GALALA UNIVERSITY
Powered by Arizona State University

Total benzene DB

CSE221: *Database systems*

Spring semester 2024

Project Team

Student ID	Student Name	Project Evaluation	
221101013	Youssef Gamal	Report	
		Presentation	
		Code	
		Total	
221101140	Nour Maher	Report	
		Presentation	
		Code	
		Total	
221100979	Yassin Walid	Report	
		Presentation	
		Code	
		Total	
222102535	Karim mamdouh ahmed	Report	
		Presentation	
		Code	
		Total	
221100039	Abdelrahman Tharwat Rashed Khalil	Report	
		Presentation	
		Code	
		Total	
221100128	Ahmed Mohamed Hassany	Report	
		Presentation	
		Code	
		Total	
223108628	Shrouq hesham Mahmoud	Report	
		Presentation	
		Code	
		Total	
221101027	Youssef wael hamdy ahmed	Report	
		Presentation	
		Code	
		Total	

Under supervision of Professor: Shaker El-Sappagh

GALALA UNIVERSITY
Powered by Arizona State University

Index

Abstract

The Total Benzene DB project is a comprehensive database management system designed to streamline operations at gas stations. This system aims to address the challenges of inconsistent data handling, manual processes, and inventory management inefficiencies. By automating key functions such as fuel tracking, expense management, and employee management, the system enhances data accuracy and operational efficiency. Additionally, the database provides real-time data access, aiding in better decision-making. This paper will cover the problem analysis, goals, system overview, implementation challenges, and future enhancements of the Total Benzene DB project.

Introduction

- In the fast-paced environment of gas stations, efficient data management is crucial for smooth operations and customer satisfaction.
- Traditional manual processes and inconsistent data handling can lead to significant operational challenges.

- **Project Description:**
  - The Total Benzene DB is a database management system specifically designed for gas stations to improve accuracy, efficiency, and decision-making.
  - The system integrates functionalities like fuel management, expense tracking, customer management, and detailed reporting.

Problem Analysis

- **Inconsistent Data Handling:**
  - Manual processes lead to data entry errors and inconsistencies.
- **Time-Consuming Manual Processes:**
  - Routine tasks like inventory tracking and expense recording are labor-intensive.
- **Inaccurate Inventory Tracking:**
  - Stockouts or overstocking due to poor tracking affect sales and customer satisfaction.
- **Lack of Real-Time Data Access:**
  - Delayed data updates hamper timely and effective decision-making.

Goals

- **Improve Data Accuracy and Accessibility:**
  - Develop a database system that reduces errors and allows for easy data retrieval.
- **Enhance Operational Efficiency:**
  - Automate routine tasks to save time and labor costs.
- **Support Better Decision-Making:**
  - Provide tools for real-time data analysis to aid strategic planning and operations.

Solution Overview

- The proposed solution is a robust database management system that integrates various functionalities required by a gas station.

- **Solution Advantages:**
  - **Data Accuracy:** Automated data entry reduces manual errors.
  - **Operational Efficiency:** Streamlined processes save time and reduce labor costs.
  - **Inventory Management:** Real-time tracking helps maintain optimal stock levels.
  - **Decision-Making:** Comprehensive reporting tools enable data-driven decisions.
  - **Statistics and data analysis**: Enhanced ability to analyze operational data for strategic insights.

Requirement Analysis

System Users:

Administrators: Manage overall system operations.

Employees: Enter transactions and manage daily operations.

Accountants: Access reports and analytics for decision-making.

Main Requirements:

- **Logs: System and login activity**
- **Prices: Edit gasoline prices**
- **Daily Shift: Track pump usage**
- **Oils: Manage oil inventory**
- **Expenses: Track and manage expenses**
- **Coupons: Manage and track gasoline coupons**
- **Deferred Customers: Manage customers who pay later**
- **Cash Register: Daily statistics and cash movements**
- **Employees: Manage employee data and roles**
- **Fuel types/ pumps: Diesel, 80, 92, 95**
- **System Users: admin, accountant, or employee**

System Overview

System Components:

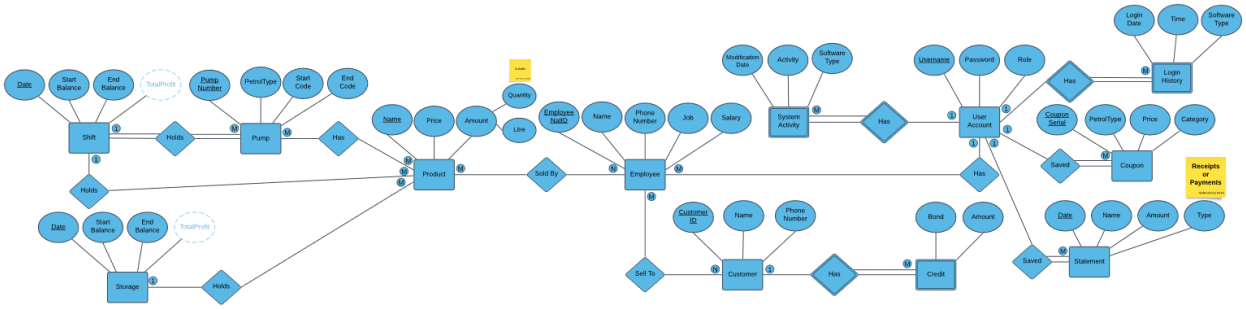
- **Desktop Software: GUI using python library pyqt5**
- **Database: Central repository for all data using MySQL**

System Functionalities & features:

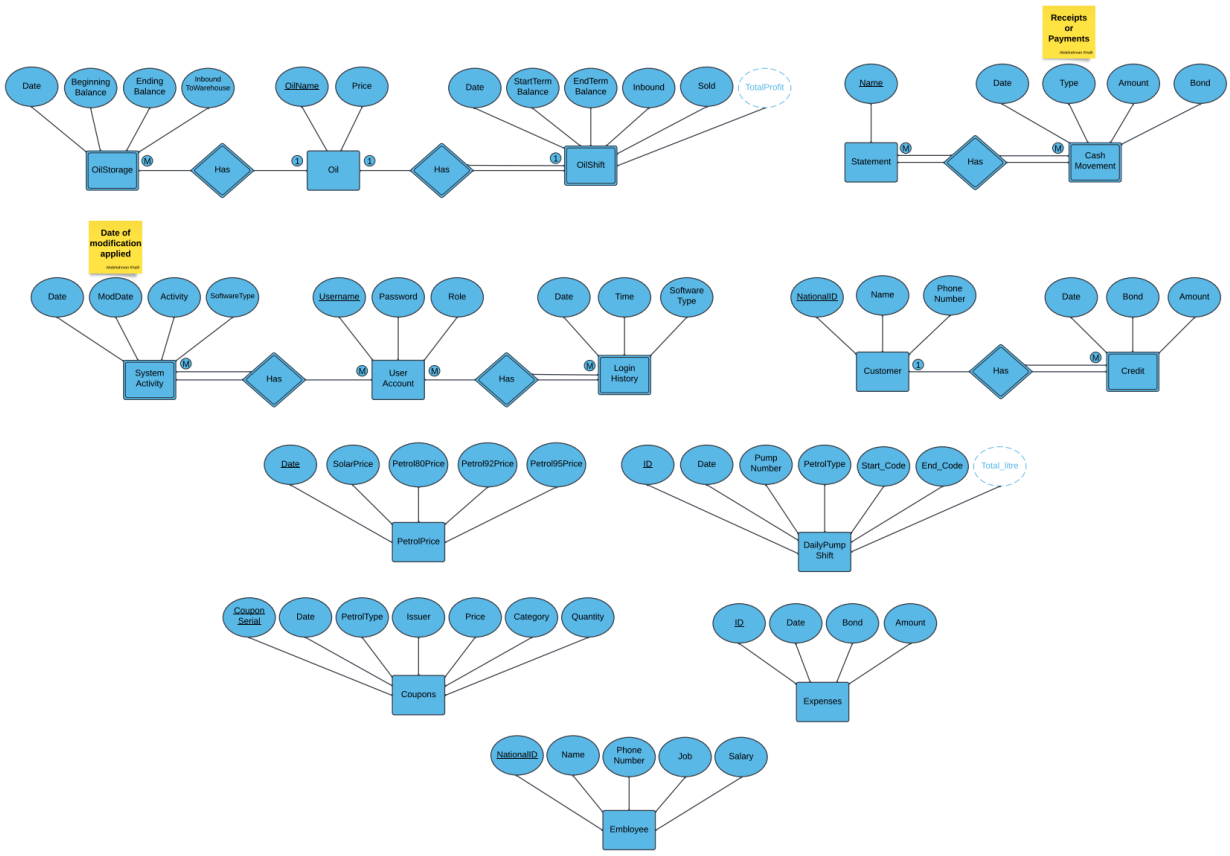
- **User Management: Add, delete, update users with roles (admin, accountant, employee)**
- **Fuel Management: Track fuel types, prices, daily shift codes, total liters**
- **Oil Management: Add, delete, edit oil data, track inventory**
- **Expenses Management: Record and track expenses with purpose and amount**
- **Coupons: Manage gasoline coupons, track matching statistics**
- **Deferred Customers: Handle customers who pay later, track transactions**
- **Cash Register: Daily statistics, cash movements, receipts, payments**
- **Employee Management: Manage employee data, salaries, roles**
- **Customer Management: Track customer details, purchase history**
- **Inventory Management: Monitor fuel levels, store items**
- **Sales and Transactions Monitoring: Record and analyze sales data in real time.**
- **Reporting and Analytics: Generate reports for sales, inventory, and employee performance.**

System Design

- **New ER Diagram:**

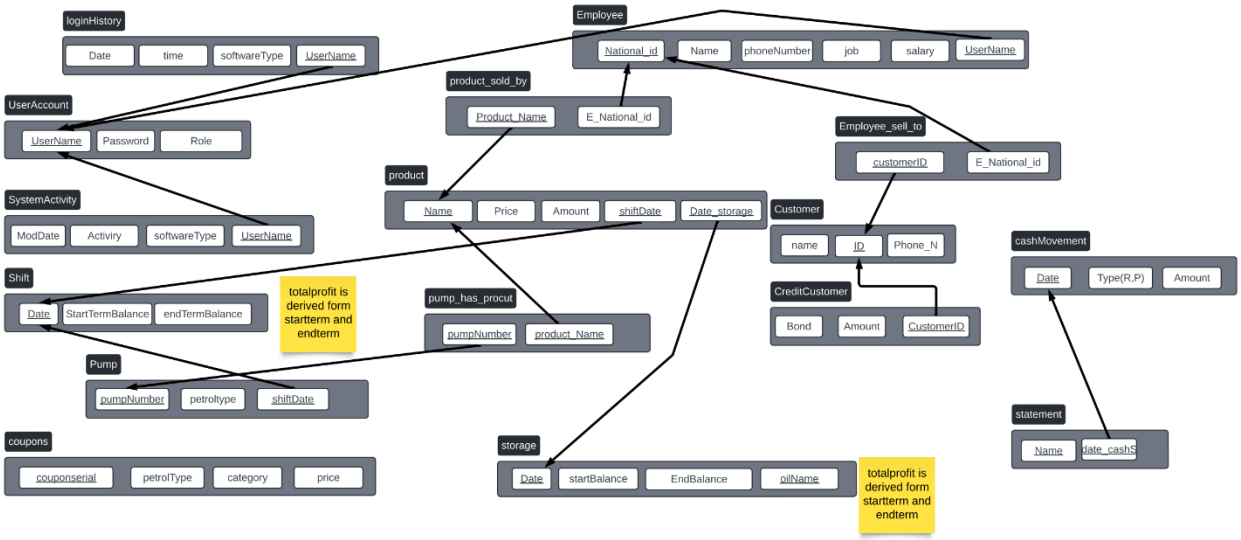


- **Old ER Diagram:**

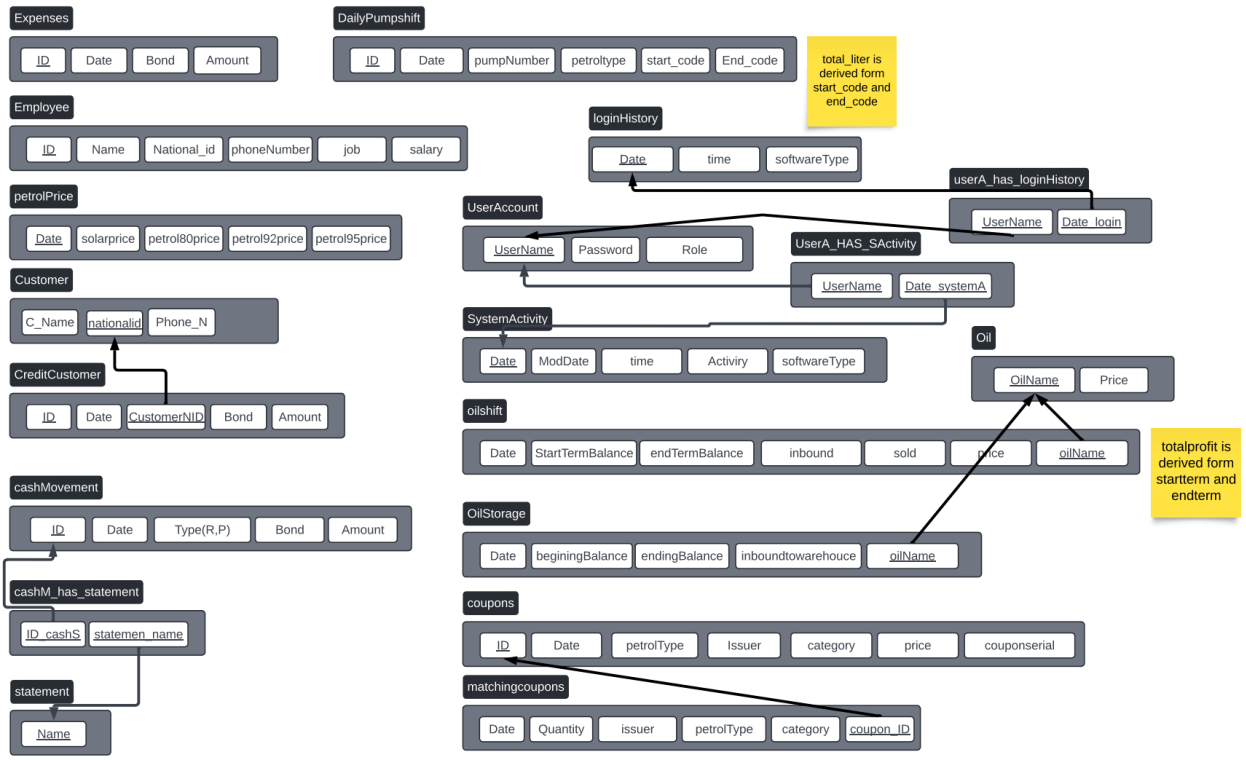


- **Database Schema**

New Relational model



Old Relational Model



Tables

1. **Table: petrolprice**
   - **date (TIMESTAMP NOT NULL)**
   - **solarprice (FLOAT NOT NULL)**
   - **petrol80price (FLOAT NOT NULL)**
   - **petrol92price (FLOAT NOT NULL)**
   - **petrol95price (FLOAT NOT NULL)**

2. **Table: dailyPumpshift**
   - **id (INT NOT NULL AUTO_INCREMENT)**
   - **date (TIMESTAMP NOT NULL)**
   - **pumpNumber (INT NOT NULL)**
   - **petrolType (VARCHAR(255) NOT NULL)**
   - **start_code (FLOAT NOT NULL)**
   - **end_code (FLOAT NOT NULL)**

3. **View: dailyPumpshift_liter**
   - **id**
   - **date**
   - **pumpNumber**
   - **petrolType**
   - **start_code**
   - **end_code**
   - **total_liter**

4. **Table: oil**
   - **oilName (VARCHAR(255) NOT NULL)**
   - **price (FLOAT)**

5. **Table: oilshift**
   - **date (TIMESTAMP NOT NULL)**
   - **StartTermBalance (FLOAT)**
   - **endTermBalance (FLOAT)**
   - **sold (INT NOT NULL)**
   - **price (FLOAT)**
   - **oilname (VARCHAR(255) NOT NULL)**

6. **View: totalprofit_view**
   - **StartTermBalance**
   - **endTermBalance**

- **inbound**
- **sold**
- **price**
- **oilname**
- **total_profit**

7. **Table: OilStorage**
   - **date (TIMESTAMP NOT NULL)**
   - **beginingBalance (FLOAT)**
   - **endingBalance (FLOAT)**
   - **inboundtowarehouse (VARCHAR(255))**
   - **oilname (VARCHAR(255) NOT NULL)**

8. **Table: expenses**
   - **id (INT NOT NULL AUTO_INCREMENT)**
   - **date (TIMESTAMP NOT NULL)**
   - **bond (VARCHAR(255) NOT NULL)**
   - **Amount (FLOAT NOT NULL)**

9. **Table: coupons**
   - **id (INT NOT NULL AUTO_INCREMENT)**
   - **date (TIMESTAMP NOT NULL)**
   - **petrolType (VARCHAR(255) NOT NULL)**
   - **Issuer (VARCHAR(255) NOT NULL)**
   - **category (INT NOT NULL)**
   - **price (FLOAT)**
   - **total (FLOAT)**
   - **couponserial (VARCHAR(255) NOT NULL)**

10. **Table: matchingcoupons**
    - **date (TIMESTAMP NOT NULL)**
    - **Quantity (INT NOT NULL)**
    - **petrolType (VARCHAR(255) NOT NULL)**
    - **issuer (VARCHAR(255) NOT NULL)**
    - **category (INT NOT NULL)**
    - **total (FLOAT)**

11. **Table: customer**
    - **name (VARCHAR(255) NOT NULL)**
    - **national_ID (VARCHAR(255) NOT NULL)**

- **phone_number (VARCHAR(255) NOT NULL)**

12. **Table: creditCustomer**
    - **id (INT NOT NULL AUTO_INCREMENT)**
    - **date (TIMESTAMP NOT NULL)**
    - **customer_NID (VARCHAR(255) NOT NULL)**
    - **bond (VARCHAR(255) NOT NULL)**
    - **Amount (FLOAT)**

13. **Table: statement**
    - **name (VARCHAR(255) NOT NULL)**
    - **Table: CashMovement**
    - **id (INT NOT NULL AUTO_INCREMENT)**
    - **type (ENUM('مقبوضات', 'مدفوعات') NOT NULL)**
    - **date (TIMESTAMP NOT NULL)**
    - **bond (VARCHAR(255) NOT NULL)**
    - **Amount (FLOAT)**

14. **Table: CM_has_S**
    - **id_cm (INT NOT NULL)**
    - **name_S (VARCHAR(255) NOT NULL)**

15. **Table: Employee**
    - **id (INT AUTO_INCREMENT)**
    - **name (VARCHAR(255) NOT NULL)**
    - **nationalID (VARCHAR(255) NOT NULL)**
    - **phoneNumber (VARCHAR(255) NOT NULL)**
    - **job (VARCHAR(255) NOT NULL)**
    - **salary (FLOAT NOT NULL)**

16. **Table: UserAccount**
    - **UserName (VARCHAR(255) NOT NULL)**
    - **Password (VARCHAR(255) NOT NULL)**
    - **Role (VARCHAR(255) NOT NULL)**

17. **Table: loginHistory**
    - **id (INT AUTO_INCREMENT)**
    - **date (TIMESTAMP NOT NULL)**

- **time (TIME NOT NULL)**
- **softwareType (VARCHAR(255) NOT NULL)**

18. **Table: userA_has_loginHistory**
    - **User_Name (VARCHAR(255) NOT NULL)**
    - **h_id (INT NOT NULL)**

19. **Table: SystemActivity**
    - **id (INT AUTO_INCREMENT)**
    - **date (TIMESTAMP NOT NULL)**
    - **ModDate (TIMESTAMP NOT NULL)**
    - **time (TIME NOT NULL)**
    - **Activity (VARCHAR(255) NOT NULL)**
    - **softwareType (VARCHAR(255) NOT NULL)**

20. **Table: UserA_Has_SActivity**
    - **UserName (VARCHAR(255) NOT NULL)**
    - **a_id (INT NOT NULL)**

Tables & views code

```
--
CREATE TABLE petrolprice(
   date TIMESTAMP NOT NULL,
   solarprice float NOT NULL,
   petrol80price float NOT NULL,
   petrol92price float NOT NULL,
   petrol95price float NOT NULL,
   CONSTRAINT PRIMARY KEY (date)

);

--
CREATE TABLE dailyPumpshift(
```

```
    id INT NOT NULL AUTO_INCREMENT,
    date TIMESTAMP NOT NULL,
    pumpNumber INT NOT NULL,
    petrolType Varchar(255) NOT NULL,
    start_code Float NOT NULL,
    end_code float NOt NULL,
    primary key (id)
);

--
CREATE VIEW dailyPumpshift_liter AS
SELECT
    id,
    date,
    pumpNumber,
    petrolType,
    start_code,
    end_code,
    (start_code - end_code) AS total_liter
FROM
    dailyPumpshift;

--
CREATE TABLE oil (
  oilName VARCHAR(255) NOT NULL,
  price float,
  primary key (oilName)
 );
```

```
--
CREATE TABLE oilshift(
   date TIMESTAMP NOT NULL,
   StartTermBalance float ,
   endTermBalance float ,
   sold INT NOT NULL,
   price float ,
   oilname VARCHAR(255) NOT NULL,
   FOREIGN KEY (oilname) REFERENCES oil(oilName)
);


--
CREATE VIEW totalprofit_view AS
SELECT
   StartTermBalance,
   endTermBalance,
   inbound,
   sold,
   price,
   oilname,
   (StartTermBalance - endTermBalance) AS total_profit
FROM
   oilshift;


--
CREATE TABLE OilStorage(
   date TIMESTAMP NOT NULL,
   beginingBalance float,
   endingBalance float,
```

```
    inboundtowarehouse VARCHAR(255),
    oilname VARCHAR(255) NOT NULL,
    FOREIGN KEY (oilname) REFERENCES oil(oilName)
);

--
CREATE TABLE expenses(
    id INT NOT NULL AUTO_INCREMENT,
    date TIMESTAMP NOT NULL,
    bond VARCHAR(255) NOT NULL,
    Amount float  NOT NULL,
    CONSTRAINT PRIMARY KEY (id)
);

--
CREATE TABLE coupons(
    id INT NOT NULL AUTO_INCREMENT,
    date TIMESTAMP NOT NULL,
    petrolType VARCHAR(255) NOT NULL,
    Issuer VARCHAR(255) NOT NULL,
    category INT NOT NULL,
    price float ,
    total float,
    couponserial VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
);

--
CREATE TABLE matchingcoupons(
```

```
    date TIMESTAMP NOT NULL,
    Quantity INT NOT NULL,
    petrolType VARCHAR(255) NOT NULL,
    issuer VARCHAR(255) NOT NULL,
    category INT NOT NULL,
            total float
);

--
CREATE TABLE customer(
    name varchar(255) not NULL,
    national_ID varchar(255) NOT NULL,
    phone_number varchar(255) NOT NULL,
    PRIMARY KEY (national_ID)
);

--
CREATE TABLE creditCustomer(
    id INT NOT NULL AUTO_INCREMENT,
    date TIMESTAMP NOT NULL,
    customer_NID varchar(255) NOT NULL,
    bond varchar(255) NOT NULL,
    Amount float,
    CONSTRAINT PRIMARY KEY (id),
    CONSTRAINT fk_customer_nid FOREIGN KEY (customer_NID) REFERENCES
customer(national_ID)
);

--
CREATE TABLE statement(
```

```
    name varchar(255) not null,
    Primary key(name)
);

--
CREATE TABLE CashMovement(
    id INT NOT NULL AUTO_INCREMENT,
    type ENUM('مدفوعات' ,'مقبوضات') NOT NULL,
    date TIMESTAMP NOT NULL,
    bond varchar(255) NOT NULL,
    Amount float,
    PRIMARY KEY (id)
);

--
CREATE TABLE CM_has_S(
    id_cm INT NOT NULL,
    name_S varchar(255) not null,
    FOREIGN KEY (id_cm) REFERENCES CashMovement(id),
    FOREIGN KEY (name_S) REFERENCES statement(name)
);

--
CREATE TABLE Employee(
    id INT AUTO_INCREMENT,
    name varchar(255) NOT NULL,
    nationalID varchar(255) NOT NULL,
    phoneNumber varchar(255) NOT NULL,
    job varchar(255) NOT NULL,
```

```
    salary Float not NULL,
    CONSTRAINT PRIMARY KEY (id)
);

-- start the id from 100
ALTER TABLE Employee AUTO_INCREMENT = 100;


--
CREATE TABLE UserAccount (
    UserName VARCHAR(255) NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Role VARCHAR(255) NOT NULL,
    PRIMARY KEY (UserName)
);


--
CREATE TABLE loginHistory(
        id INT AUTO_INCREMENT,
    date TIMESTAMP NOT NULL,
    time TIME NOT NULL,
    softwareType VARCHAR(255) NOT NULL,
        PRIMARY KEY (id)
);


--
CREATE TABLE userA_has_loginHistory (
    User_Name VARCHAR(255) NOT NULL,
    h_id INT NOT NULL,
    FOREIGN KEY (User_Name) REFERENCES UserAccount(UserName),
```

```
    FOREIGN KEY (h_id) REFERENCES loginHistory(id)
);

--
CREATE TABLE SystemActivity (
        id INT AUTO_INCREMENT,
    date TIMESTAMP NOT NULL,
    ModDate TIMESTAMP NOT NULL,
    time TIME NOT NULL,
    Activity VARCHAR(255) NOT NULL,
    softwareType VARCHAR(255) NOT NULL,
        PRIMARY KEY (id)
);

--
CREATE TABLE UserA_Has_SActivity (
    UserName VARCHAR(255) NOT NULL,
    a_id INT NOT NULL,
    FOREIGN KEY (UserName) REFERENCES UserAccount(UserName),
    FOREIGN KEY (a_id) REFERENCES SystemActivity(id)

);
```

GUI

functionality:

- Provides an intuitive interface for easy navigation, reducing the learning curve and enabling quick proficiency.

- Automates routine tasks like data entry, calculations, and record-keeping, saving time and minimizing errors.

- Ensures real-time data updates, providing users with the latest information for managing fuel sales, inventory, and cash movements.

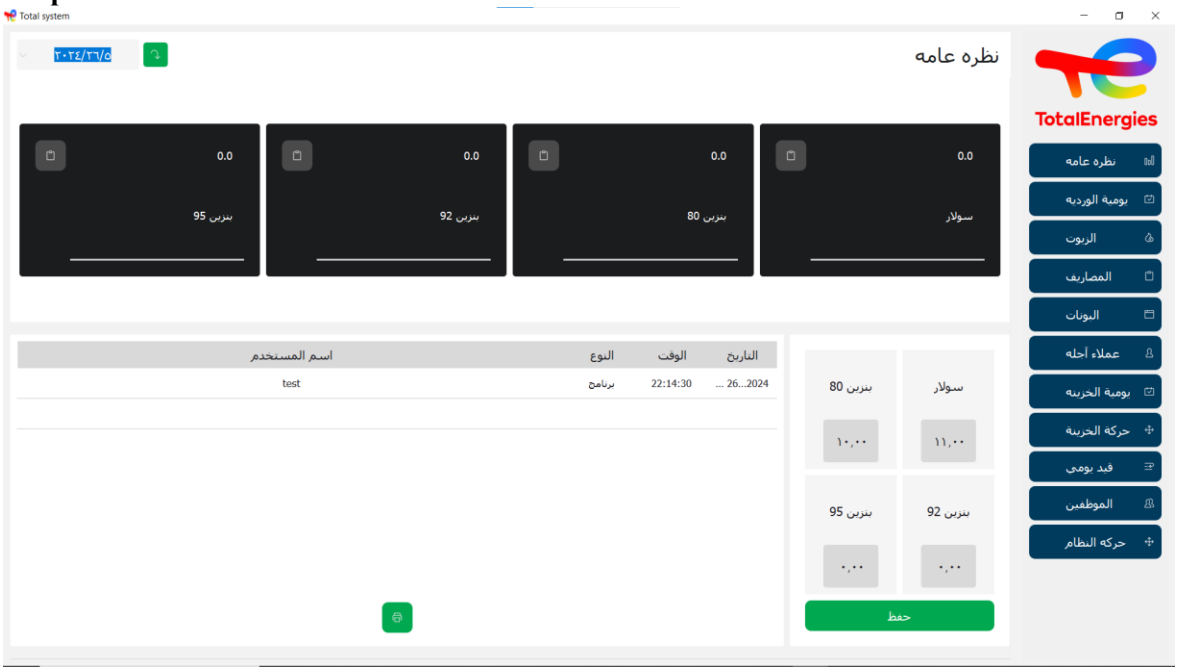
- Maintains high levels of data accuracy with structured forms, validation checks, and error messages, ensuring reliable information.

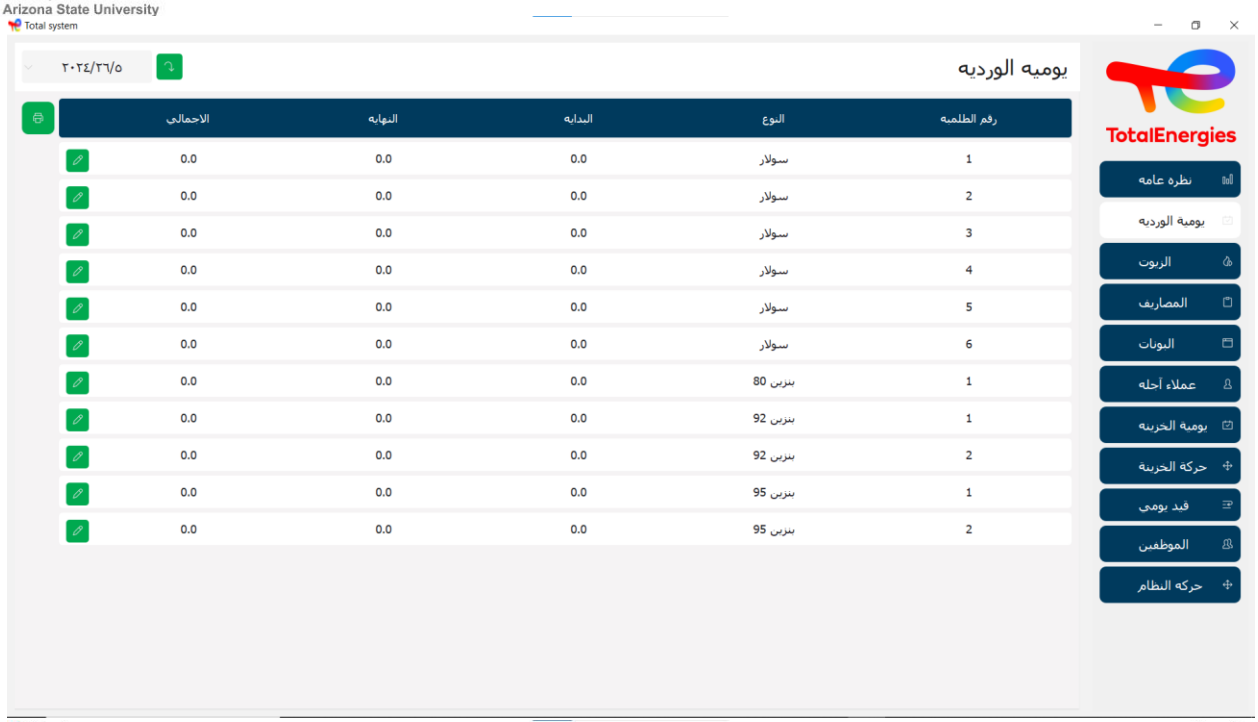
- Offers a visually appealing and organized layout, improving the overall user experience and making tasks easier to perform.

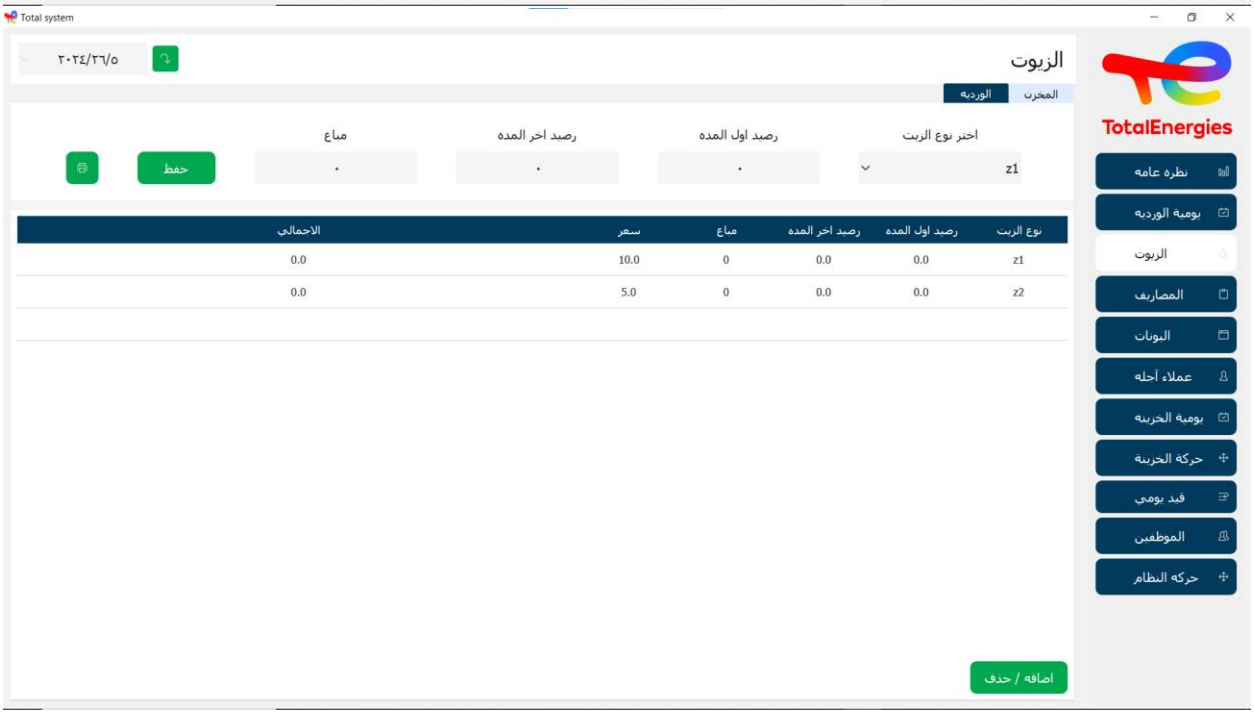
- Generates detailed reports and dashboards, helping managers monitor key performance indicators (KPIs), track financial metrics, and identify trends.

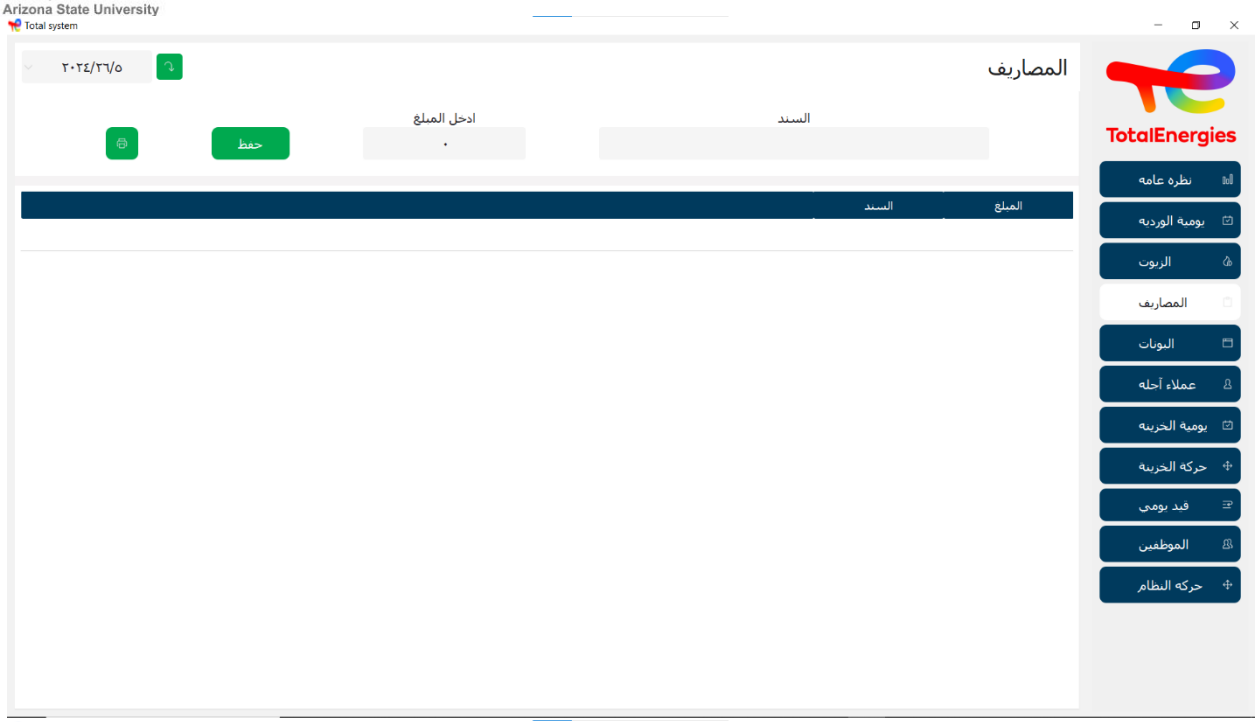
- Enhances security through user authentication and role-based access control, ensuring that only authorized personnel can access certain features and data.

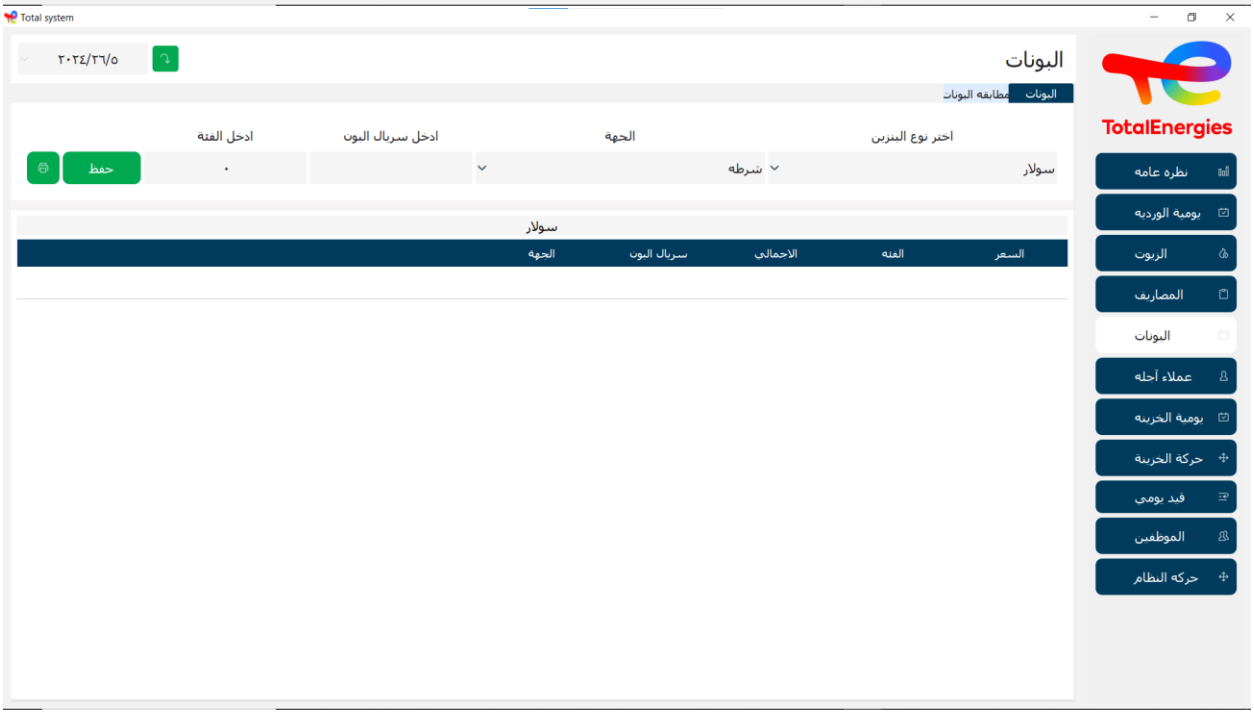
Snapshots:

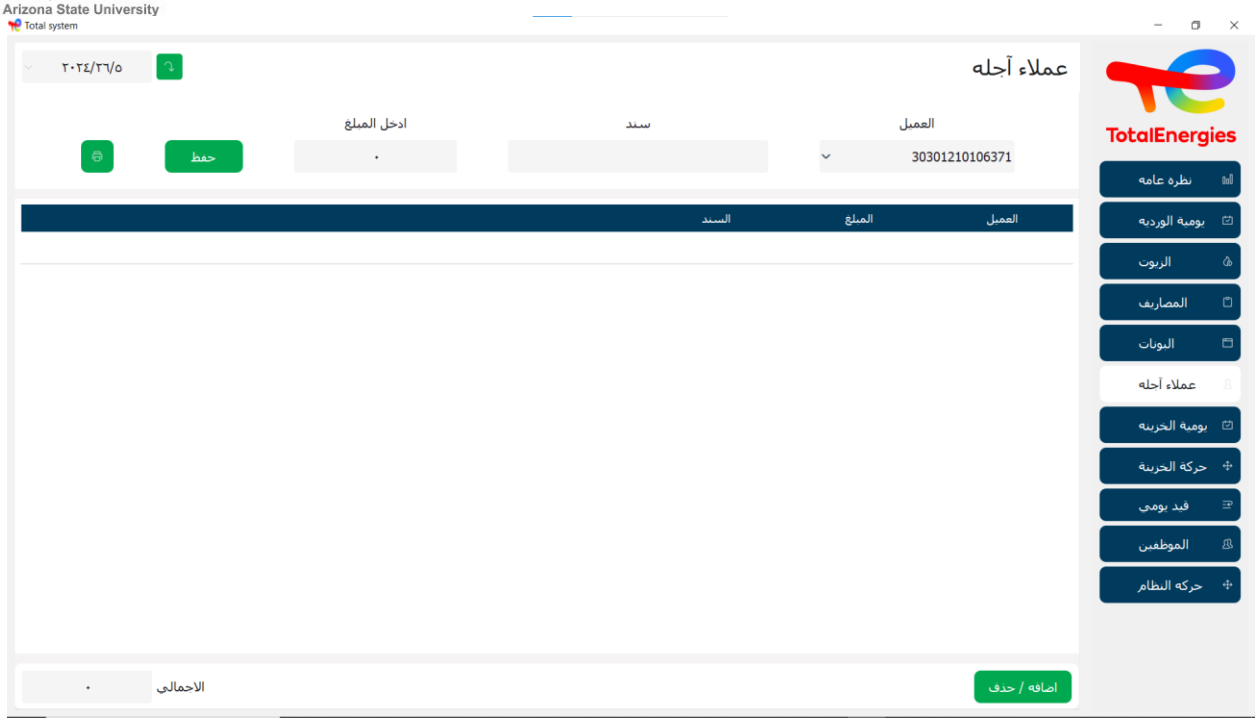



Total system
يوميه الورديه
٢٠٢٤/٢٦/٥
رقم الطلمبه
النوع
البدايه
النهايه
الاجمالي
1 سولار 0.0 0.0 0.0
2 سولار 0.0 0.0 0.0
3 سولار 0.0 0.0 0.0
4 سولار 0.0 0.0 0.0
5 سولار 0.0 0.0 0.0
6 سولار 0.0 0.0 0.0
1 بنزين 80 0.0 0.0 0.0
1 بنزين 92 0.0 0.0 0.0
2 بنزين 92 0.0 0.0 0.0
1 بنزين 95 0.0 0.0 0.0
2 بنزين 95 0.0 0.0 0.0
TotalEnergies
نظره عامه
يومية الورديه
الزيوت
المصاريف
البونات
عملاء أجله
يومية الخزينه
حركة الخزينة
قيد يومي
الموظفين
حركه النظام


Total system
الزيوت
٢٠٢٤/٢٦/٥
المخزن
الورديه
اختر نوع الزيت
z1
رصيد اول المده
رصيد اخر المده
مباع
حفظ
نوع الزيت
رصيد اول المده
رصيد اخر المده
مباع
سعر
الاجمالي
z1 0.0 0.0 0 10.0 0.0
z2 0.0 0.0 0 5.0 0.0
اضافه / حذف
TotalEnergies
نظره عامه
يومية الورديه
الزيوت
المصاريف
البونات
عملاء أجله
يومية الخزينه
حركة الخزينة
قيد يومي
الموظفين
حركه النظام


Total system
المصاريف
٢٠٢٤/٢٦/٥
السند
ادخل المبلغ
٠
حفظ
المبلغ
السند
TotalEnergies
نظره عامه
يومية الورديه
الربوت
المصاريف
البونات
عملاء آجله
يومية الخزينه
حركه الخزينه
قيد يومي
الموظفين
حركه النظام


Total system
البونات
٢٠٢٤/٢٦/٥
البونات
مطابقه البونات
اختر نوع البنزين
سولار
الجهة
شرطه
ادخل سريال البون
ادخل الفئة
٠
حفظ
سولار
السعر
الفئه
الاجمالي
سريال البون
الجهه
TotalEnergies
نظره عامه
يومية الورديه
الربوت
المصاريف
البونات
عملاء آجله
يومية الخزينه
حركه الخزينه
قيد يومي
الموظفين
حركه النظام



AI model using GEMINI LLM

Integration of AI Using Gemini:

- Objective:

- To enhance the functionality of the Total Benzene DB system by integrating artificial intelligence capabilities while maintaining data privacy.

- Approach:

- The Gemini Language Learning Model (LLM) is integrated into the system to assist with database queries.

- To ensure data privacy, the Gemini LLM does not directly interact with the database.

- Process:

1. Query Creation:

- The Gemini LLM takes input from the user and generates a corresponding SQL query.

2. Query Extraction:

- A specialized function extracts the query from the user input, ensuring that the Gemini LLM does not directly access the database.

3. Query Execution:

- Another function automatically executes the extracted SQL query against the database.

4. Output Handling:

- The output of the executed query is processed by a function that formats the results into tables.

5. Output Presentation:

- The formatted tables are displayed in a dedicated graphical user interface (GUI), allowing users to view the results in a user-friendly manner.

- Benefits:

- Enhanced Usability: Users can interact with the system using natural language inputs, simplifying the query process.

- Data Privacy: By isolating the Gemini LLM from direct database interactions, data privacy is maintained.

- Efficiency: Automated query generation and execution streamline data retrieval processes.

- User-Friendly Output: Results are presented in a clear and organized manner through a dedicated GUI.

This integration showcases the potential of combining artificial intelligence with database management to create a more efficient and user-friendly system while holding data privacy standards.

Implementation

- **Development Process:**
  - **Planning:**
    - Gather requirements and design the system architecture.
  - **Development:**
    - Code the database schema and create the GUI.
  - **Testing:**

- Conduct unit tests, integration tests, and user acceptance tests to ensure system functionality.

- **Deployment and Maintenance:**
    - **Deployment:**
        - Deploy the backend on a server and distribute the desktop application.
    - **Maintenance:**
        - Plan for regular updates, bug fixes, and monitor system performance and user feedback.

- **Implementation Challenges:**
    - **Challenge:**
        - Designing the Entity-Relationship Diagram (ERD) for a gas station, given its complexity and the presence of several non-intuitive concepts.
    - **Solution:**
        - Thorough research and iterative design were utilized to understand the various concepts involved in the gas station's operations. By breaking down the problem into smaller, manageable parts and continuously refining the design, a robust ERD was eventually developed.

- **Unresolved Challenges:**
    - **Issue:**
        - Limited mobile compatibility was identified as a challenge during the development of the Total Benzene DB system. The initial design and implementation were primarily optimized for desktop usage, resulting in limited accessibility and usability on mobile devices.
    - **Plan:**
        - To address this challenge, a strategic plan for future updates has been formulated. The goal is to enhance the system's compatibility and usability on mobile platforms by incorporating a mobile-friendly interface.

Experience Obtained

- **Skills and Knowledge Gained:**
    - **Technical Skills:**
        - Enhanced understanding of database design, SQL programming, and system integration.
    - **Project Management:**
        - Improved ability to manage project timelines and deliverables.
- **Lessons Learned:**
    - Recognized the importance of thorough requirement analysis and user feedback in system design.
    - Gained experience in troubleshooting and solving real-world implementation challenges.

Future Enhancements

- **Potential Improvements:**
    - **Mobile Compatibility:**
        - Develop a mobile-friendly interface to improve accessibility for users on the go.
    - **Advanced Analytics:**
        - Implement machine learning algorithms for predictive analytics to forecast demand and optimize operations.
    - **Scalability:**
        - Optimize the system to support larger gas station chains, including additional services like restaurants and car maintenance.
    - **User Feedback:**
        - Continuously gather and incorporate feedback to refine and improve the system's features and functionalities.

Conclusion

- **Summary:**
    - The Total Benzene DB system significantly enhances data accuracy, operational efficiency, and decision-making capabilities for gas stations.
    - By automating key processes and providing real-time data access, the system addresses many of the operational challenges faced by gas stations.

References

Smith, John. "Database Management Systems: Principles and Practice." New York: Pearson, 2020.

Garcia, Maria. "Introduction to SQL: Mastering the Relational Database Language." Boston: Addison-Wesley, 2019.

Johnson, David. "Python GUI Programming Cookbook: Develop functional and responsive user interfaces with tkinter and PyQt5." Birmingham: Packt Publishing, 2021.

Brown, Emily. "MySQL for Beginners: A Comprehensive Guide to MySQL Database Programming." San Francisco: O'Reilly Media, 2018.

Patel, Ravi. "Artificial Intelligence: Foundations and Applications." Cambridge: Cambridge University Press, 2022.