

PES UNIVERSITY

Department of Computer Science and Engineering

Course Code: UE23CS351A

Mini Project Report on

Fuel Management

System

Submitted by:

Ritesh Reddy R S (PES1UG23CS482)

Section H

Under the guidance of:

Shruthi B P

Academic Year: 2025

ABSTRACT

A web-based database management system designed to automate fuel station operations, integrating real-time inventory, transaction management, and employee oversight for operational excellence and data integrity.

System Overview & Architecture

The Fuel Management System utilizes a robust client-server architecture with a relational database. It features a user-friendly web front-end and a back-end for data processing, business logic, and secure database communication, ensuring scalability and efficient real-time operations.

Purpose Automate and streamline all core fuel station operations.	Scope Real-time inventory, transaction processing, employee management, and comprehensive reporting.	Automation Reduces errors, improves efficiency, and provides instant data access.
-----------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------

Core Entities

1	2	3
Fuel Types, pricing, and current stock levels managed.	Transactions Records of sales, purchases, and deliveries.	Employees Roles, access controls, and performance tracking.
4	5	
Customers Loyalty programs and purchase history management.	Suppliers Procurement and delivery schedule management.	

Technology Stack

- **Backend:** Flask (Python)
- **Database:** Microsoft SQL Server
- **Connectivity:** ODBC Driver 17
- **Frontend:** HTML, CSS
- **Version Control:** GitHub
- **Query Language:** T-SQL

User Requirement Specification

Purpose

Automates essential fuel station operations (inventory, transactions, employee management), ensuring real-time updates, transparency, and data integrity via a relational database.

Scope

Supports fuel type, automated restocking, employee/pump management, and analytical reporting. Admin/employees interact via a Flask web interface with SQL Server.

Detailed Description

A Flask web application integrating with MS SQL Server (ODBC). Utilizes CRUD, stored procedures, triggers, and advanced queries for workflow automation. Features dashboards for inventory, transactions, and financial metrics.

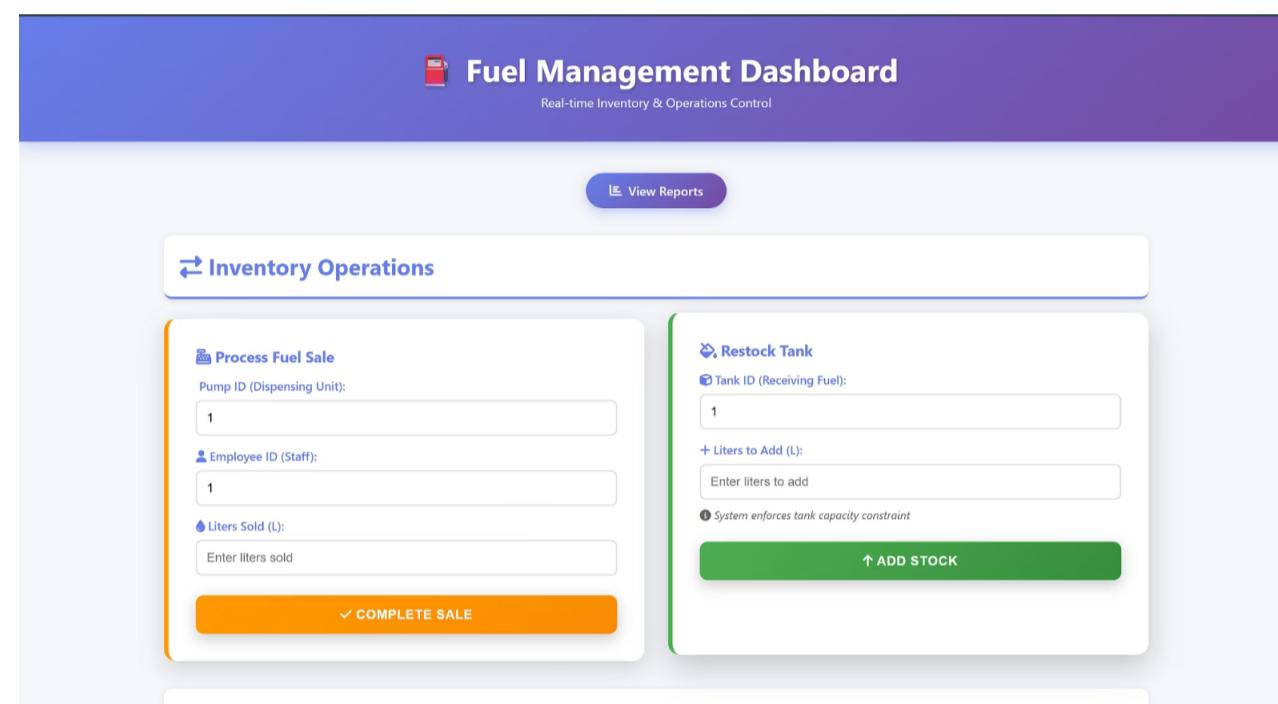
Functional Requirements

Functionality	Description
1. Manage Fuel Types	Add, edit, remove fuel types and prices.
2. Employee Management	Create, update, delete employee records with roles.
3. Transaction Recording	Log sales: employee, fuel, quantity, cost.
4. Restock Operations	Automated fuel tank restocking with overflow checks.
5. Dashboard Visualization	Display real-time tank levels and fuel prices.
6. Complex Queries	Generate join, nested, and aggregate analytical reports.
7. Error Handling	Validate inputs and rollback transactions on errors.

User Interface - Web Application

Dashboard Overview

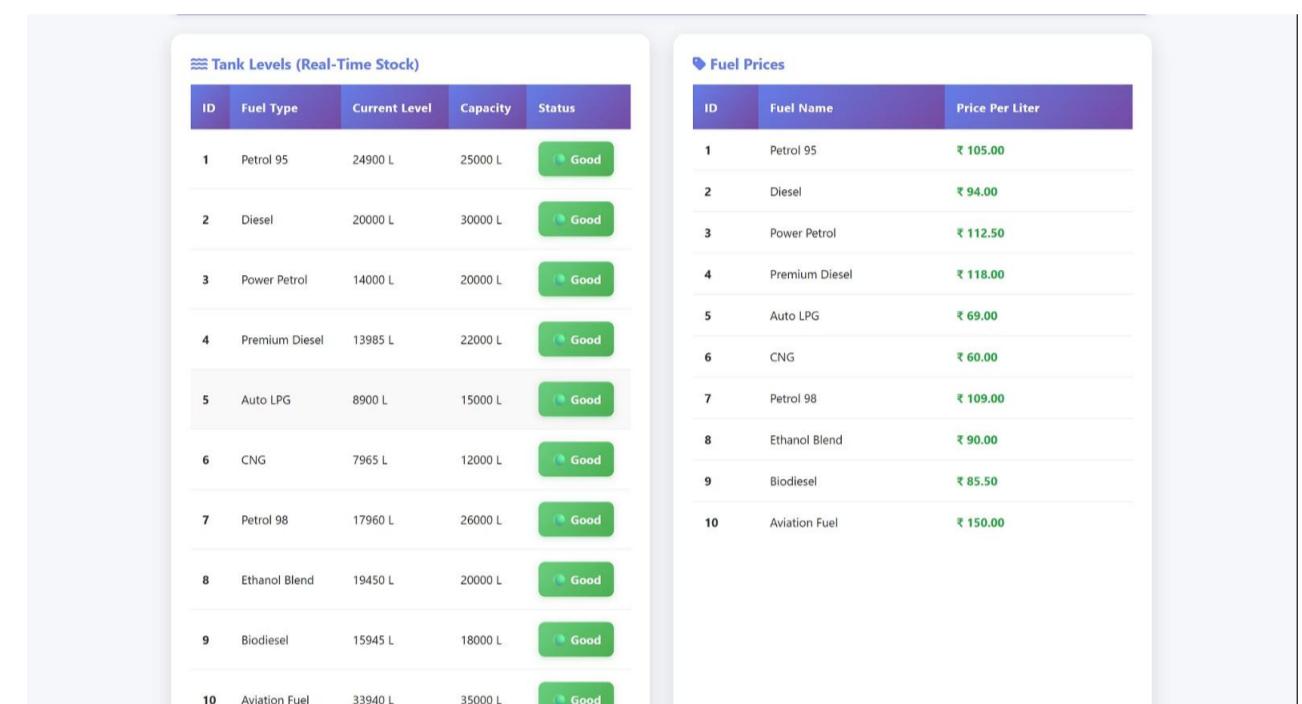
This screenshot displays the main dashboard of the Fuel Management System, offering a quick glance at key operational metrics. It typically includes summaries of fuel levels, recent transactions, and system alerts for efficient monitoring.



The dashboard features a top navigation bar with 'Fuel Management Dashboard' and 'Real-time Inventory & Operations Control'. Below is a 'View Reports' button. The main area has two sections: 'Inventory Operations' and 'Fuel Price Management'. 'Inventory Operations' includes 'Process Fuel Sale' (with fields for Pump ID, Employee ID, and Liters Sold) and 'Restock Tank' (with fields for Tank ID and Liters to Add). A 'COMPLETE SALE' button is also present. The 'Fuel Price Management' section shows a table of current fuel prices.

Fuel Price Management

This interface allows administrators to manage and update fuel types and their corresponding prices. It provides fields for entering new prices, adjusting existing ones, and viewing historical pricing data to ensure accuracy and timely updates.

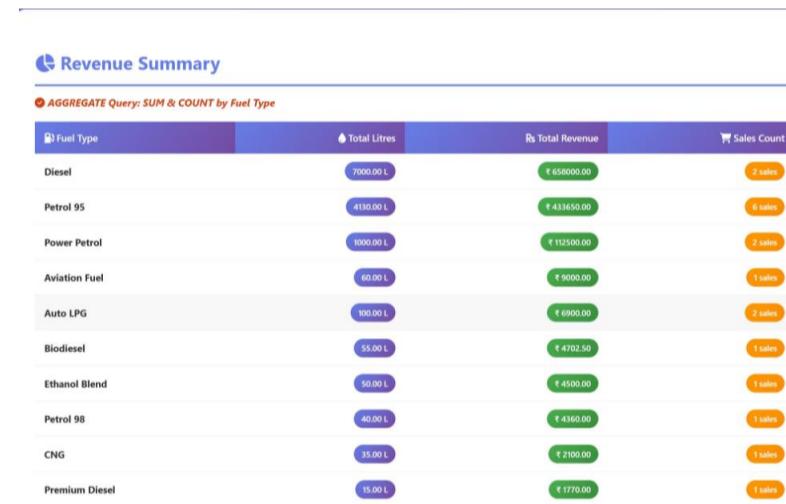


The interface shows two tables. The left table, 'Tank Levels (Real-Time Stock)', lists tank IDs, fuel types, current levels, capacities, and statuses. The right table, 'Fuel Prices', lists fuel names and their respective prices per liter.

ID	Fuel Type	Current Level	Capacity	Status
1	Petrol 95	24900 L	25000 L	Good
2	Diesel	20000 L	30000 L	Good
3	Power Petrol	14000 L	20000 L	Good
4	Premium Diesel	13985 L	22000 L	Good
5	Auto LPG	8900 L	15000 L	Good
6	CNG	7965 L	12000 L	Good
7	Petrol 98	17960 L	26000 L	Good
8	Ethanol Blend	19450 L	20000 L	Good
9	Biodiesel	15945 L	18000 L	Good
10	Aviation Fuel	33940 L	35000 L	Good

ID	Fuel Name	Price Per Liter
1	Petrol 95	₹ 105.00
2	Diesel	₹ 94.00
3	Power Petrol	₹ 112.50
4	Premium Diesel	₹ 118.00
5	Auto LPG	₹ 69.00
6	CNG	₹ 60.00
7	Petrol 98	₹ 109.00
8	Ethanol Blend	₹ 90.00
9	Biodiesel	₹ 85.50
10	Aviation Fuel	₹ 150.00

AGGREGATE QUERY

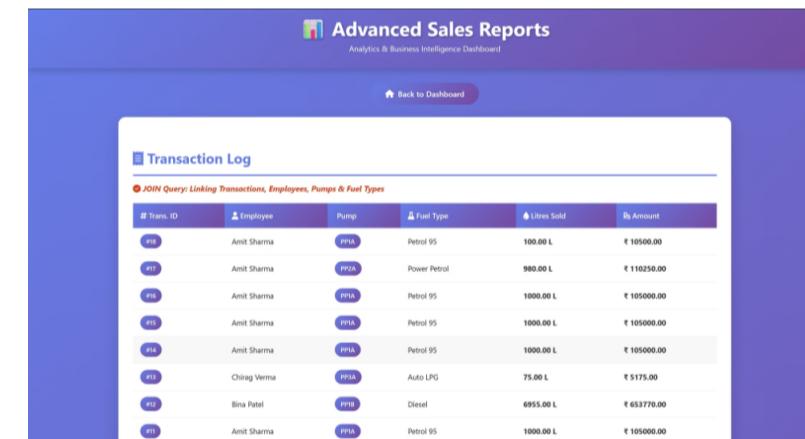


The 'Revenue Summary' section shows an aggregate query for fuel type. It displays total litres, total revenue, and sales count for various fuel types like Diesel, Petrol 95, Power Petrol, etc.

Fuel Type	Total Litres	Total Revenue	Sales Count
Diesel	1000.00 L	₹ 60000.00	10 Sales
Petrol 95	4500.00 L	₹ 450000.00	5 Sales
Power Petrol	1000.00 L	₹ 10000.00	2 Sales
Aviation Fuel	50.00 L	₹ 5000.00	1 Sales
Auto LPG	300.00 L	₹ 6000.00	2 Sales
Biodiesel	55.00 L	₹ 4750.00	1 Sales
Ethanol Blend	50.00 L	₹ 4000.00	1 Sales
Petrol 98	40.00 L	₹ 4000.00	1 Sales
CNG	25.00 L	₹ 2500.00	1 Sales
Premium Diesel	15.00 L	₹ 1770.00	1 Sales

This screen provides a detailed log of all fueling transactions, including fuel type, quantity, date, time, and associated vehicle or driver. It allows for auditing and historical analysis of fuel consumption.

JOIN QUERY

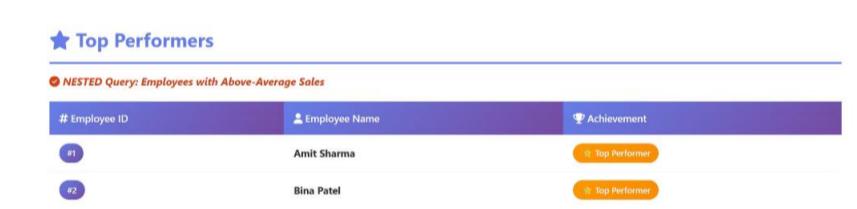


The 'Advanced Sales Reports' section shows a join query linking transactions, employees, pump & fuel types. It displays a transaction log with details like transaction ID, employee name, fuel type, liters sold, and amount.

Transaction ID	Employee	Fuel Type	Liters Sold	Amount
T1	Amit Sharma	Petrol 95	100.00 L	₹ 1000.00
T2	Amit Sharma	Power Petrol	900.00 L	₹ 110250.00
T3	Amit Sharma	Petrol 95	1000.00 L	₹ 100000.00
T4	Amit Sharma	Petrol 95	1000.00 L	₹ 105000.00
T5	Amit Sharma	Auto LPG	75.00 L	₹ 1575.00
T6	Chirag Verma	Diesel	6955.00 L	₹ 651779.00
T7	Bina Patel	Petrol 95	1000.00 L	₹ 105000.00

Manage your entire fleet with ease. This interface allows you to add, edit, or remove vehicles, track their status, and assign them to specific drivers or departments, ensuring efficient resource allocation.

NESTED QUERY



The 'Top Performers' section shows a nested query for employees with above-average sales. It lists employee IDs, names, and achievement levels.

Employee ID	Employee Name	Achievement
E1	Amit Sharma	Top Performer
E2	Bina Patel	Top Performer

This section is dedicated to managing driver profiles, including their credentials, assigned vehicles, and access permissions. It helps maintain accountability and streamline the fueling process for personnel.

CRUD Operations

This section demonstrates the fundamental Create and Read operations (CRUD) for managing employee data within the system.

Create Operation

```
INSERT INTO EMPLOYEES (Name, Role)
VALUES ('Rahul Mehta', 'Pump
Operator');
```

→ Adds a new employee record successfully.

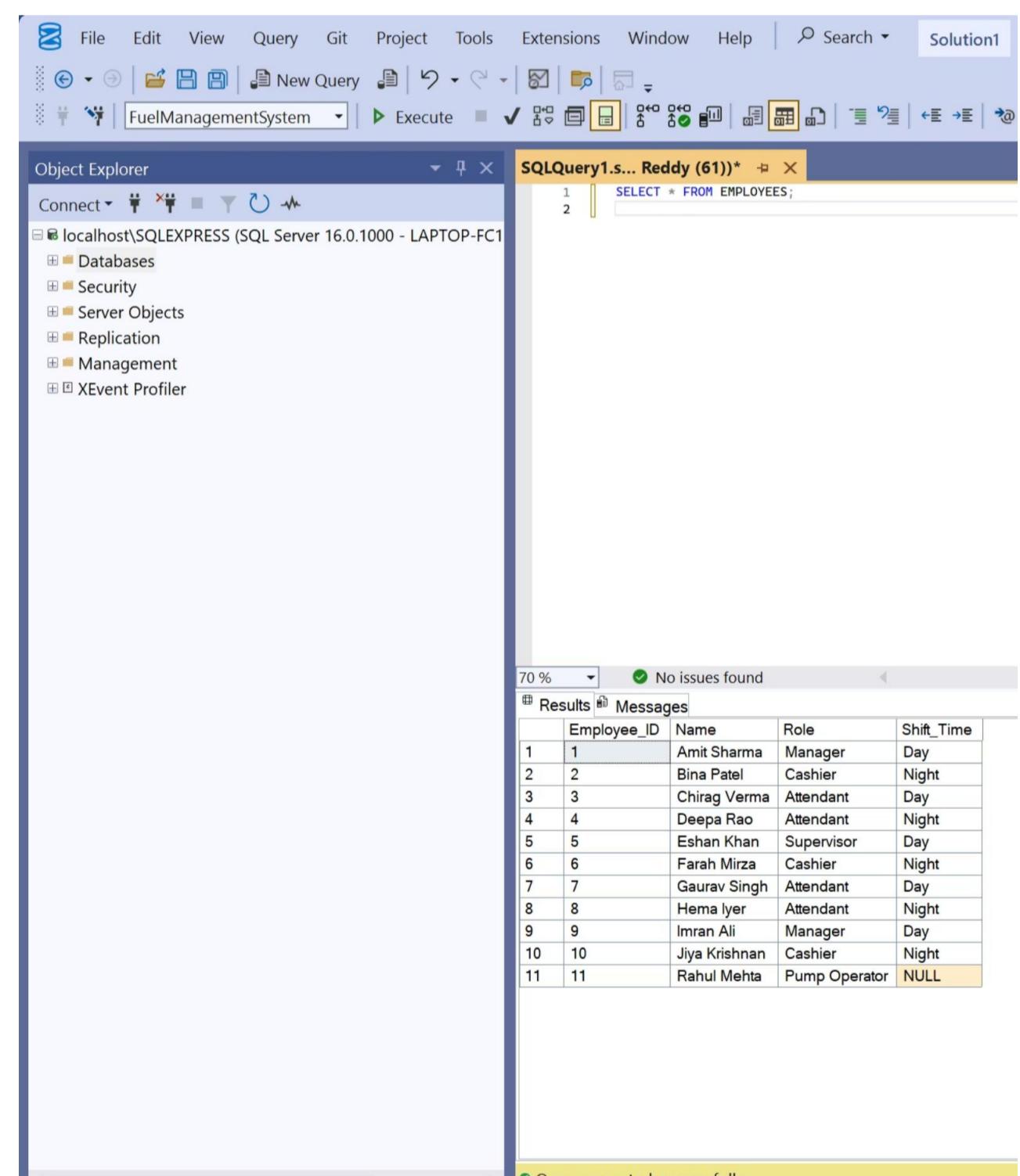
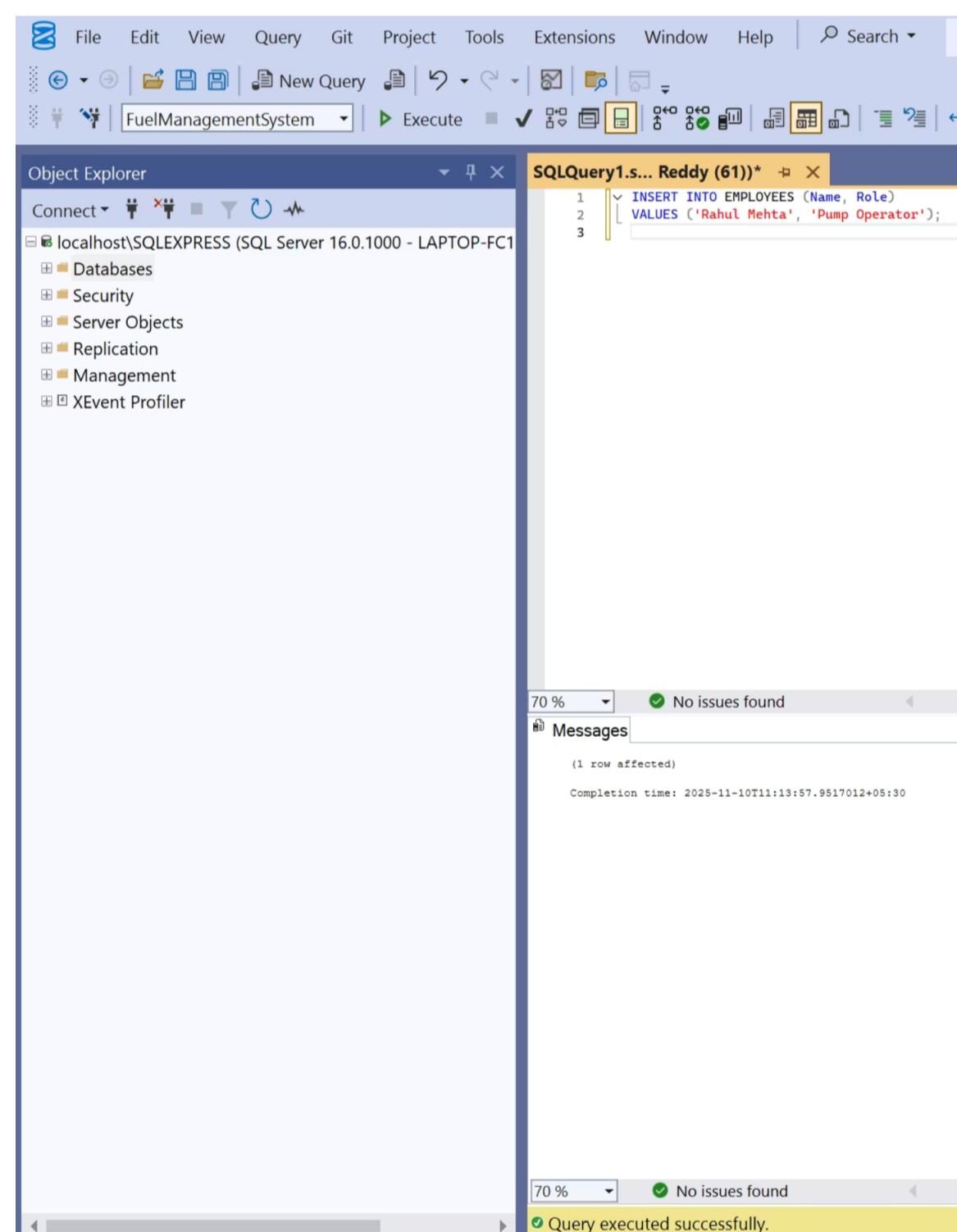
Query executed successfully

Read Operation

```
SELECT * FROM EMPLOYEES;
```

→ Displays all employee records including the new entry.

Query executed successfully



CRUD Operations

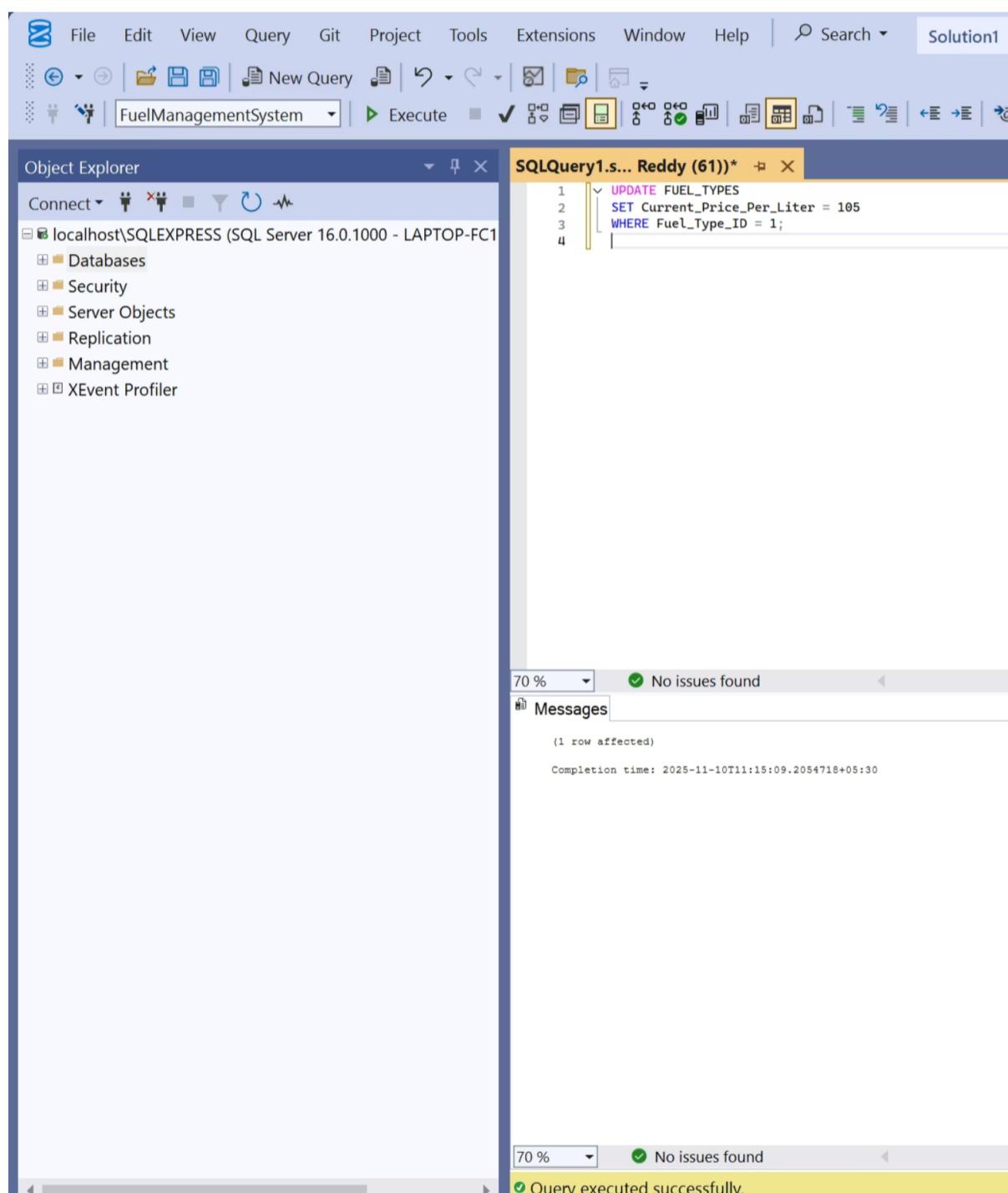
This section continues our exploration of CRUD operations, focusing on how data within the system is updated and deleted to maintain accuracy and relevance.

Update Operation

```
UPDATE FUEL_TYPES SET
Current_Price_Per_Liter = 105 WHERE
Fuel_Type_ID = 1;
```

→ Updates the fuel price for Petrol.

Query executed successfully



The screenshot shows the SQL Server Management Studio interface. On the left is the Object Explorer pane, which lists the database 'localhost\SQLEXPRESS (SQL Server 16.0.1000 - LAPTOP-FC1)' with its databases, security, server objects, replication, management, and XEvent Profiler. The main area is a query window titled 'SQLQuery1.s... Reddy (61)*'. It contains the following SQL code:

```
1 UPDATE FUEL_TYPES
2     SET Current_Price_Per_Liter = 105
3     WHERE Fuel_Type_ID = 1;
```

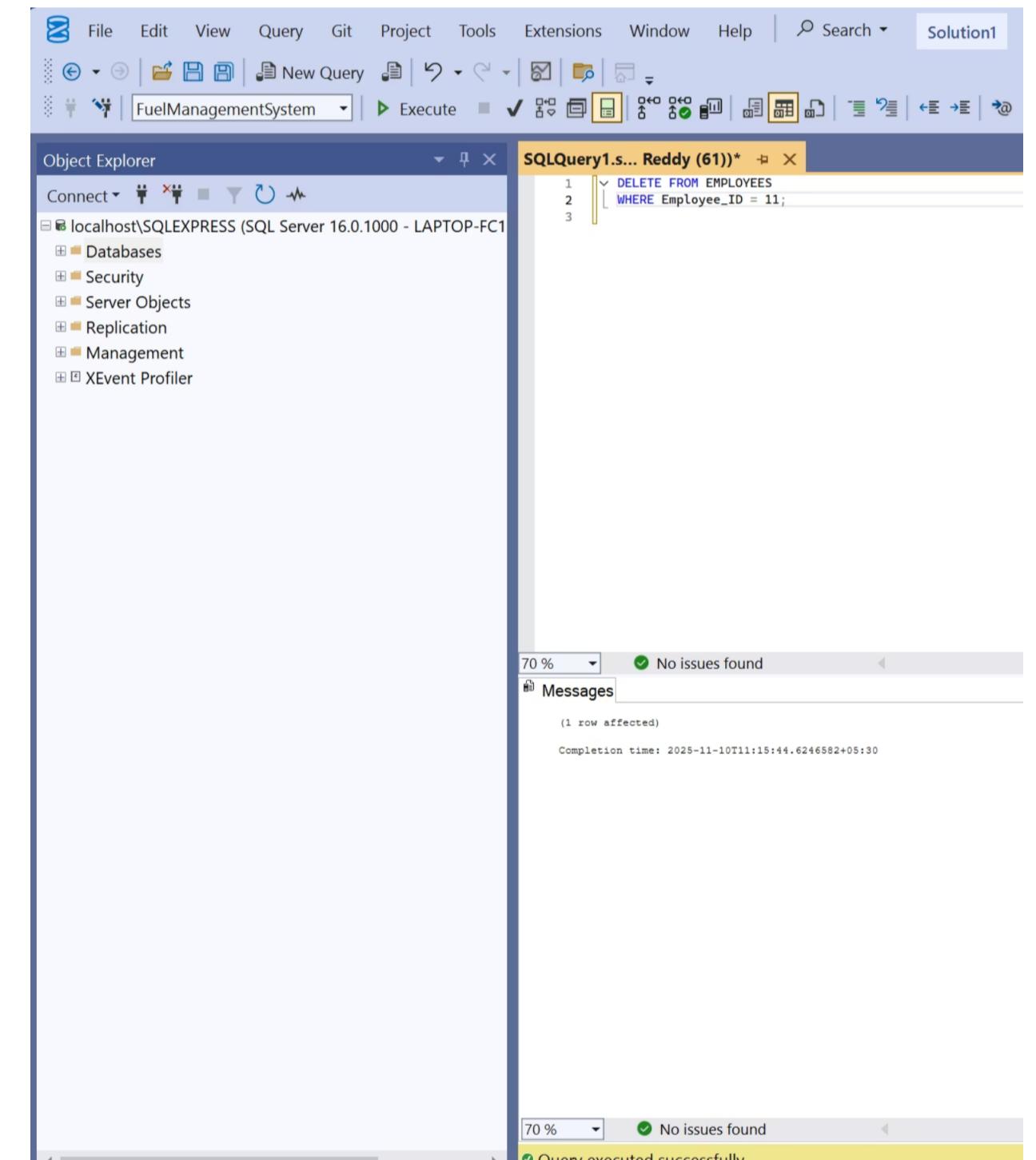
Below the query window is the 'Messages' pane, which displays the output: '(1 row affected)' and 'Completion time: 2025-11-10T11:15:09.2054718+05:30'. At the bottom of the screen, a yellow status bar says 'Query executed successfully.'

Delete Operation

```
DELETE FROM EMPLOYEES WHERE
Employee_ID = 11;
```

→ Removes the specified employee record.

Query executed successfully



The screenshot shows the SQL Server Management Studio interface. The Object Explorer pane is identical to the previous screenshot. The main area is another query window titled 'SQLQuery1.s... Reddy (61)*'. It contains the following SQL code:

```
1 DELETE FROM EMPLOYEES
2 WHERE Employee_ID = 11;
```

Below the query window is the 'Messages' pane, which displays the output: '(1 row affected)' and 'Completion time: 2025-11-10T11:15:44.6246552+05:30'. At the bottom of the screen, a yellow status bar says 'Query executed successfully.'

These CRUD operations demonstrate the system's ability to manage data effectively, ensuring data integrity and providing complete database functionality for the Fuel Management System.

Database Design & Implementation

Relational Schema Architecture

The system utilizes a normalized relational schema with integrity constraints. Primary and foreign keys ensure referential integrity. Stored procedures `ProcessSale()` and `RestockTank()` manage core business logic, while triggers automate tank level updates, ensuring data consistency.

01

Data Definition Language Complete DDL with PK, FK, and unique constraints for data integrity.

02

Stored Procedures Encapsulated business logic for sales and restocking with transactional guarantees.

03

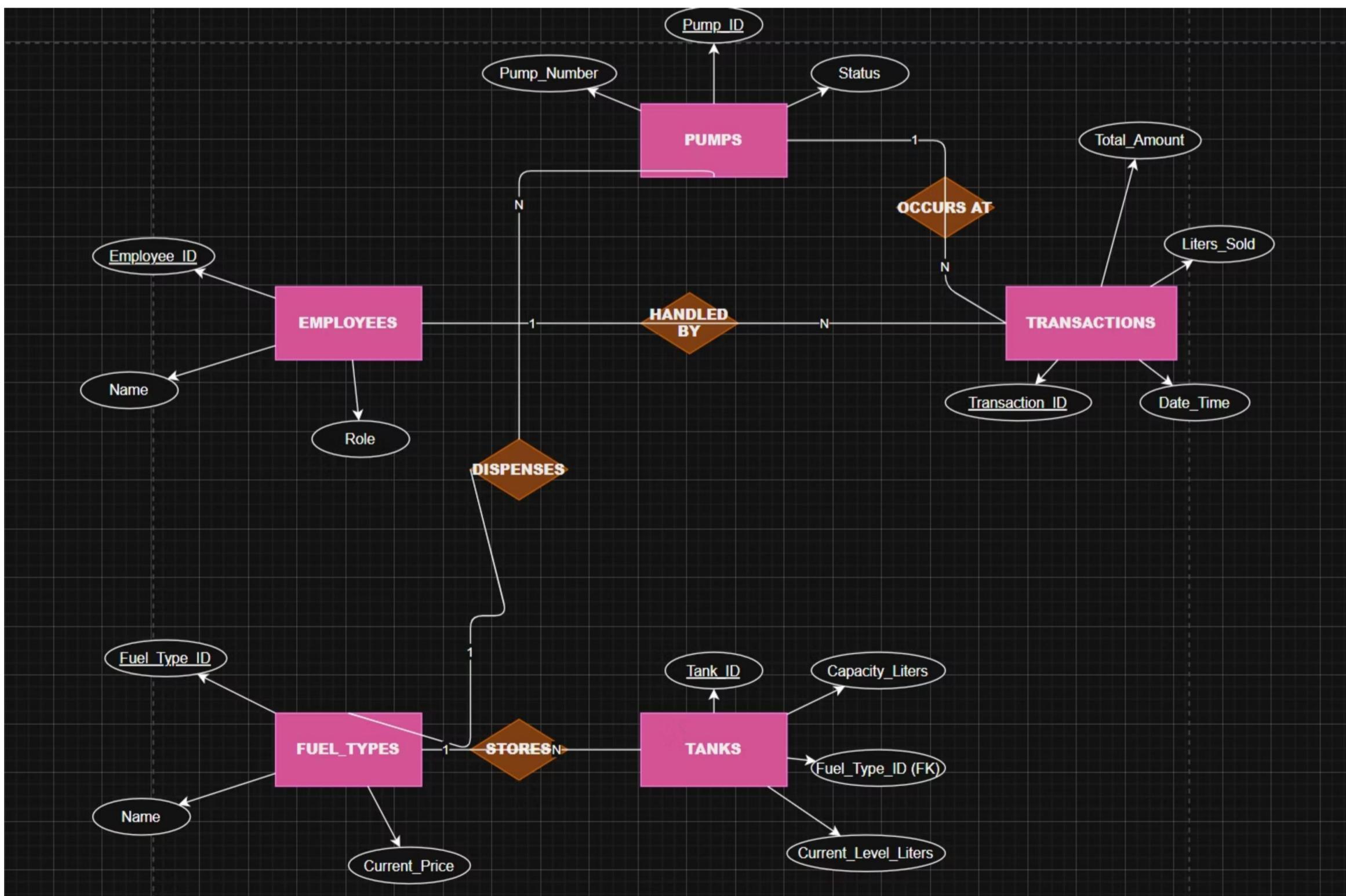
Trigger-Based Automation Real-time tank level updates triggered automatically on transactions.

04

Advanced Querying Complex SQL queries (JOINS, aggregates, subqueries) for reporting.

Entity-Relationship Diagram

Visual representation of core entity relationships.



Detailed Code Snippets

Stored Procedure: ProcessSale

```
CREATE PROCEDURE ProcessSale      @PumpID INT, @EmployeeID INT, @LitersSold
FLOATASBEGIN      DECLARE @PricePerLiter FLOAT;      INSERT INTO TRANSACTIONS (Pump_ID,
Employee_ID, Liters_Sold, Total_Amount, [datetime])      VALUES (@PumpID,
@EmployeeID, @LitersSold, @LitersSold * @PricePerLiter, GETDATE());END;
```

Handles sales, calculates total.

Stored Procedure: RestockTank

```
CREATE PROCEDURE RestockTank      @TankID INT, @LitersAdded FLOATASBEGIN      DECLARE
@Capacity FLOAT, @Current FLOAT;      IF (@Current + @LitersAdded > @Capacity)
RAISERROR('Capacity exceeded.', 16, 1);      ELSE      UPDATE TANKS SET
Current_Level_Liters = @Current + @LitersAdded;END;
```

Updates tank levels, checks capacity.

Trigger: UpdateTankLevel

```
CREATE TRIGGER trg_UpdateTankLevel ON TRANSACTIONS AFTER INSERTASBEGIN      UPDATE T
SET T.Current_Level_Liters = T.Current_Level_Liters - i.Liters_Sold      FROM TANKS T
INNER JOIN PUMPS P ON T.Fuel_Type_ID = P.Fuel_Type_ID;END;
```

Auto-updates tank levels post-sale.

Function: GetTotalRevenue

```
CREATE FUNCTION GetTotalRevenue()RETURNS FLOATASBEGIN      DECLARE @Total FLOAT;
SELECT @Total = SUM(Total_Amount) FROM TRANSACTIONS;      RETURN @Total;END;
```

Retrieves total revenue.

Flask Integration Example

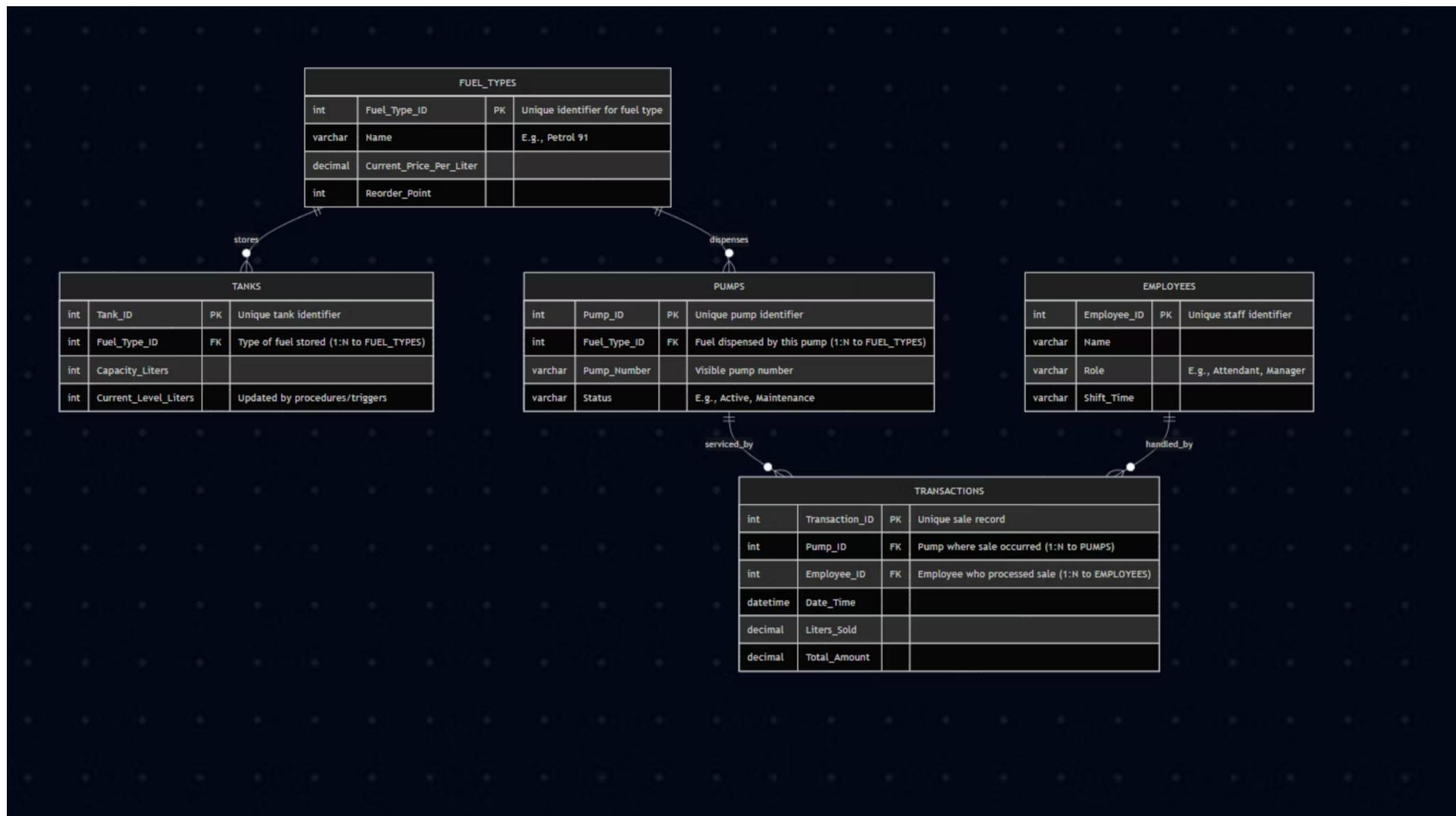
```
@app.route('/process_sale', methods=['POST'])def process_sale():    """Calls the
ProcessSale stored procedure."""
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("{CALL dbo.ProcessSale (?, ?, ?)}",
    pump_id, employee_id, liters_sold)
    conn.commit()
    flash("Sale processed
successfully! Inventory updated by Trigger.", 'success')
```

Flask integration with SQL stored procedures.

Key Implementation Features

- **Stored Procedures:** Encapsulate sales & restocking logic
- **Functions:** Enable calculations & data retrieval
- **Error Handling:** Validation & rollback for data integrity
- **Triggers:** Automatic inventory updates for data consistency
- **Flask Integration:** Connects web interface to database operations
- **Real-time Updates:** Automatic sync of transactions & inventory

Database Relationship Schema



Key Capabilities & Features

The system delivers comprehensive functionality designed for efficient fuel station management. Through real-time dashboards, automated transaction processing, and intelligent reporting, the platform transforms operational workflows and provides actionable business intelligence.

01

Real-Time Dashboard

Live visualization of tank levels and fuel pricing.

02

Transactional Processing

Validated SQL procedures for sales and restocking, with full audit trails.

03

Automated Inventory Updates

Database triggers for automatic stock level updates.

04

Advanced Reporting

Sophisticated queries for business insights.

05

Role-Based Management

Role-based employee access control.

Results & Conclusion

Operational Excellence

Automated fuel tracking and stock updates eliminate manual processes and reduce operational overhead

Data Accuracy

Systematic error reduction through trigger-based consistency and validated transaction processing

Business Intelligence

Analytical insights enabling informed management decisions through comprehensive reporting capabilities

System Impact

The Fuel Management System successfully demonstrates core DBMS concepts through practical implementation. By integrating relational design principles, stored procedures, and automated triggers, the system streamlines daily fuel station operations while maintaining data integrity. This project exemplifies how sophisticated database architecture transforms manual workflows into efficient, scalable business processes.

GitHub Repository: <https://github.com/riteshreddyyy/Fuel-Management-System>