Owners:

One-to-Many relation with Petrolpump: One owner can own multiple petrol pumps.

Petrolpump:

Many-to-One relation with Owners: Each petrol pump is owned by one owner. One-to-Many relation with Employee: Each petrol pump can have multiple employees. One-to-Many relation with Invoice: Each petrol pump can have multiple invoices. One-to-Many relation with Tanker: Each petrol pump can have multiple tankers. Employee:

Many-to-One relation with Petrolpump: Each employee works at one petrol pump.

Customer:

One-to-Many relation with Invoice: Each customer can have multiple invoices. Tanker:

Many-to-One relation with Petrolpump: Each tanker is associated with one petrol pump. These mappings describe the relationships between the entities in your database schema.

6.3 Relational Schema Diagram

A Schema is a pictorial representation of the relationship between the database tables in the database that is created. The database schema of a database system is its structure described in a formal language sup ported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language.The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modelled in the database.

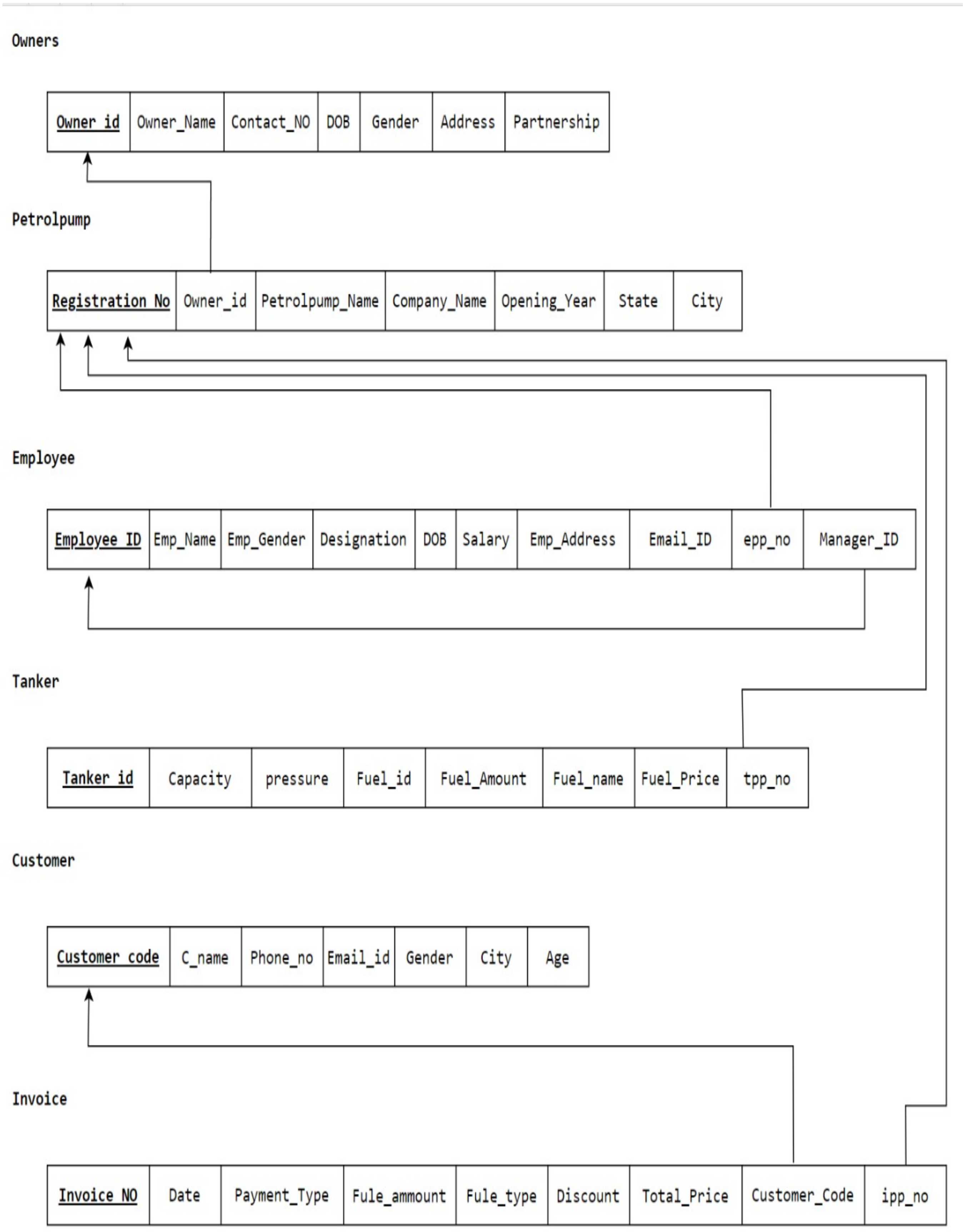


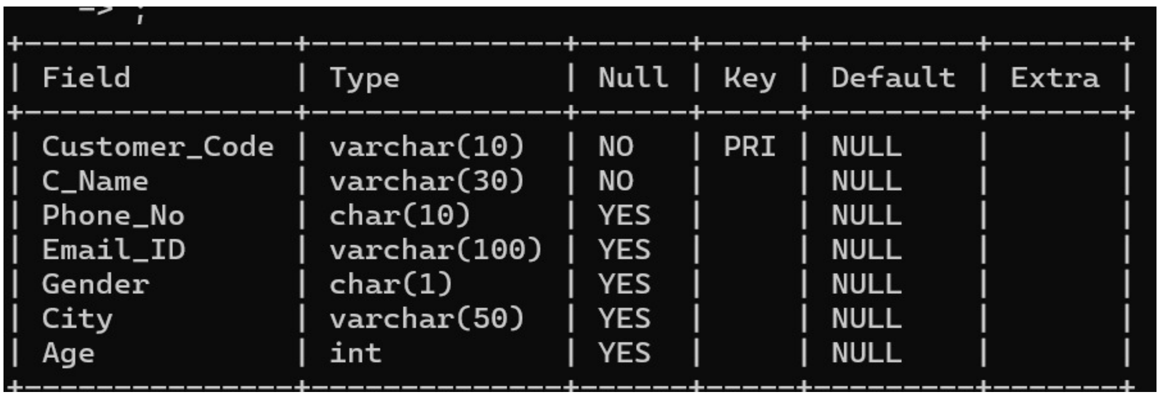
Figure 6.2: Schema Diagram of Petrol Pump Management System

Chapter 7

Implementation

7.1 Table Structure

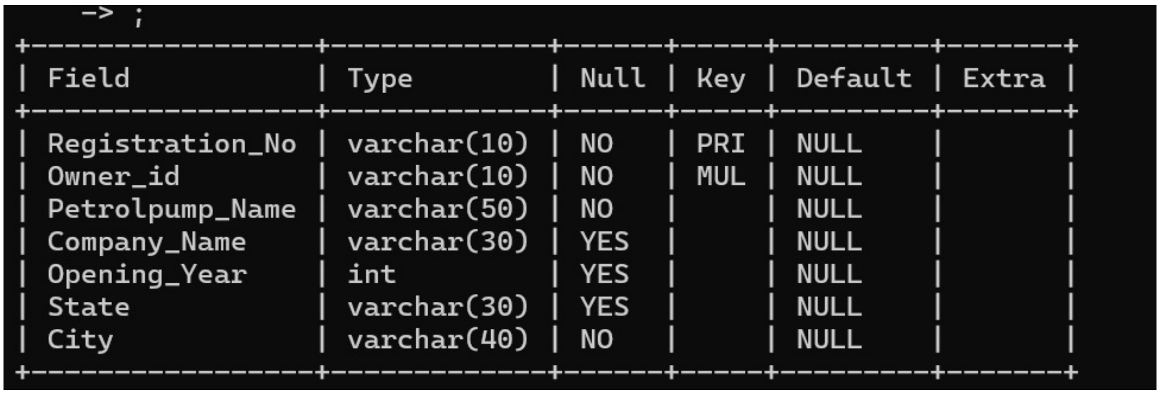
CUSTOMER



```
    -> ;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| Customer_Code | varchar(10)  | NO   | PRI | NULL    |       |
| C_Name        | varchar(30)  | NO   |     | NULL    |       |
| Phone_No      | char(10)     | YES  |     | NULL    |       |
| Email_ID      | varchar(100) | YES  |     | NULL    |       |
| Gender        | char(1)      | YES  |     | NULL    |       |
| City          | varchar(50)  | YES  |     | NULL    |       |
| Age           | int          | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

Figure 7.1: Customer Table

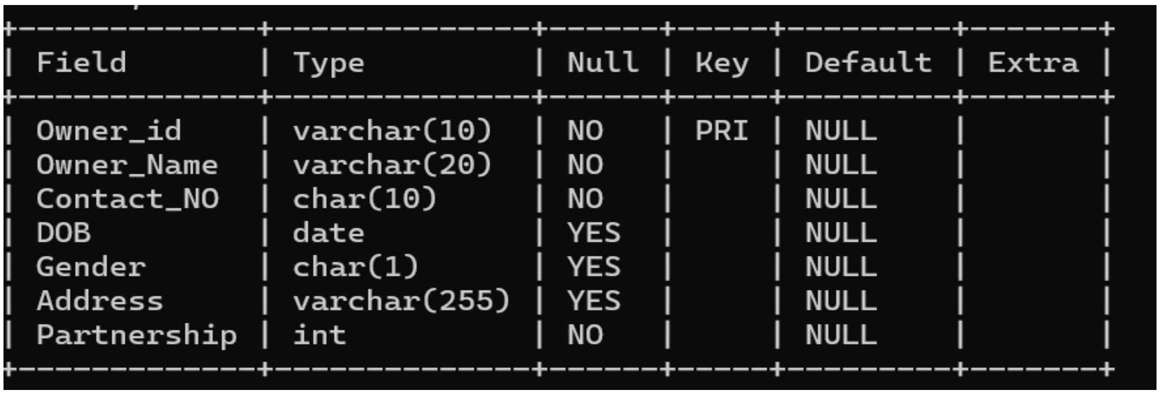
PETROLPUMP



```
    -> ;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| Registration_No  | varchar(10) | NO   | PRI | NULL    |       |
| Owner_id         | varchar(10) | NO   | MUL | NULL    |       |
| Petrolpump_Name  | varchar(50) | NO   |     | NULL    |       |
| Company_Name     | varchar(30) | YES  |     | NULL    |       |
| Opening_Year     | int         | YES  |     | NULL    |       |
| State            | varchar(30) | YES  |     | NULL    |       |
| City             | varchar(40) | NO   |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
```

Figure 7.2: PetrolPump Table

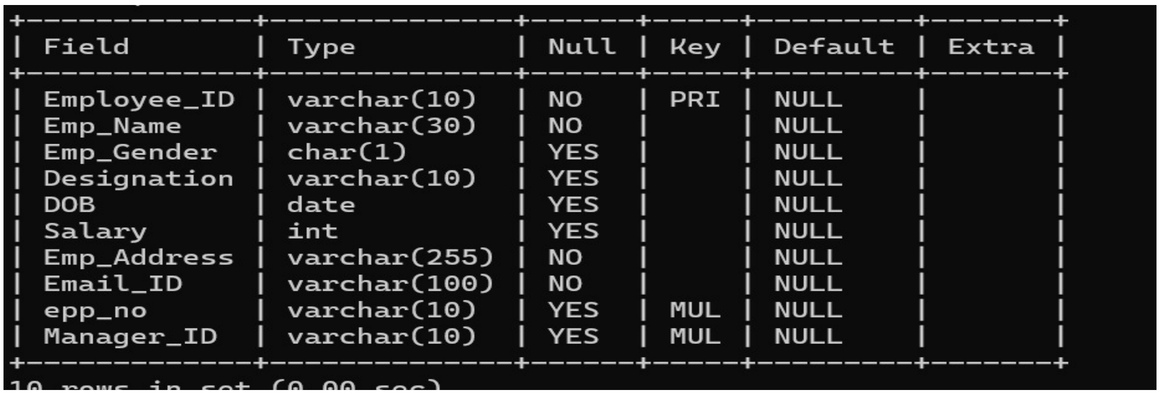
OWNER



```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Owner_id    | varchar(10)  | NO   | PRI | NULL    |       |
| Owner_Name  | varchar(20)  | NO   |     | NULL    |       |
| Contact_NO  | char(10)     | NO   |     | NULL    |       |
| DOB         | date         | YES  |     | NULL    |       |
| Gender      | char(1)      | YES  |     | NULL    |       |
| Address     | varchar(255) | YES  |     | NULL    |       |
| Partnership | int          | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

Figure 7.3: Owner Table

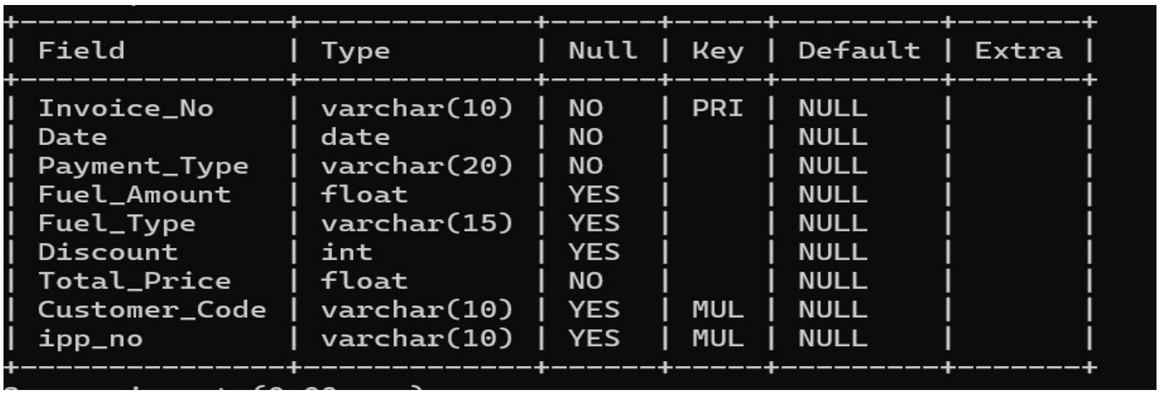
EMPLOYEE



```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Employee_ID | varchar(10)  | NO   | PRI | NULL    |       |
| Emp_Name    | varchar(30)  | NO   |     | NULL    |       |
| Emp_Gender  | char(1)      | YES  |     | NULL    |       |
| Designation | varchar(10)  | YES  |     | NULL    |       |
| DOB         | date         | YES  |     | NULL    |       |
| Salary      | int          | YES  |     | NULL    |       |
| Emp_Address | varchar(255) | NO   |     | NULL    |       |
| Email_ID    | varchar(100) | NO   |     | NULL    |       |
| epp_no      | varchar(10)  | YES  | MUL | NULL    |       |
| Manager_ID  | varchar(10)  | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

Figure 7.4: Employee Table

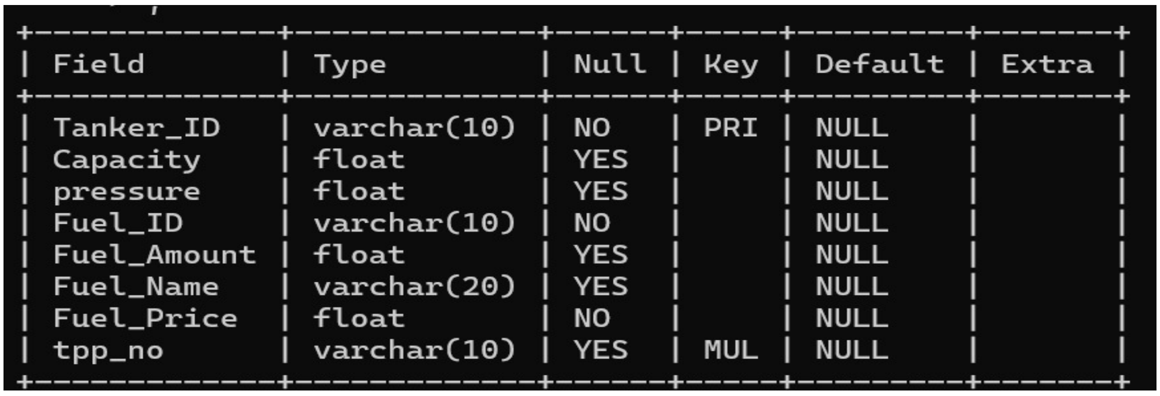
INVOICE



```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Invoice_No    | varchar(10) | NO   | PRI | NULL    |       |
| Date          | date        | NO   |     | NULL    |       |
| Payment_Type  | varchar(20) | NO   |     | NULL    |       |
| Fuel_Amount   | float       | YES  |     | NULL    |       |
| Fuel_Type     | varchar(15) | YES  |     | NULL    |       |
| Discount      | int         | YES  |     | NULL    |       |
| Total_Price   | float       | NO   |     | NULL    |       |
| Customer_Code | varchar(10) | YES  | MUL | NULL    |       |
| ipp_no        | varchar(10) | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

Figure 7.5: Invoice Table

TANKER

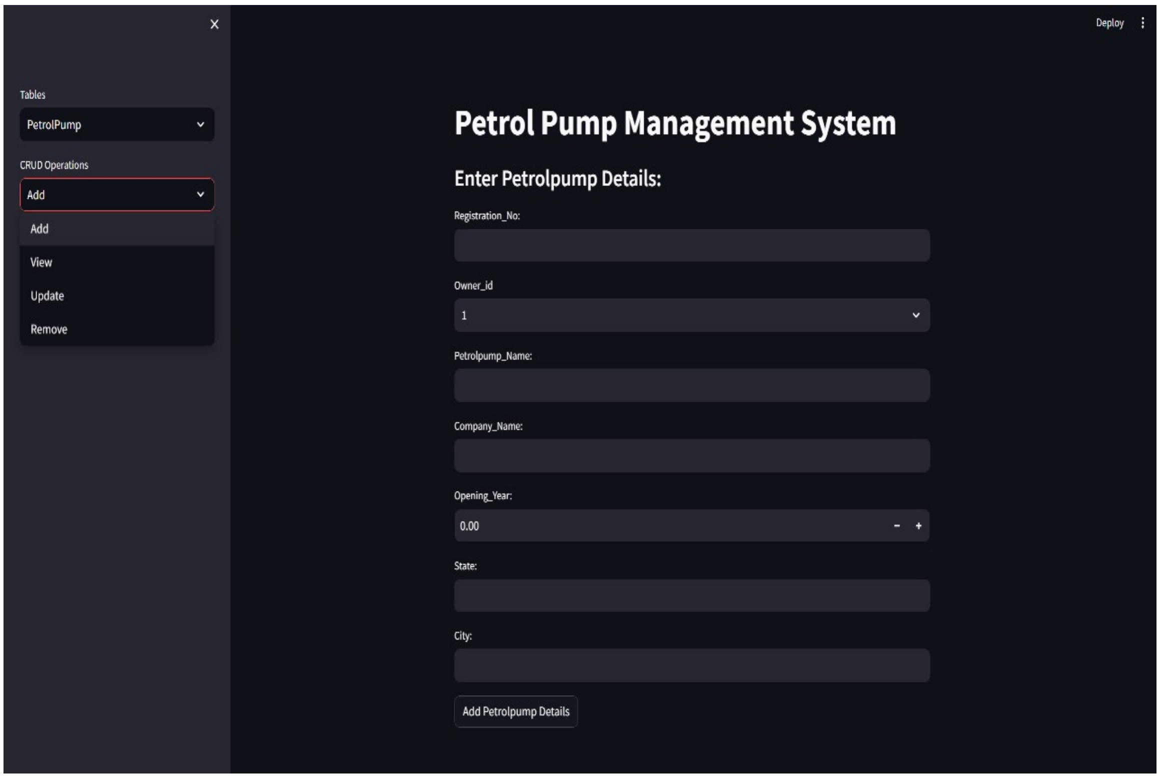


```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Tanker_ID   | varchar(10) | NO   | PRI | NULL    |       |
| Capacity    | float       | YES  |     | NULL    |       |
| pressure    | float       | YES  |     | NULL    |       |
| Fuel_ID     | varchar(10) | NO   |     | NULL    |       |
| Fuel_Amount | float       | YES  |     | NULL    |       |
| Fuel_Name   | varchar(20) | YES  |     | NULL    |       |
| Fuel_Price  | float       | NO   |     | NULL    |       |
| tpp_no      | varchar(10) | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

Figure 7.6: Tanker Table

7.2 Codes Used For Modules:

Insert



Tables: PetrolPump

CRUD Operations: Add

Petrol Pump Management System

Enter Petrolpump Details:

Registration_No:

Owner_id: 1

Petrolpump_Name:

Company_Name:

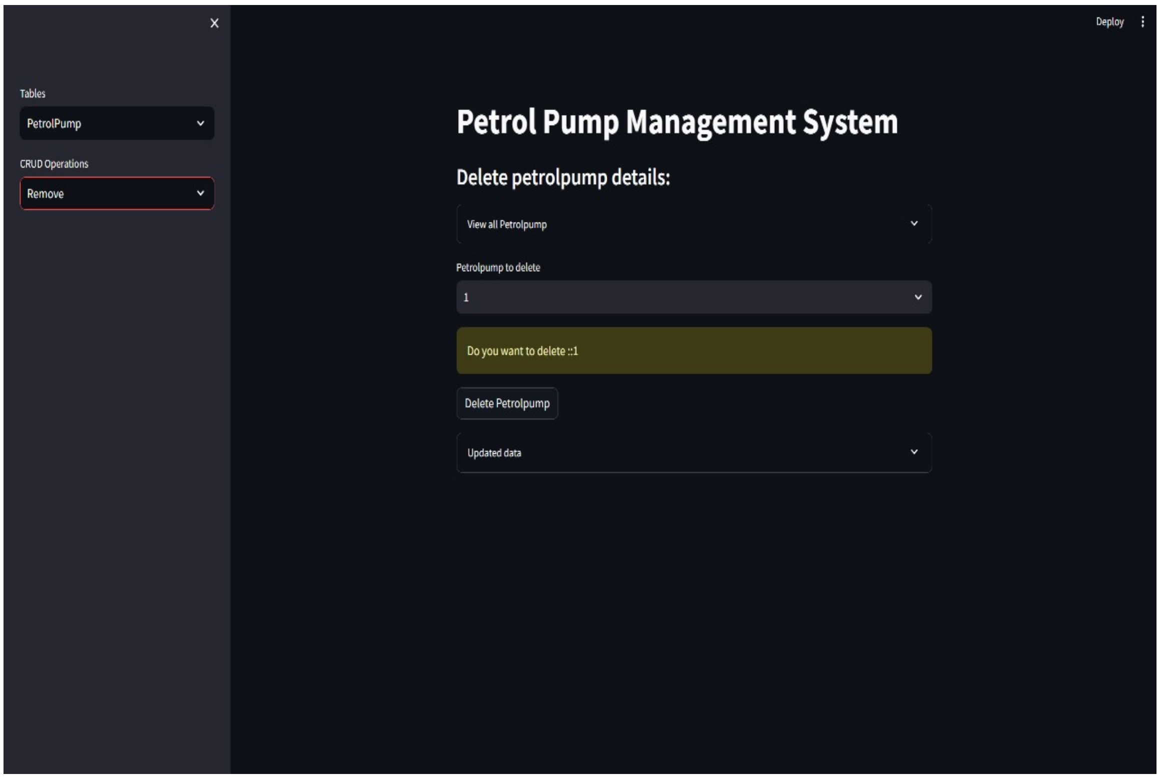
Opening_Year: 0.00

State:

City:

Figure 7.7: Snippets for Insert

Delete



Petrol Pump Management System

Delete petrolpump details:

View all Petrolpump

Petrolpump to delete: 1

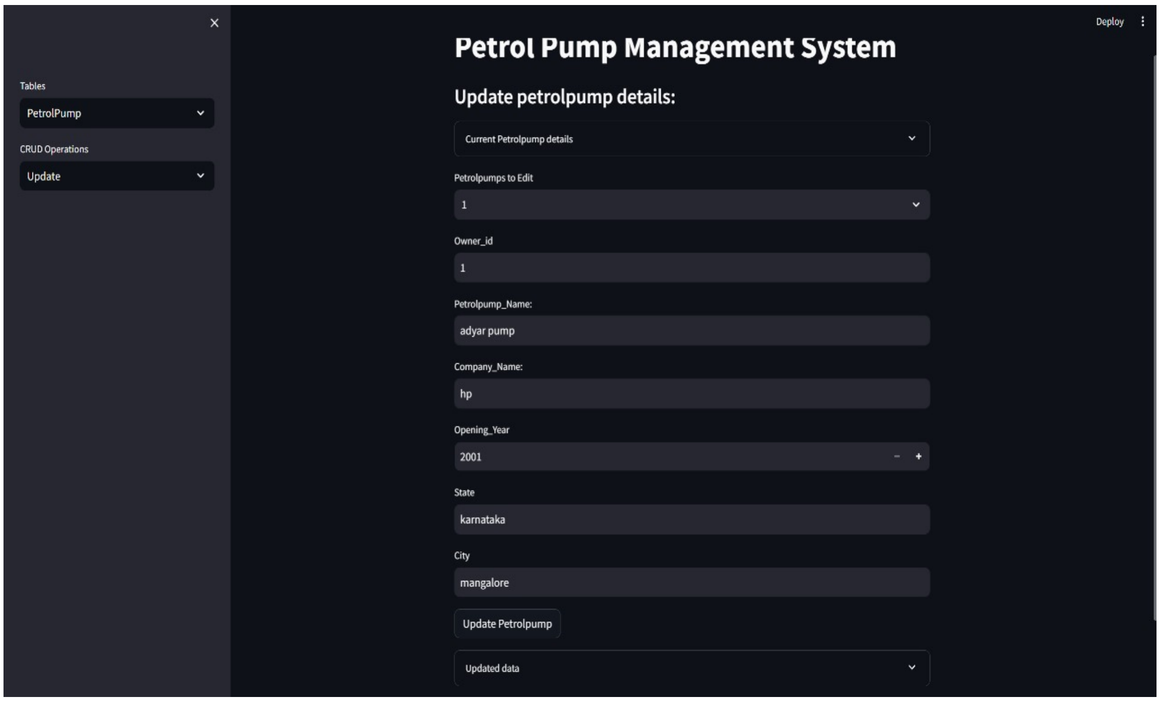
Do you want to delete ::1

Delete Petrolpump

Updated data

Figure 7.8: Snippets for Delete

Update



Petrol Pump Management System

Update petrolpump details:

Current Petrolpump details

Petrolpumps to Edit: 1

Owner_id: 1

Petrolpump_Name: adyar pump

Company_Name: hp

Opening_Year: 2001

State: karnataka

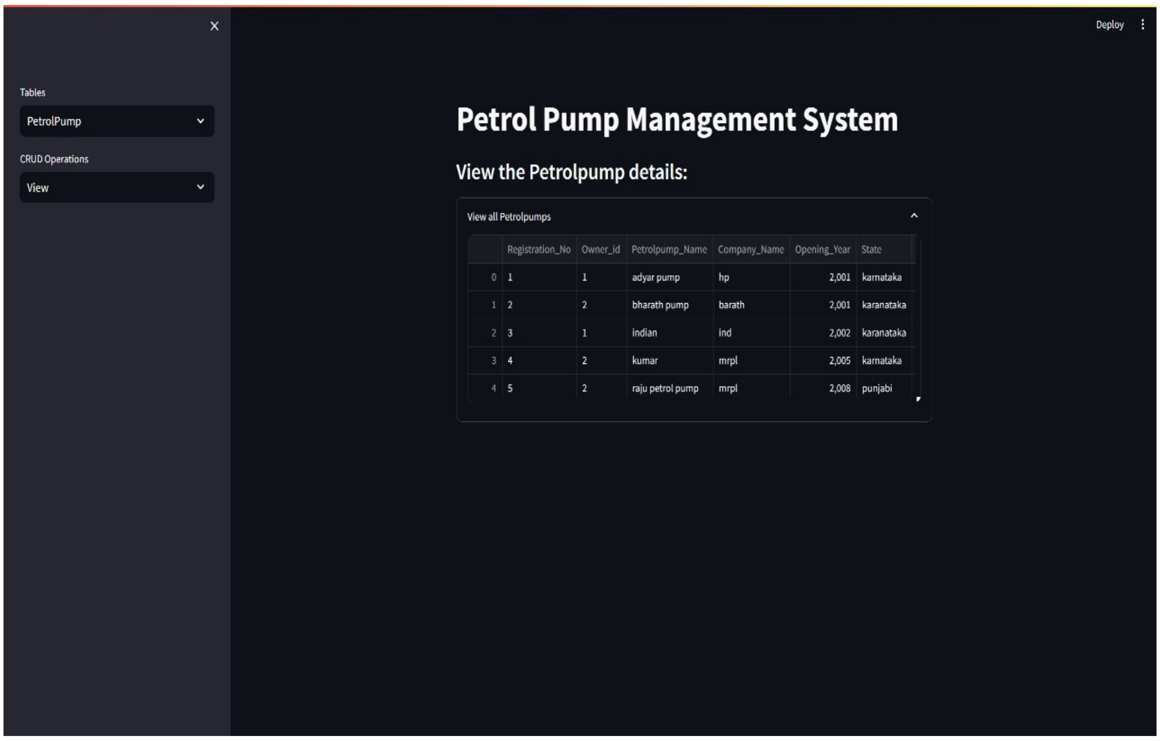
City: mangalore

Update Petrolpump

Updated data

Figure 7.9: Snippets for Update

Display



Petrol Pump Management System

View the Petrolpump details:

View all Petrolpumps

	Registration_No	Owner_id	Petrolpump_Name	Company_Name	Opening_Year	State
0	1	1	adyar pump	hp	2,001	karnataka
1	2	2	bharath pump	barath	2,001	karanataka
2	3	1	indian	ind	2,002	karanataka
3	4	2	kumar	mrpl	2,005	karnataka
4	5	2	raju petrol pump	mrpl	2,008	punjabi

Figure 7.10: Snippets for Update

Trigger

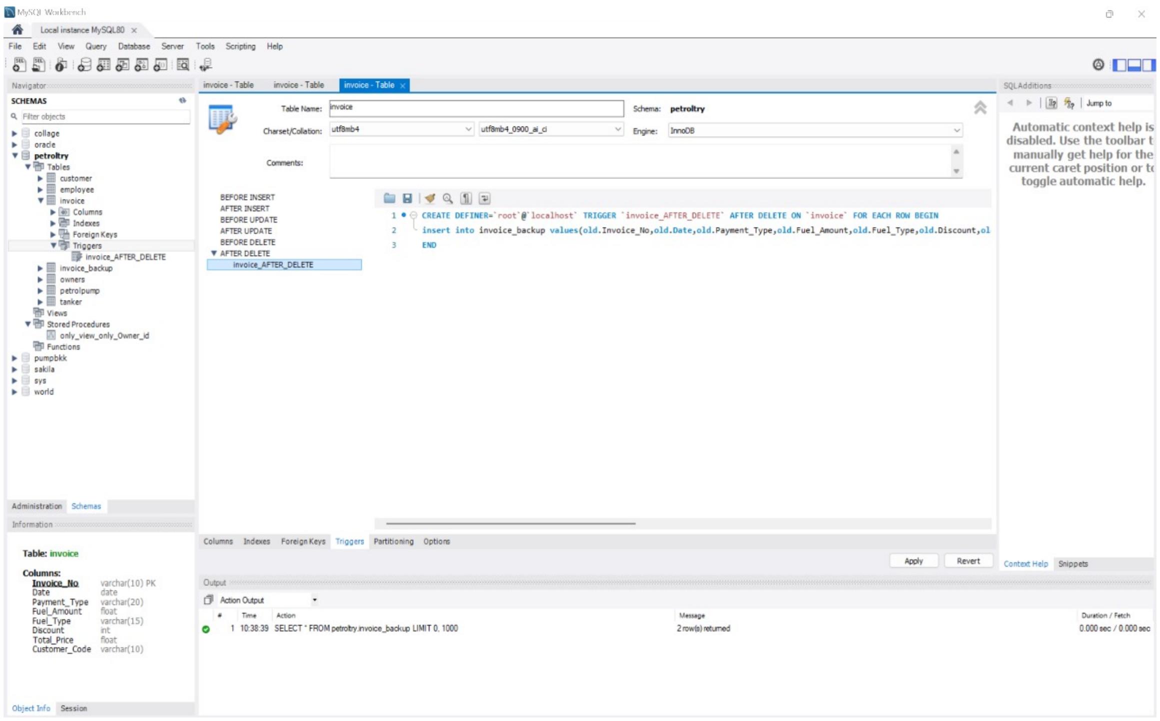


Table Name: invoice
Schema: petroltry
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `invoice_AFTER_DELETE` AFTER DELETE ON `invoice` FOR EACH ROW BEGIN
insert into invoice_backup values(old.Invoice_No,old.Date,old.Payment_Type,old.Fuel_Amount,old.Fuel_Type,old.Discount,ol
END
```

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

Table: invoice

Columns:

Invoice_No	varchar(10) PK
Date	date
Payment_Type	varchar(20)
Fuel_Amount	float
Fuel_Type	varchar(15)
Discount	int
Total_Price	float
Customer_Code	varchar(10)

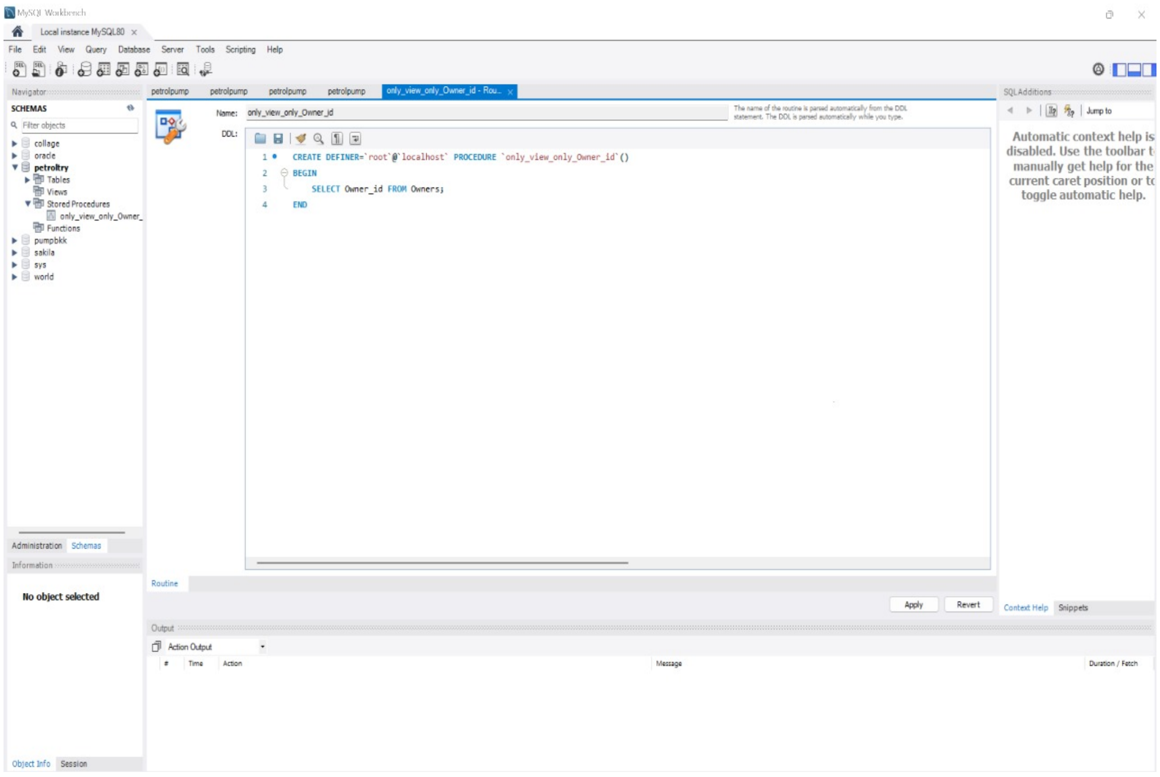
Output

Action Output

```
#  Time      Action                                        Message              Duration / Fetch
1  10:38:39  SELECT * FROM petroltry.invoice_backup LIMIT 0, 1000   2 row(s) returned   0.000 sec / 0.000 sec
```

Figure 7.11: Snippets for Trigger

Stored Procedure


Name: only_view_only_Owner_id

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `only_view_only_Owner_id`()
BEGIN
    SELECT Owner_id FROM Owners;
END
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

No object selected

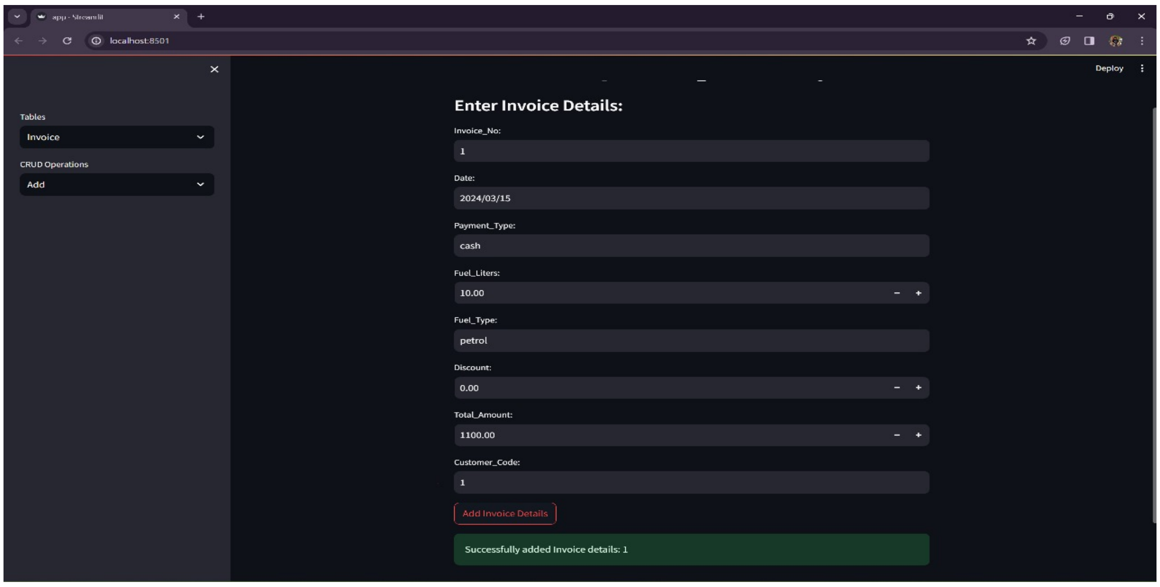
Figure 7.12: Snippets for Stored Procedure

Chapter 8

Results and Discussion

Bill Details:

This page allows Employee to generate invoice to customer.



Tables
Invoice

CRUD Operations
Add

Enter Invoice Details:

Invoice_No:
1

Date:
2024/03/15

Payment_Type:
cash

Fuel_Liters:
10.00

Fuel_Type:
petrol

Discount:
0.00

Total_Amount:
1100.00

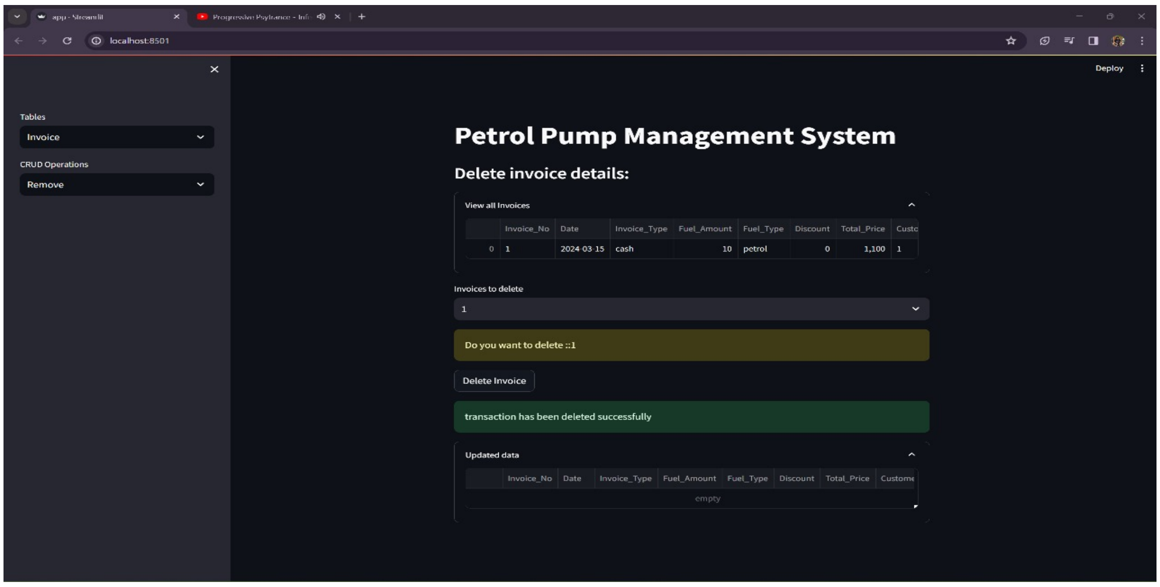
Customer_Code:
1

Successfully added Invoice details: 1

Figure 8.1: Invoice Details

Deleting bills:

Here Employee can delete bills of the customers by selecting that particular Invoice ID.



Petrol Pump Management System

Delete invoice details:

View all Invoices

	Invoice_No	Date	Invoice_Type	Fuel_Amount	Fuel_Type	Discount	Total_Price	Custo
0	1	2024-03-15	cash	10	petrol	0	1,100	1

Invoices to delete: 1

Do you want to delete ::1

Delete Invoice

transaction has been deleted successfully

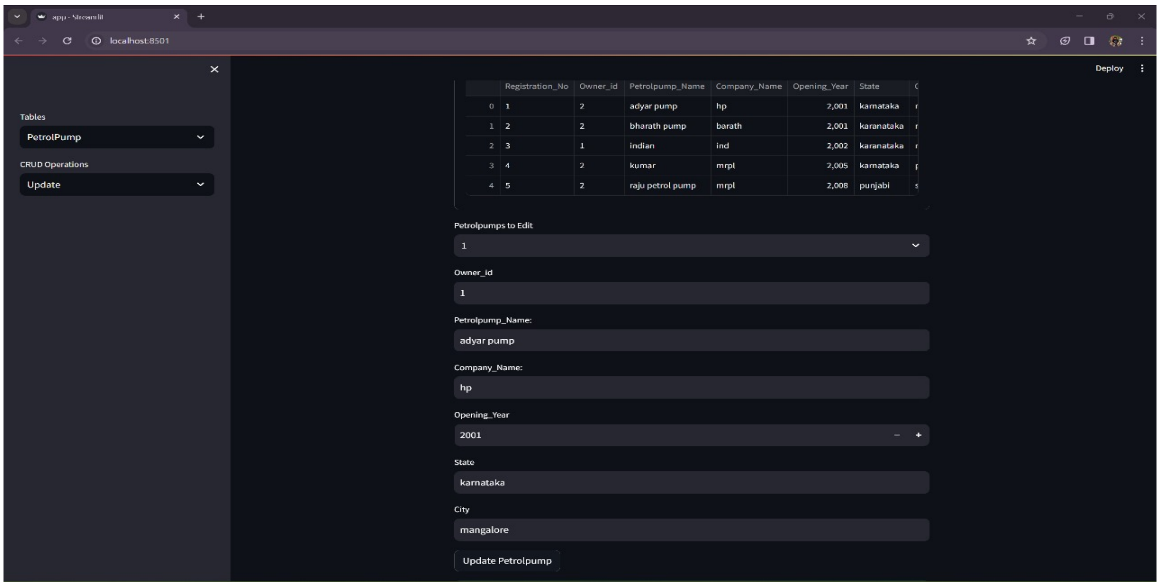
Updated data

	Invoice_No	Date	Invoice_Type	Fuel_Amount	Fuel_Type	Discount	Total_Price	Custome
				empty				

Figure 8.2: Deleting Invoice

Update Owner Details:

Here owner can update their details.



	Registration_No	Owner_id	Petrolpump_Name	Company_Name	Opening_Year	State
0	1	2	adyar pump	hp	2,001	kamataka
1	2	2	bharath pump	barath	2,001	karanataka
2	3	1	indian	ind	2,002	karanataka
3	4	2	kumar	mrpl	2,005	kamataka
4	5	2	raju petrol pump	mrpl	2,008	punjabi

Petrolpumps to Edit: 1

Owner_id: 1

Petrolpump_Name: adyar pump

Company_Name: hp

Opening_Year: 2001

State: karnataka

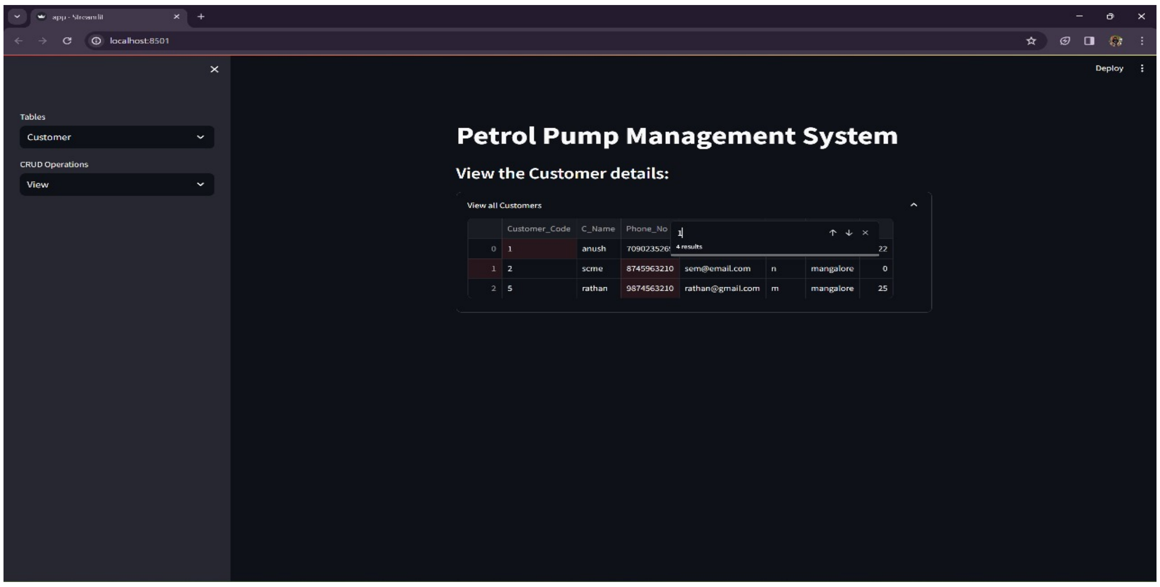
City: mangalore

Update Petrolpump

Figure 8.3: Updating Owner Details

Search Customer:

Here Employee can search for customer using customer id.



Petrol Pump Management System

View the Customer details:

View all Customers

1

4 results

	Customer_Code	C_Name	Phone_No				
0	1	anush	709023526				22
1	2	scme	8745963210	sem@email.com	n	mangalore	0
2	5	rathan	9874563210	rathan@gmail.com	m	mangalore	25

Figure 8.4: Searching Customer ID

Chapter 9

Conclusion and Future work

The Petrol Pump Management System developed will offers an efficient solution for managing various operations of a petrol pump. Through the implementation of a well-structured database and a user-friendly interface, the system facilitates tasks such as inventory management, sales tracking, employee management, and customer transactions. By automating these processes, the system enhances operational efficiency, reduces errors, and provides better control over the petrol pump's activities. In the future, there are several avenues for further development and enhancement of the Petrol Pump Management System. Integration with secure payment gateways could streamline transactions, ensuring convenience for customers and efficiency for the pump. Loyalty programs and mobile app development could enhance customer engagement and convenience, driving repeat business and satisfaction. Additionally, initiatives promoting environmental sustainability, such as the adoption of alternative fuels and energy-efficient practices, could position the petrol pump as a responsible corporate citizen. Continual iteration based on user feedback and technological advancements will be crucial to keep the system at the forefront of petrol pump management solutions, ensuring its effectiveness and relevance in the long term. This project can be further improved by keeping a track of the login and logout time of each cashier. Keeping track of when the cashier or the customer was registered or deleted, giving a warning when the product quantity goes low. Keeping a track of the sales that have happened on a particular day, Issuing gift voucher or discount coupons for the customers.

References

[1] Database systems Models, Languages, Design and Application Programming, Ramez Elmasri and Shamkant B. Navathe, 6th Edition, Pearson.

[2] Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.

[3] Areeg, A., Siddig, A., and Mohamed, A. (2017). Fuel management system. 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), Khartoum, Sudan

[4] Okemiri, A., and Nweso, E. (2016). Critical review of petrol station management system with emphasis on the advantages if digitalized in Nigeria. Journal of Multidisciplinary Engineering Science and Technology. 3(1),15-28.