Contents

[Introduction 3](#_Toc171766012)

[Executive Summary 3](#_Toc171766013)

[User and System Requirements 4](#_Toc171766014)

[User Requirements 4](#_Toc171766015)

[System Requirements 4](#_Toc171766016)

[Functional Requirements: 4](#_Toc171766017)

[Non-functional Requirements: 5](#_Toc171766018)

[Entities and relationships 6](#_Toc171766019)

[Relationships 7](#_Toc171766020)

[One-to-Many Relationships: 7](#_Toc171766021)

[Many-to-Many Relationships: 8](#_Toc171766022)

[One-to-One Relationships: 8](#_Toc171766023)

[Types of keys 8](#_Toc171766024)

[Primary Key (PK) 8](#_Toc171766025)

[Foreign Key 9](#_Toc171766026)

[Candidate Key 9](#_Toc171766027)

[Super Key 10](#_Toc171766028)

[ER diagram for Poly Pipe Company 10](#_Toc171766029)

[About ER Diagram 11](#_Toc171766030)

[Logical database diagram 11](#_Toc171766031)

[Converting ER diagram into logical database diagrams (Tables) 12](#_Toc171766032)

[Customers Table 12](#_Toc171766033)

[CustomerTelephoneNumbers Table 12](#_Toc171766034)

[Installations Table 12](#_Toc171766035)

[Workers Table 12](#_Toc171766036)

[Stores Table 12](#_Toc171766037)

[Supplies Table 13](#_Toc171766038)

[Payments Table 13](#_Toc171766039)

[InstallationWorkers Table 13](#_Toc171766040)

[Data Redundancy in Poly Pipe Company Database 13](#_Toc171766041)

[First Normal Form (1NF) 14](#_Toc171766042)

[Second Normal Form (2NF) 14](#_Toc171766043)

[Third Normal Form (3NF) 14](#_Toc171766044)

[implementation of Normalization in Poly Pipe Company Database: 14](#_Toc171766045)

[Example Tables 14](#_Toc171766046)

[Advantages and Disadvantages of Normalization 15](#_Toc171766047)

[Advantages of Normalization: 15](#_Toc171766048)

[Disadvantages of Normalization 16](#_Toc171766049)

[Anomalies in Database Systems 16](#_Toc171766050)

[1. Insertion Anomaly: 16](#_Toc171766051)

[2. Update Anomaly: 17](#_Toc171766052)

[3. Deletion Anomaly: 17](#_Toc171766053)

[Interface design and development 18](#_Toc171766054)

[Design tools Used 18](#_Toc171766055)

[First page 19](#_Toc171766056)

[Sign in /Signup page popup 20](#_Toc171766057)

[Menu page 21](#_Toc171766058)

[Customer interface (customer page) 22](#_Toc171766059)

[Supply page 23](#_Toc171766060)

[Store page 24](#_Toc171766061)

[Worker page 25](#_Toc171766062)

[Validation 26](#_Toc171766063)

[Examples 27](#_Toc171766064)

[Data Type Validation 27](#_Toc171766065)

[Constraint Validation 27](#_Toc171766066)

[Structured Validation 27](#_Toc171766067)

[Data Range Validation 28](#_Toc171766068)

[Code Validation 28](#_Toc171766069)

[Required Validation 28](#_Toc171766070)

[Business Rule Validation 28](#_Toc171766071)

[Validation and Verification (PP.Pankaj, 2020) 29](#_Toc171766072)

[Relationship of User and System 29](#_Toc171766073)

[Assess the Quality of System and User Requirements 29](#_Toc171766074)

[User Requirements 29](#_Toc171766075)

[System Requirements: 30](#_Toc171766076)

[Database 30](#_Toc171766077)

[Creating database 30](#_Toc171766078)

[Quarries to create 30](#_Toc171766079)

[Quarries to realationships 33](#_Toc171766080)

[Cascade operation Quarries 34](#_Toc171766081)

[Data insertion 37](#_Toc171766082)

[Checking for Proper operation 40](#_Toc171766083)

[Customer tables 40](#_Toc171766084)

[Installation Table 41](#_Toc171766085)

[Payment Table 42](#_Toc171766086)

[Stores Table 42](#_Toc171766087)

[Supplies Table 43](#_Toc171766088)

[Workers Table 43](#_Toc171766089)

[Database operations 44](#_Toc171766090)

[Select Operation 44](#_Toc171766091)

[Update Operation 44](#_Toc171766092)

[Where Operation 45](#_Toc171766093)

[Between Operation 45](#_Toc171766094)

[IN Operation 46](#_Toc171766095)

[GROUP BY Operation 46](#_Toc171766096)

[ORDER BY Operation 47](#_Toc171766097)

[Having Operation 47](#_Toc171766098)

[Test Plan 48](#_Toc171766099)

[Manual Database Testing Process 48](#_Toc171766100)

[Prepare Your Test Environment: 48](#_Toc171766101)

[Craft Test Cases: 48](#_Toc171766102)

[Execute Test Cases: 49](#_Toc171766103)

[Verify Results: 49](#_Toc171766104)

[Document and Report: 49](#_Toc171766105)

[Automated Database Testing Process 49](#_Toc171766106)

[Set Up Automated Test Environment 49](#_Toc171766107)

[Develop Automated Test Scripts: 49](#_Toc171766108)

[Execute Tests: 49](#_Toc171766109)

[Validate Results: 49](#_Toc171766110)

[Review and Report: 49](#_Toc171766111)

[Database Testing Tools 50](#_Toc171766112)

[Types of Database Testing 50](#_Toc171766113)

[System Testing Overview 50](#_Toc171766114)

[System Testing Definition: 50](#_Toc171766115)

[Types of System Testing 50](#_Toc171766116)

[Justification for System Testing 52](#_Toc171766117)

[Create Operation using GUI 53](#_Toc171766118)

[Check For Database 54](#_Toc171766119)

[Delete Operation 55](#_Toc171766120)

[Check For Database 55](#_Toc171766121)

[Edit Operation (Update) 57](#_Toc171766122)

[Before Update 57](#_Toc171766123)

[Update Operation 57](#_Toc171766124)

[Implementation 58](#_Toc171766125)

[Feedback 59](#_Toc171766126)

[User Feedback: 59](#_Toc171766127)

[User Interface and Usability: 59](#_Toc171766128)

[Data Entry and Retrieval: 59](#_Toc171766129)

[Feature Requests: 59](#_Toc171766130)

[Security Concerns: 59](#_Toc171766131)

[Training and Support: 59](#_Toc171766132)

[Developer Feedback: 60](#_Toc171766133)

[Performance and Scalability: 60](#_Toc171766134)

[Integration Challenges: 60](#_Toc171766135)

[Technical Documentation: 60](#_Toc171766136)

[Feedback Mechanism: 60](#_Toc171766137)

[Security Architecture: 60](#_Toc171766138)

[User Guide 61](#_Toc171766139)

[User Guide Database 61](#_Toc171766140)

[Logical Design 61](#_Toc171766141)

[Security Mechanisms 64](#_Toc171766142)

[**Operations** 64](#_Toc171766143)

[User Guide For GUI 65](#_Toc171766144)

[Functions 65](#_Toc171766145)

[Important 65](#_Toc171766146)

[Customer form and Create operation Windows 66](#_Toc171766147)

[Guide 67](#_Toc171766148)

[CRUD Operation 68](#_Toc171766149)

[Create a Customer 69](#_Toc171766150)

[GUI posts 71](#_Toc171766151)

[Conclusions 73](#_Toc171766152)

[REFERENCES 75](#_Toc171766153)

# List of Tables

[Table 1:ER diagram For Polly pipe Company Database 15](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766160)

[Table 2:Logical Database Diagram 16](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766161)

[Table 3:Customer Table 16](#_Toc171766162)

[Table 4:Customer Telephone Number 17](#_Toc171766163)

[Table 5:Installations Details 17](#_Toc171766164)

[Table 6:Store Table 17](#_Toc171766165)

[Table 7:Supplies Table 17](#_Toc171766166)

[Table 8:Payment Table 18](#_Toc171766167)

[Table 9:Installation worker table 18](#_Toc171766168)

[Table 10:Customers table 19](#_Toc171766169)

[Table 11:Customers Telephone Number Table 19](#_Toc171766170)

[Table 12:Validation and verification according to various aspects 34](#_Toc171766171)

# Table of figures

[Figure 1:My sql Logo 25](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766178)

[Figure 2: WordPress Icon 25](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766179)

[Figure 3:PHP my admin 25](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766180)

[Figure 4:First page 25](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766181)

[Figure 5:First page without login 26](#_Toc171766182)

[Figure 6:Sign in sign up popup windows 27](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766183)

[Figure 7:Listdown Menu 28](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766184)

[Figure 8:Working Of List down Menu 28](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766185)

[Figure 9:Insert Customer 29](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766186)

[Figure 10:Customer page 29](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766187)

[Figure 11:New Supply page 30](#_Toc171766188)

[Figure 12:New Payment Page 30](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766189)

[Figure 13:New store Page 31](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766190)

[Figure 14:Worker Form Page 32](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766191)

[Figure 15:New Worker page 33](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766192)

[Figure 16:Customer table database 47](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766193)

[Figure 17:Customer Number table check 47](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766194)

[Figure 18:Check Installation workers Table 48](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766195)

[Figure 19:Check installation Table 48](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766196)

[Figure 20:Check Payment table 49](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766197)

[Figure 21:Check store table 49](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766198)

[Figure 22:Check Supplies Table 50](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766199)

[Figure 23:Check Workers Table 50](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766200)

[Figure 24:Select Operation 51](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766201)

[Figure 25:Update Operation 51](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766202)

[Figure 26:Where Operation 52](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766203)

[Figure 27:Between Operation 52](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766204)

[Figure 28:IN operation 53](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766205)

[Figure 29:Group By Operation 53](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766206)

[Figure 30:Order By Operation 54](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766207)

[Figure 31:Having Operation 54](#_Toc171766208)

[Figure 32:Add a customer 60](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766209)

[Figure 33:Add a number to the same customer 60](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766210)

[Figure 34:Updated Customer 61](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766211)

[Figure 35:Updated Number 61](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766212)

[Figure 36:Delete a Customer 62](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766213)

[Figure 37:Deleted Database entry 62](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766214)

[Figure 38:Cascadely Deleted Number In the Database 63](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766215)

[Figure 39:Deleted Entry Shoun In GUI 63](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766216)

[Figure 40:Worker Table before Update 64](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766217)

[Figure 41:Updating Last Name 64](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766218)

[Figure 42:Updated Database 65](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766219)

[Figure 43:Updated GUI Fields 65](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766220)

[Figure 44:Database 68](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766221)

[Figure 45:Web GUI 72](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766222)

[Figure 46:Insert Forms 73](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766223)

[Figure 47:Forms In the GUI 73](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766224)

[Figure 48:Guide For CRUD 74](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766225)

[Figure 49:Crud Operation 75](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766226)

[Figure 50:Creating a Customer Using GUI 76](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766227)

[Figure 51:Updated Database Entry 77](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766228)

[Figure 52:GUI Update 77](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766229)

[Figure 53:Inbuild User Guide For Database 78](#_Toc171766230)

[Figure 54:Inbuild USer GUIDE GUI 79](file:///E:\%5b03%5d%20Proffetional\%5b03%5d%20Assignment\%5b05%5dGadapi\%5b06%5dDATAbase%20WEb\Database%20Solution%20for%20Polly%20Pipe.docx#_Toc171766231)

# Introduction

Data-Driven Decision-making (DDD), as one of the most critical disciplines, relates to issues associated with data storage, analysis, and security. This assignment seeks to explain how the writer’s team created a coherent database system for Polly Pipe, a water sports firm in Braintree, England, that offers water plumbing and installation services.

This is followed by a list of user and system requirements which are fundamental in establishing the framework of creating a sound database. Some of the activities involved in this process are; Generating a conceptual Entity-Relationship (ER) diagrams for the purpose of providing a visual representation of the entities, their relationships, and the restraints present in the databases. The conceptual model as stated above is transformed to the creation of a logical database design; this involves defining primary and foreign keys and maintaining referential integrity to avoid data anomalies.

After the design phase of the project, the report also shows a creation of the table structure of the relational database in SQL Data Definition Language (DDL) statement forms. The procedure of creating a database is accomplished with the help of an appropriate Integrated Development Environment or Interactive Administration Tool, for example with MySQL Workbench or phpMyAdmin. An intuitive shell is designed which allows performing the elementary operations towards the data stored in the databases: insert, update, delete, etc. ; yet, strong protection means are incorporated to prevent unauthorized access.

The report also provides DML type of queries to illustrate the type of queries used in extracting and manipulating data in the database. The sample test plan highlights the methodologies to use with the database to test it against the determined requirements, the test cases, and results.

The feedback from ‘ordinary’ users and application developers is then gathered to analyze the actual suitability of the database. According to the gathered feedback, changes are proposed to be made as the following are some of the major areas to be improved.

In the last of it the technical documentation and the user guide is also incorporated. The technical documentation includes various diagrams including the use case diagrams, class diagrams, flow charts and Data flow diagrams (DFD) that outlines the flow of data in the system. The manual overviews practical tips on its usage and gives the detailed description of each step in using the database interface.

# Executive Summary

This paper describes Polly Pipe company’s Database – a water sports provider and installer situated in Braintree, England. This consists of identifying the user and system requirements, along with developing the initial conceptual ER model followed by mapping the logical design for the database. Created in an appropriate program through means of SQL Data Definition Language statements, the system has sufficient levels of security and a simple to use model for the data. Such DML queries’ examples are described to explain how data manipulation occurs effectively. An extensive testing plan ceases system malfunctions and a usability review of the feedback coming from users and developers is useful. Technical documentation is comprised of diagrams outlining data flow and interactions with the systems as well as a working guide to the service. Based on continuous innovation, this best-of-breed database system designed by a specialist in one of the leading IT companies in Vietnam will help Polly Pipe better manage its operations by properly dealing with data and effective security arrangements.

# User and System Requirements

## User Requirements

User requirements specify which functionalities and features should be implemented in the system to satisfy the user’s needs. These are presented in natural language, frequently using diagrams, to define the nature of services to be delivered and bound of the operation of the system. They include general system descriptions of functions up to the detailed functional requirements that are crucial to the functionality of a system. This initial phase is considered important in software development because it creates the framework through which the system is built, developed, and supported.

This paper focuses on the process of integrating the accounting system of “Poly Pipe” company, a water sports provider and installer service provider with a database system based in Braintree, England. Due to its large amount of customers and complex installations, good management and organization of customer information as well as service is also required. The system should:

* Document the details of customers and installations as much as possible.
* Maintenance of specialized installations to fit the individual client’s requirements.
* Group installations according to type to aid in their organization.
* Ways of managing work and people include the following; Proper assignment of workers to the different installations depending on their specialization and the time they are willing to spend at the installations.
* Assist in the conveying of job details to the workers.

Poly Pipe’s management hires development team to turn these requirements into the functional database system necessary for keeping records of the company’s sales and inventory. This people involved the determination of attributes which should uniquely define a record, those that should be referenced from others, the entities, and relationships important for organizing the data. Thus, ensuring these user requirements are met would better the database system of Poly Pipe in performance, customer satisfaction, and future capability.

## System Requirements

Descriptive system requirements include the specifics of the functionality of the software system, offered services, and limitations of the system’s functioning. As for the system requirements, they and often contained in the contract between the system buyer and developers, sometimes called a functional specification, define in detail what has to be created.

### Functional Requirements:

Descriptive requirements fall under a category that has a close relationship with the main tasks assigned by the users. These are core specifications that are fundamental to the running of the company and thus have to be met for continuous functionality. For Poly Pipe, functional requirements include:For Poly Pipe, functional requirements include:

* Database Management: Update customers’ records, stored data, worker information, and any other pertinent information.
* CRUD Operations: The functionalities that should be enabled include; create record, update record, delete record and search record.
* Worker Management: Something that should be applied is the formation of a reminder system to regularly wake up the workers about their assignments.

### Non-functional Requirements:

Non-functional requirements improve usability, and thus user experience, even though having them is not strictly required. These requirements focus on making the system more useful and enhance its functionality for efficient use. Key non-functional requirements for Poly Pipe's database system are:Key non-functional requirements for Poly Pipe's database system are:

* User-Friendly Interface: The system should be user friendly from interfaces to its design.
* Efficient Resource Usage: Spare resources such as servers and system to the least possible in order to reduce the overheadcosts.
* Aesthetic Design: Make the appearance of the site appealing and intuitive, so that clients will appreciate the fact that the given product was designed keeping them in mind.
* Cost-Effectiveness: Sustain solutions with low upstream costs in terms of points of interaction and low cost of maintenance and future-proof implementation.
* Security Measures: This includes protecting the system from getting attacked by viruses, having the system get infected by malware, and to avoid unauthorized persons getting an easy access to the system.
* Ease of Maintenance: Create the system in such a way that modifications and maintenance are easy to perform in order to avoid a long lasting system downtime.
* Scalability: Make sure that the system design can handle any extension and improvement of the system in the future.

It is vital to include a System Requirements Specification (SRS), also known as Software Requirements Specification since it defines the intended system features and functionality, users’ requirement, and operational environment limitations. It acts as a guide to the developers so that they can always satisfy the client as well as the end customers.

# Entities and relationships

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Attributes** | **Primary Key** | **Foreign Keys** |
| Customers | CustomerID, Name, Email, Category | CustomerID |  |
| CustomerTelephoneNumbers | CustomerID, TelephoneNumber | (CustomerID, TelephoneNumber) | CustomerID (FK to Customers) |
| Installations | InstallationID, NameOfPlace, PlaceAddress, Type, CustomerID | InstallationID | CustomerID (FK to Customers) |
| Workers | WorkerID, FirstName, LastName, Category, Email, TelephoneNumber, ItemNumber, WorkNumber, WorkPlaceNumber | WorkerID | ItemNumber (FK to Supplies), WorkNumber (FK to Installations) |
| Stores | ItemID, IssuedDate, ResponsiblePerson, ItemPrice | ItemID |  |
| Supplies | ItemNumber, PricePerUnit, Quantity, TotalPrice, Purpose, StoreID | ItemNumber | StoreID (FK to Stores) |
| Payments | BillNumber, BuyerName, Category, DatePay, CustomerID | BillNumber | CustomerID (FK to Customers) |
| InstallationWorkers | InstallationID, WorkerID | (InstallationID, WorkerID) | InstallationID (FK to Installations), WorkerID (FK to Workers) |
| WorkerStores | WorkerID, ItemID | (WorkerID, ItemID) | WorkerID (FK to Workers), ItemID (FK to Stores) |

Here's a brief description of each entity in the Poly Pipe Company database:Here's a brief description of each entity in the Poly Pipe Company database:

* **Customers:** This entity captures detail information of the customers such as CustomerID of the customer, the name of the customer, email address, and the category that the customer belongs to. It becomes a data base in maintaining the details of customers and their needs and preferences for classification.
* **CustomerTelephoneNumbers:** Consequently, this entity is related to Customers through the CustomerID and enables the handling of several telephone numbers linked to each customer. They favor contact management as all details of the communication process are well captured and easily arranged.
* **Installations**: Members of Installations performed by the Poly Pipe Company which has Installation ID, Name of Place, Place Address, Installation Type, and related Customers File using Customer ID. This entity helps in the monitoring and control of various installation projects from different places.
* **Workers**: This table is more detailed than the Workers table and outlines the workers’ identification, names, employment category, communication details such as email and phone number, number item identification code ItemNumber and number installation identification code WorkNumber. This is useful in planning a suitable workforce for an organization as well as assignment of duties.
* **Stores**: It save information concerning available supplies through some attributes such as ItemID, issuance date and the person responsible for the issuance, item price and any other related parameter. The capability of this entity is to facilitate inventory management and tracking of supplies used in the installations and other company processes.
* **Supplies**: Contains attributes that highlight information of supply such as ItemNumber, price per unit, quantity, price in extended form, use of supply, and StoreID linking it with Stores table. It effectively coordinates supply chain and guarantees delivery of materials required for the functions of the company.
* **Payments**: Records the payment transaction made by the customers based on the factors such as BillNumber, buyers name, category of payment, date of payment and customer number that refers to Customers. It enables proper maintenance of financial records of the organization and assists in billing its customers.
* **InstallationWorkers**: Reflects the existence of the many-to-many relationship that is between Installations and Workers but only connects concrete workers and installations that the former is linked to. It can be useful in the management of labor distribution in installation projects in a manner that is effective.

## Relationships

### One-to-Many Relationships:

* TelephoneNumbers to Customers (A certain customer can have many telephone numbers)
* The relationships between Customers and Payments are many-to-many (one Customer can make several Payments).
* Customers to Installations (This relation is a one to many relation where one customer can have many installations. )

### Many-to-Many Relationships:

* Workers to Installations (The relationship is many-to-many as several workers may be assigned to the same installation).
* Stores to Workers (As many workers can be assigned to many stores as many times as possible)

### One-to-One Relationships:

* Installations to Supplies table where each installation is related to certain supplies that are used for it.
* Slow File Generation, Downloads, Transfers & Archiving to Customer (Each installation is for one customer)
* Labor to Stock (One form is that a worker reserves responsibility for a single Store item. )

These entities and relationships describe a work in progress with regard to database design to manage customers, installations, workers, supplies payments and store items needed by Poly Pipe Company.

## Types of keys

* Primary Key (PK)
* Foreign Key (FK)
* Candidate Key
* Alternate Key
* Composite Key
* Super Key
* Unique Key
* Surrogate Key

### Primary Key (PK)

A primary key is a key used in the database to uniquely identify a record in a database. This must not be NULL values and a table can only have one primary key .This should be a unique index for a record. This can be in a two types.

* Single-column Primary Key

If there is a unique index that can be used to identify a record in a database that fields can be used as a primary key .

Here’s How to Implement in a Database:

**CREATE TABLE Customer (**

**CustomerID INT PRIMARY KEY,**

**CustomerName VARCHAR(255),**

**Address VARCHAR(255)**

**);**

* Composite Primary Key

If there is no unique index that can be used to identify a record in a database we should create such index by combining two fields.

Here’s How to Implement in a Database:

CREATE TABLE OrderDetails (

OrderID INT,

ProductID INT,

Quantity INT,

PRIMARY KEY (OrderID, ProductID)

);

### Foreign Key

A field that uniquely identify a row in another table. this can have duplicate values NULL values as such. Here simple idea is to direct to a another table from a one table

EXAMPLE:

Here’s how to implement it in a Sql database:

**CREATE TABLE Orders (**

**OrderID INT PRIMARY KEY,**

**OrderDate DATE,**

**CustomerID INT,**

**FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)//here direct to customer table**

**);**

**CREATE TABLE Customer (**

**CustomerID INT PRIMARY KEY,**

**CustomerName VARCHAR(255),**

**Address VARCHAR(255)**

**);**

Foreign key is important to make connection between tables to implement functions such:

* On delete Cascade(if a customer is deleted all records will be deleted)

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE,

CustomerID INT,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

ON DELETE CASCADE

);

* On Update Cascade(if customer id Updated in other tables also customer id is Updated)

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE,

CustomerID INT,

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

ON UPDATE CASCADE

);

### Candidate Key

This is when there are more than one suitable candidates for Primary key one is choosen as primary key and others are . identified as a candidate keys.

Implementing:

CREATE TABLE Employee (

EmployeeID INT NOT NULL,

NationalInsuranceNumber VARCHAR(10) NOT NULL,

Email VARCHAR(255) NOT NULL,

Name VARCHAR(255),

PRIMARY KEY (EmployeeID),

UNIQUE (NationalInsuranceNumber),

UNIQUE (Email)

);

### Super Key

One or more attributes together to form unique identifier. Does not need to be minimal.

# ER diagram for Poly Pipe Company

Table 1:ER diagram For Polly pipe Company Database

## About ER Diagram

One of the main instruments in the Poly Pipe Company’s database system design is the ER (Entity-Relationship) diagram. Their advantages include being able to provide a visual presentation of the data entities and their attributes and also the relationship between the data entities. This is because whereas Customers, Installations, Workers, Stores, Supplies, and Payments are entities, Customers and Installations are designated as the primary keys which are unique in the whole table, whereas other entities-Workers, Stores, Supplies and Payments have foreign keys from the main entities. It serves as a framework through which the entire process of designing the relational database is conducted; thus, making it easier for all the user and system specifications to be captured correctly.

# Logical database diagram

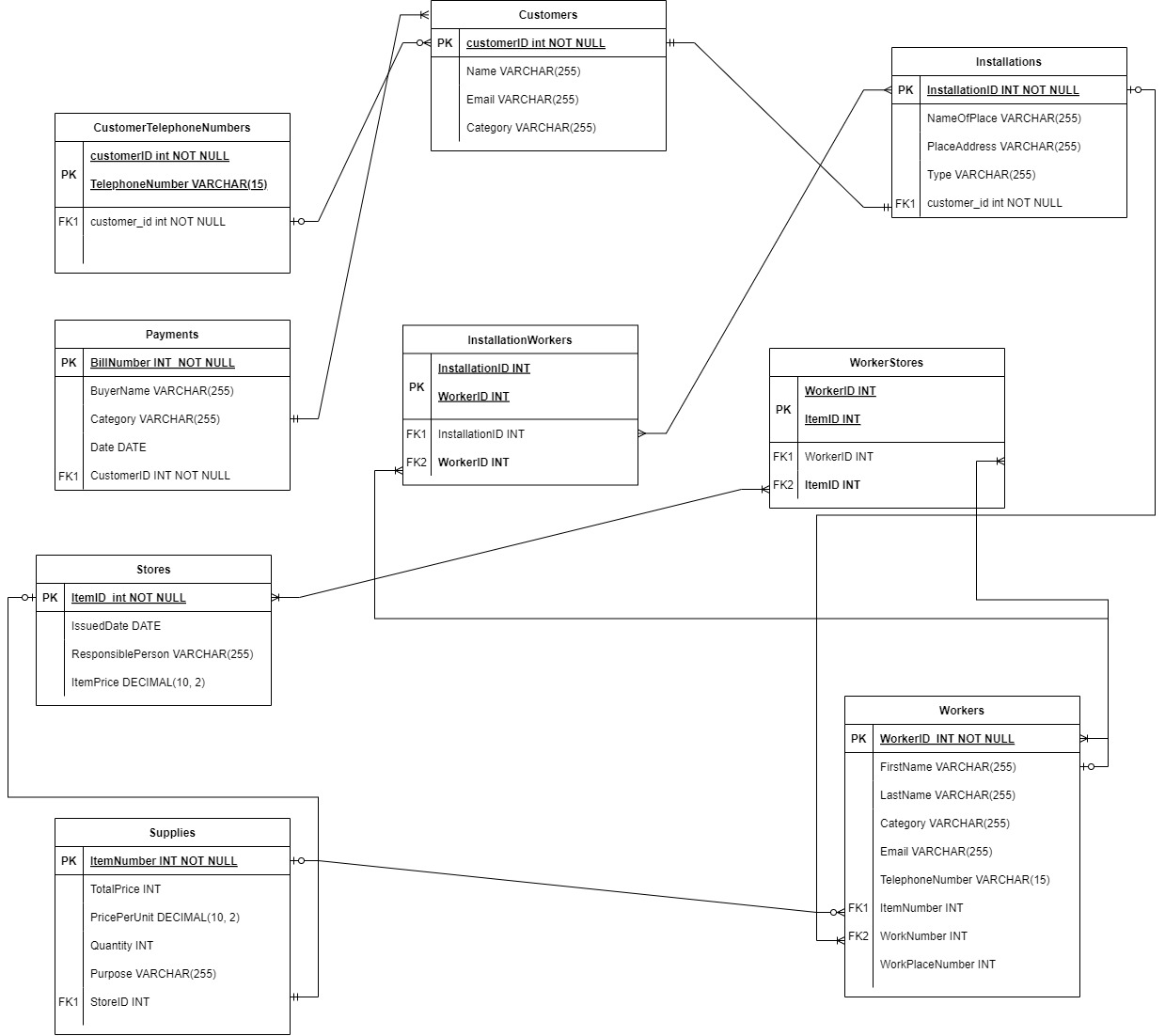


Table 2:Logical Database Diagram

# Converting ER diagram into logical database diagrams (Tables)

(Underlined words in following tables are primary keys and foreign keys)

## Customers Table

Table 3:Customer Table

|  |  |  |  |
| --- | --- | --- | --- |
| **CustomerID** | **Name** | **Email** | **Category** |
|  |  |  |  |

## CustomerTelephoneNumbers Table

Table 4:Customer Telephone Number

|  |  |
| --- | --- |
| **CustomerID** | **TelephoneNumber** |
|  |  |

## Installations Table

Table 5:Installations Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **InstallationID** | **NameOfPlace** | **PlaceAddress** | **Type** | **CustomerID** |
|  |  |  |  |  |

## Workers Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WorkerID** | **FirstName** | **LastName** | **Category** | **Email** | **Telephone**  **Number** | **ItemNumber** | **Work**  **Number** | **WorkPlace**  **Number** |
|  |  |  |  |  |  |  |  |  |

## Stores Table

Table 6:Store Table

|  |  |  |  |
| --- | --- | --- | --- |
| **ItemID** | **IssuedDate** | **ResponsiblePerson** | **ItemPrice** |
|  |  |  |  |

## Supplies Table

Table 7:Supplies Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ItemNumber** | **PricePerUnit** | **Quantity** | **TotalPrice** | **Purpose** | **StoreID** |
|  |  |  |  |  |  |

## Payments Table

Table 8:Payment Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BillNumber** | **BuyerName** | **Category** | **DatePay** | **CustomerID** |
|  |  |  |  |  |

## InstallationWorkers Table

Table 9:Installation worker table

|  |  |
| --- | --- |
| **installationID** | **WorkerID** |
|  |  |

These tables are prepared from the ER diagram and depict that how the logical data model is mapped, and the relationships between the tables are defined by the use of primary and foreign keys.

# Data Redundancy in Poly Pipe Company Database

In data redundancy, there is always copying of the same information into different storage areas to create multiple copies. This may reach a point when there is poor call flow and utilization, additional storage, and costs incurred in replacement. When implemented in a database system, data redundancy has problems like; inconsistency in updating the data as well as wastage of storage space.

A common issue found in the Poly Pipe Company database is the redundancy of data, for which normalization is applied. Normalization is one of the systematic strategies where repeated occurrence of data is broken down in tables with the intention of enhancing data accuracy. The following are the three normal forms (1NF, 2NF, 3NF) that are enacted to minimize redundancy and for better database structure:

## First Normal Form (1NF)

First Normal Form simply states that in a table each column should possess elementary (binned) values and all the values in a column must belong to the same domain. During this stage, subgroups that may be present in any fields are deleted, the idea being to have each field contain only one value.

Example:

Concerning the CustomerTelephoneNumbers table, each telephone number of a customer is split into numerous rows, in contrast with having one column with many tel. nums.

## Second Normal Form (2NF)

Second Normal Form can be attained only when a relational table is in the First Normal Form and all the attributes that are not key fields are fully dependent on the key field or fields. This implies that all of the attributes should depend on all the values of the primary key and not a few of these values.

Example:

The relationship used in the Installations table is CustomerID, which guarantees that each system installed belongs to a certain customer; thus, all the pieces of information stored in the Installations table are functionally dependent on the Installation ID.

## Third Normal Form (3NF)

Third Normal Form is usually obtained if the table is in 2NF and all the attributes depend on the primary key and not on the other attributes which are not keys. This helps in eradicating the transitive dependencies which means that non-key attributes cannot be dependent on other non-key attributes.

Example:

Every record in the Payments table is dependent on the CustomerID and there are no additional fields in Payments which might be stored in the other table, for example, customer details are stored in the Customers table and they are not stored repeatedly in the Payments table.

From the above normalization steps, the Poly Pipe company’s database is well organized with any unnecessary repetition in order to increase efficient storage and accuracy of results.

## implementation of Normalization in Poly Pipe Company Database:

To demonstrate how the normalization process is applied, below are the example tables derived from the ER diagram after normalization:To demonstrate how the normalization process is applied, below are the example tables derived from the ER diagram after normalization:

### Example Tables

|  |  |  |  |
| --- | --- | --- | --- |
| **CustomerID** | **Name** | **Email** | **Category** |
| 1 | John Doe | john@example.com | Retail |
| 2 | Jane Smith | jane@example.com | Wholesale |
| 3 | Alice Brown | alice@example.com | Retail |

#### Customers Table

Table 10:Customers table

#### CustomerTelephoneNumbers Table

Table 11:Customers Telephone Number Table

|  |  |
| --- | --- |
| **CustomerID** | **TelephoneNumber** |
| 1 | 123-456-7890 |
| 1 | 987-654-3210 |
| 2 | 555-555-5555 |

## Advantages and Disadvantages of Normalization

### Advantages of Normalization:

#### Eliminates Data Redundancy:

Normalization makes it possible for data to be stored only once and not repeatedly this enhances security on data. This cuts the storage cost and makes sure that every update or change to the data being held is in compound across the database.

#### Improves Data Integrity:

When the tables adopt normalized forms, the frequencies for most of the anomalies such as update, insert and delete anomalies are reduced significantly. This helps in maintaining accuracy and consistency of the data that is to be fed into the database system.

#### Enhances Data Maintenance:

Multiple repeats are also reduced in normalized databases and as such normalized databases require lesser in maintenance. It also reduces the number of places where changes need to be made; this is especially advantageous when presenting or erasing data.

#### Facilitates Efficient Data Retrieval:

Structured queries become more efficient when normalization is done since there is reduced repetition of data, hence improving on the speed in the process of data retrieval.

#### Supports Better Data Organization:

Through processes of developing a logical form of grouping the data into normalized tables and producing relationships between them, the study of normalization aids in the proper arrangement of data.

#### Ensures Data Scalability:

Normalized databases are more scalable since they are organize to store more data and to support more complex queries without diminishing the system performances.

### Disadvantages of Normalization

#### Complexity of Queries:

Consequently, as the database becomes more normalized, the number of tables that are with it rises. This can result to other join queries to fetch the needed data which may be cumbersome to write and comprehend.

#### Performance Overhead:

The inclusion of many tables can cause the level of join to sometime introduce performance penalty especially when dealing with large data sets or more complex queries.

#### Initial Design Effort:

Normalization of the database needs to be planned and analyzed considerably before the implementation of normalized structures. The first time when a database is to be designed and made standardized can be a little lengthy and for this, a good understanding of the data as well as the relation between them is needed.

#### Potential for Over-Normalization:

The extreme of this process is a scenario where the database is over-normalized, resulting in many joins; this affects performance. Finding the right balance between normalization and performance is the key message of this paper.

#### Difficulty in Making Changes:

There is however one major disadvantage of normalization in a database that is, after normalization, making alterations on the structure of the database for example adding new attributes in the model or changing the relationships maybe slightly intricate. It might involve one or several tables and their relationships, and this can be rather inconvenient.

#### Reduced Redundancy vs. Increased Complexity

Normalization helps to minimize redundancy in a database and the drawback of this is that the schema of the database becomes more complex. This is a trade off situation that should be well balanced in order to avoid complications of the system while lowering the level of redundancy.

Normalization in simple terms makes sure that data is correct and does not have duplicity and helps in better structuring of data. However, it also adds some difficulties as for writing the queries and may affect the performance if improperly optimized. Consequently, what is required is to strike the right balance between normalization and the implementation-computation costs to construct a good and efficient database.

## Anomalies in Database Systems

### 1. Insertion Anomaly:

An insertion anomaly implies whereby certain elements cannot be posted to the database without the company of other elements.

Example:

In the Supplies table, let us say you wish to enter a new supply item but you do not know which store the supply belongs to. If the StoreID is dependent on the Stores table as the foreign key, one cannot add this new supply item without first adding a store.

Suppose we try to insert the following query is used to insert records into the Supplies table:

VALUES (101, 5, 99, 100, ‘NewSupply’);

This will fail if StoreID is a required field which has to hold a valid store’s identifier.

### 2. Update Anomaly:

The update anomaly specifically arises when alterations to data in one row affect subsequent rows because of data repetition.

Example:

In Customers If a certain customer’s email address exists several times e. g. in Customers table and in CustomerEmails table, then if the email address in Customers table is changed for a certain customer and new email address is written instead of the old one while in the CustomerEmails table the new email address has not been registered, then the presence of such data is inconsistent.

Suppose we update the email address in Customers

UPDATE Customers

SET agenda = ‘New agenda of a meeting with the potential client. ’

WHERE CustomerID = 1;

This is especially provocative if we do not remember and do not update, for example, the same e-mail in another connected table (e. g. , CustomerEmails) Then e-mail addresses will not only be different but also contradictory and different in the entire database.

### 3. Deletion Anomaly:

Deletion anomaly comes about when, by deleting a record from the database, useful information is also deleted.

Example:

For instance, in the Installations table, if a customer is deleted and the foreign key constraint is set to cascade when deleting, the corresponding customers’ installation records will be removed.

Suppose we execute:

DELETE FROM Customers

WHERE CustomerID = 1;

If there are structures related to this customer, they too will be wiped out because of cascade delete. What this means is that we are left without critical installation data that can be used for purposes such as record keeping or when preparing reports.

# Interface design and development

The main idea of interface design for the Polly Pipe database system is to ensure the convenient management of data in the company. This consists of the input forms for new customers, workers, supplies, installation, payments, update form, and delete form. The design consists of buttons and pop-up menus to add related entities including assigning workers to installations or defining the kind of equipment required. This makes it easier for the users to fill the data set since they are informed or reminded of the areas to complete or input correct data. Other features that are displayed in the interface include the search as well as filter features that are useful in the retrieval of records within the shortest time, therefore increasing productivity. Thus, by focusing on such aspects of usability and accessibility of the interface, most of the operations and data management requirements of Polly Pipe shall be addressed.

## Design tools Used



Figure 1:My sql Logo

Figure 2: WordPress Icon



Figure 3:PHP my admin

## First page

Figure 4:First page

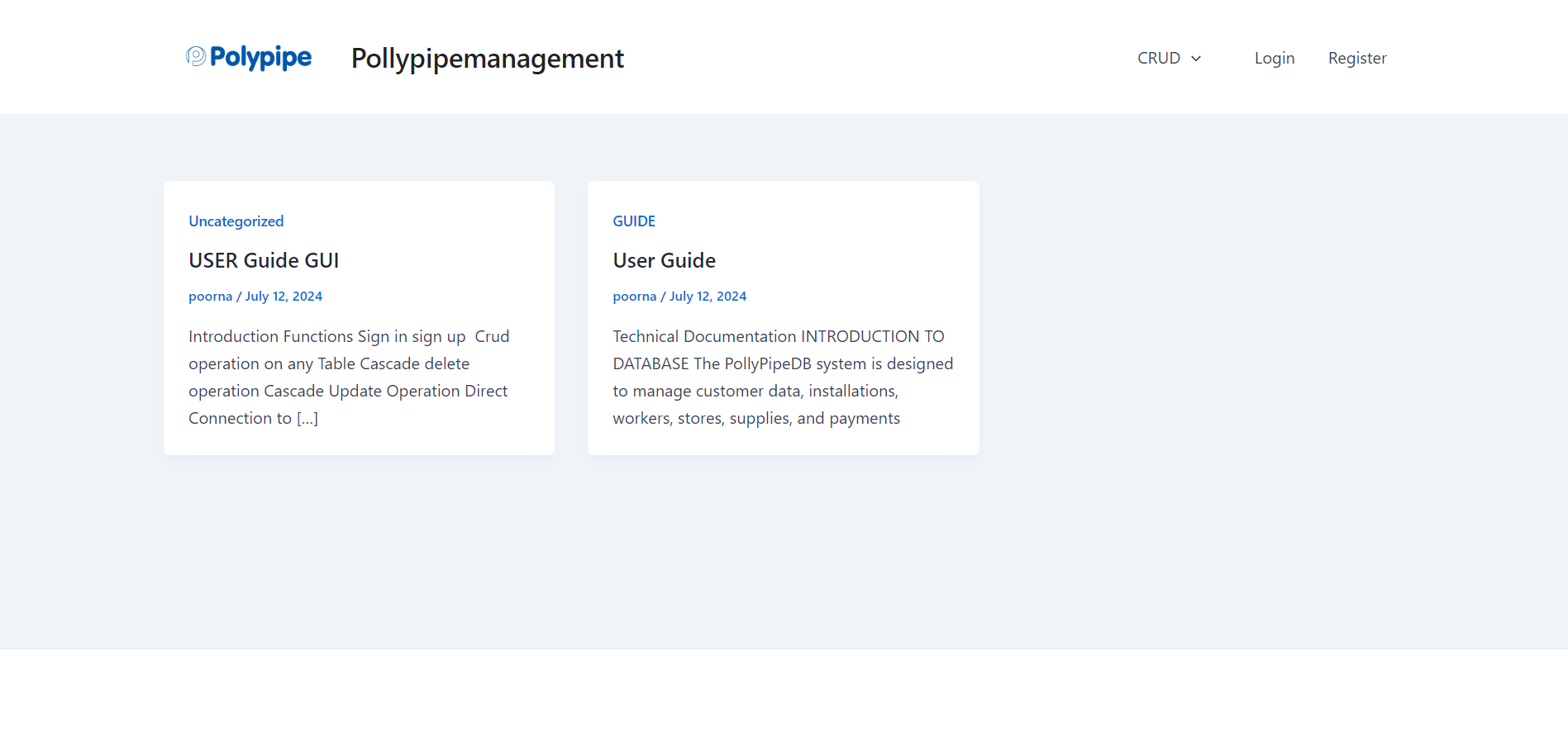


Figure 5:First page without login

Here crud operation and database managements first page can direct to the login sign up and DBM pages

## Sign in /Signup page popup

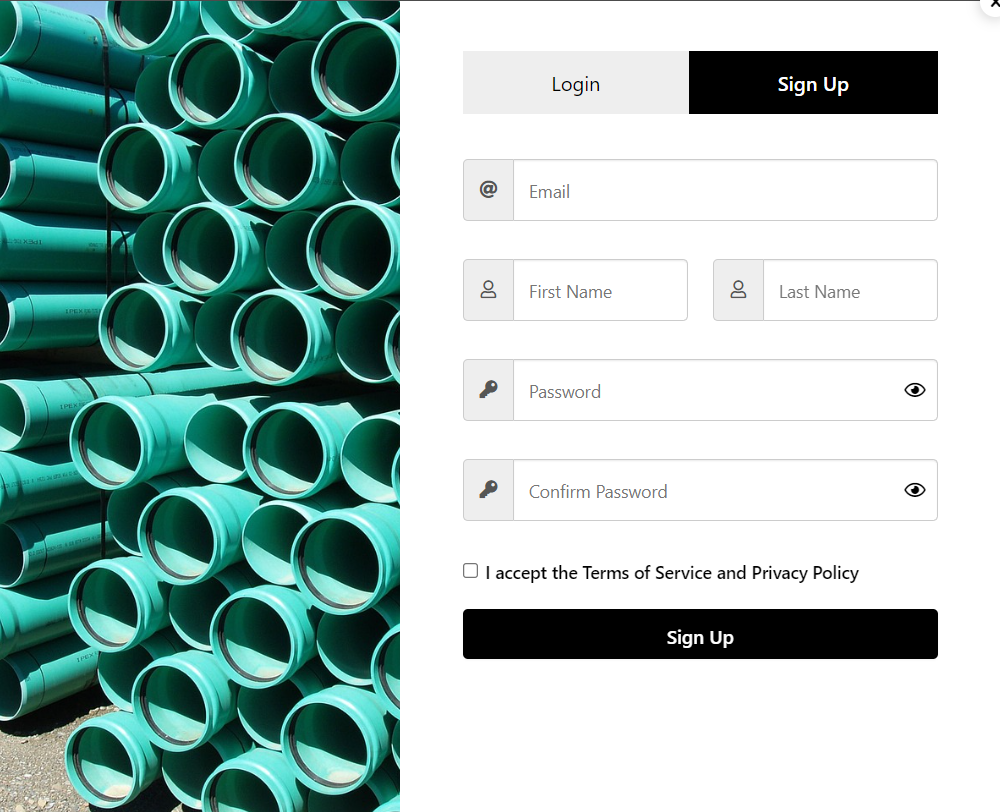
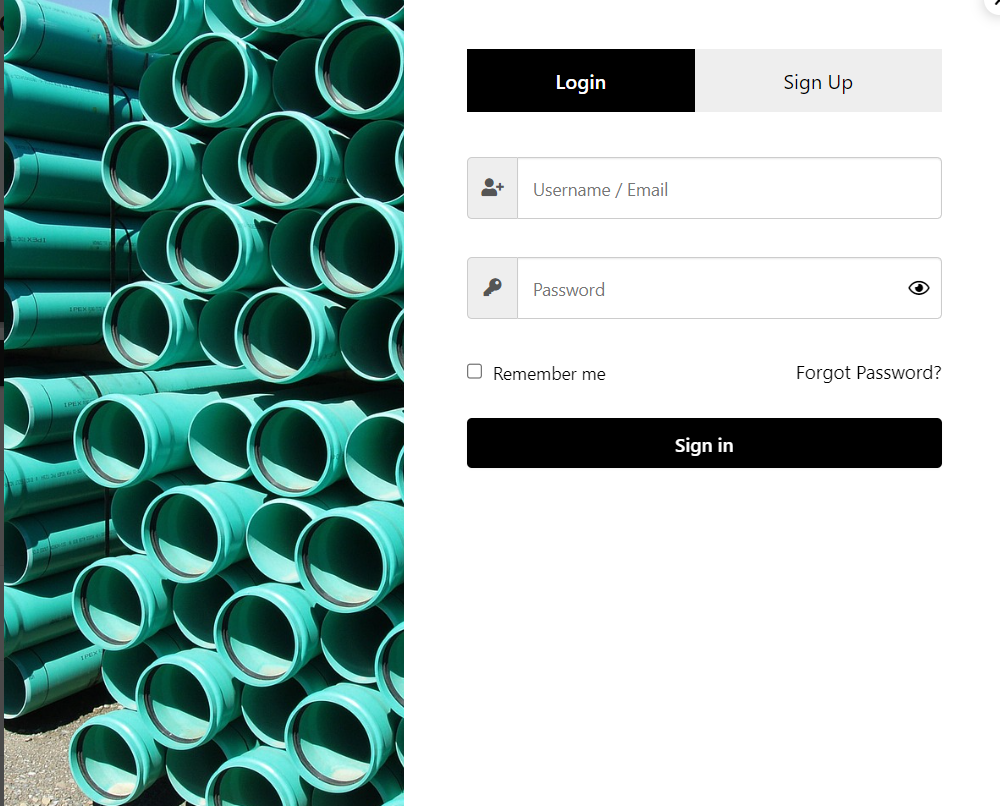


Figure 6:Sign in sign up popup windows

Here is a strong authentication method for the signup and sign in popup windows.

## Menu page

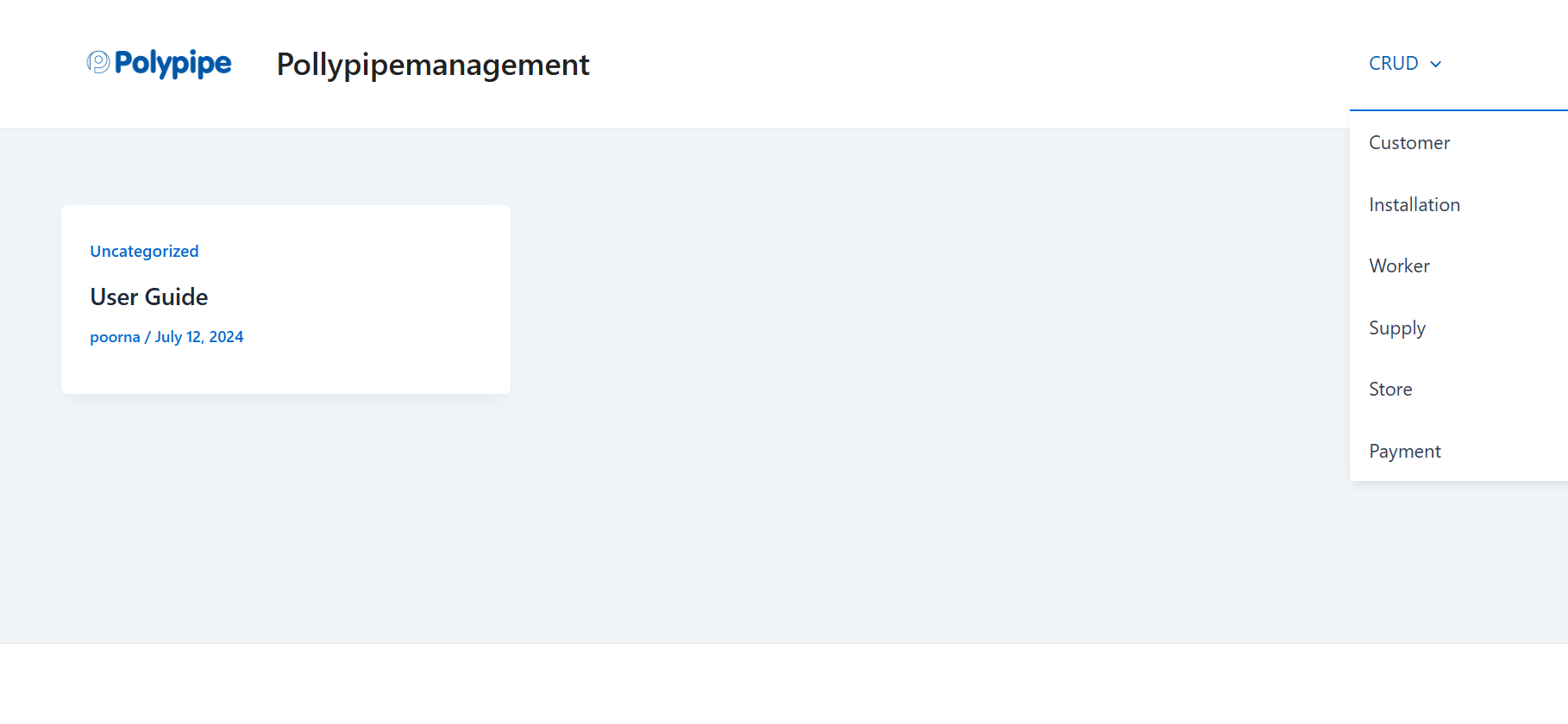
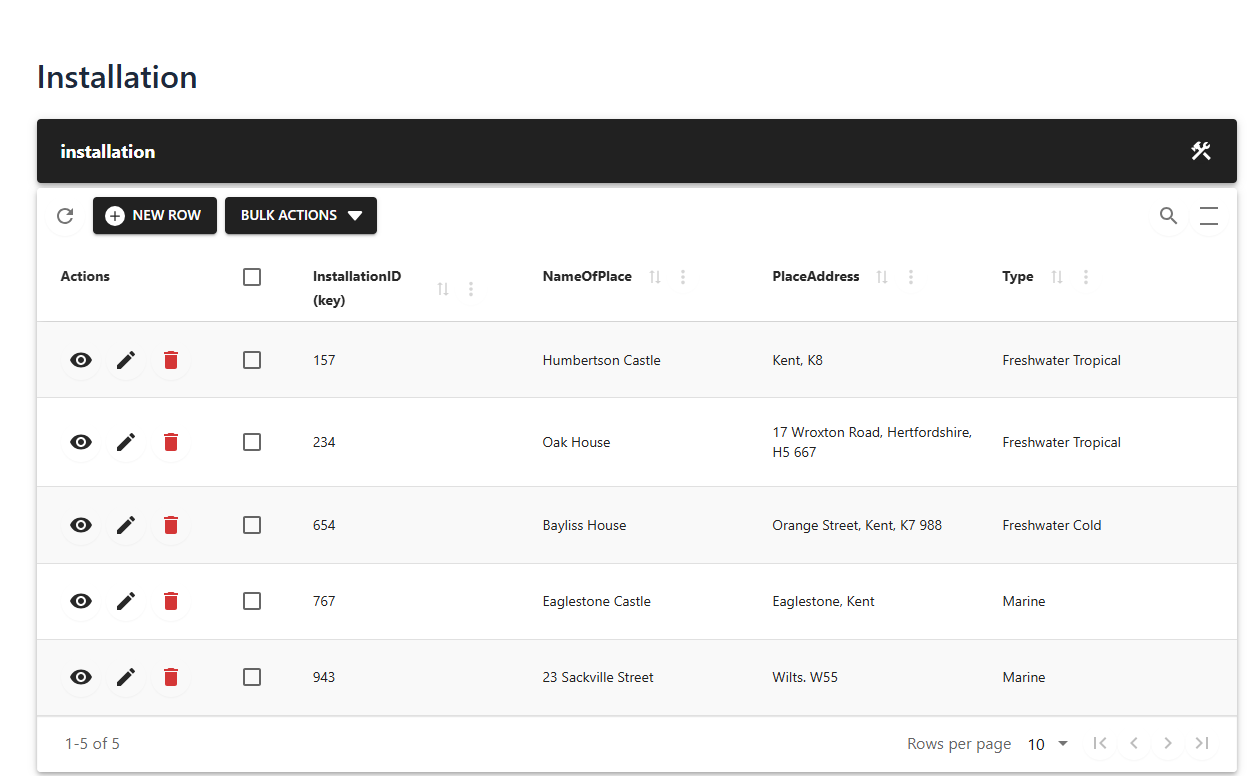
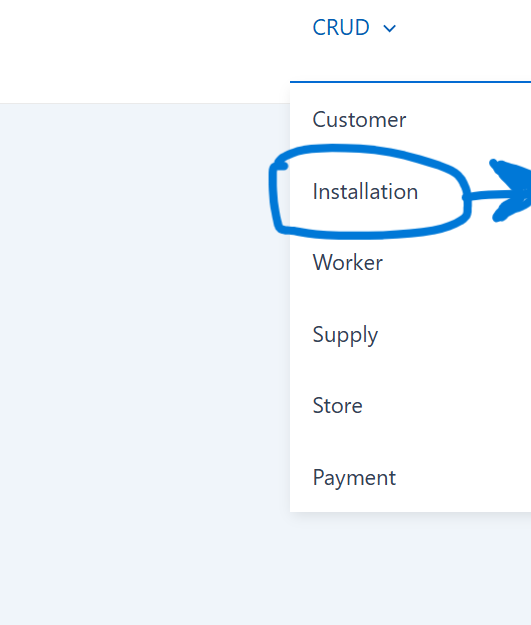
On the Gui there is a Catagory called CRUD on the right upper corner When howering your arrow head over that you can see operation List From that you can Choos whatever you want to Change

Figure 7:Listdown Menu

Figure 8:Working Of List down Menu



## Customer interface (customer page)

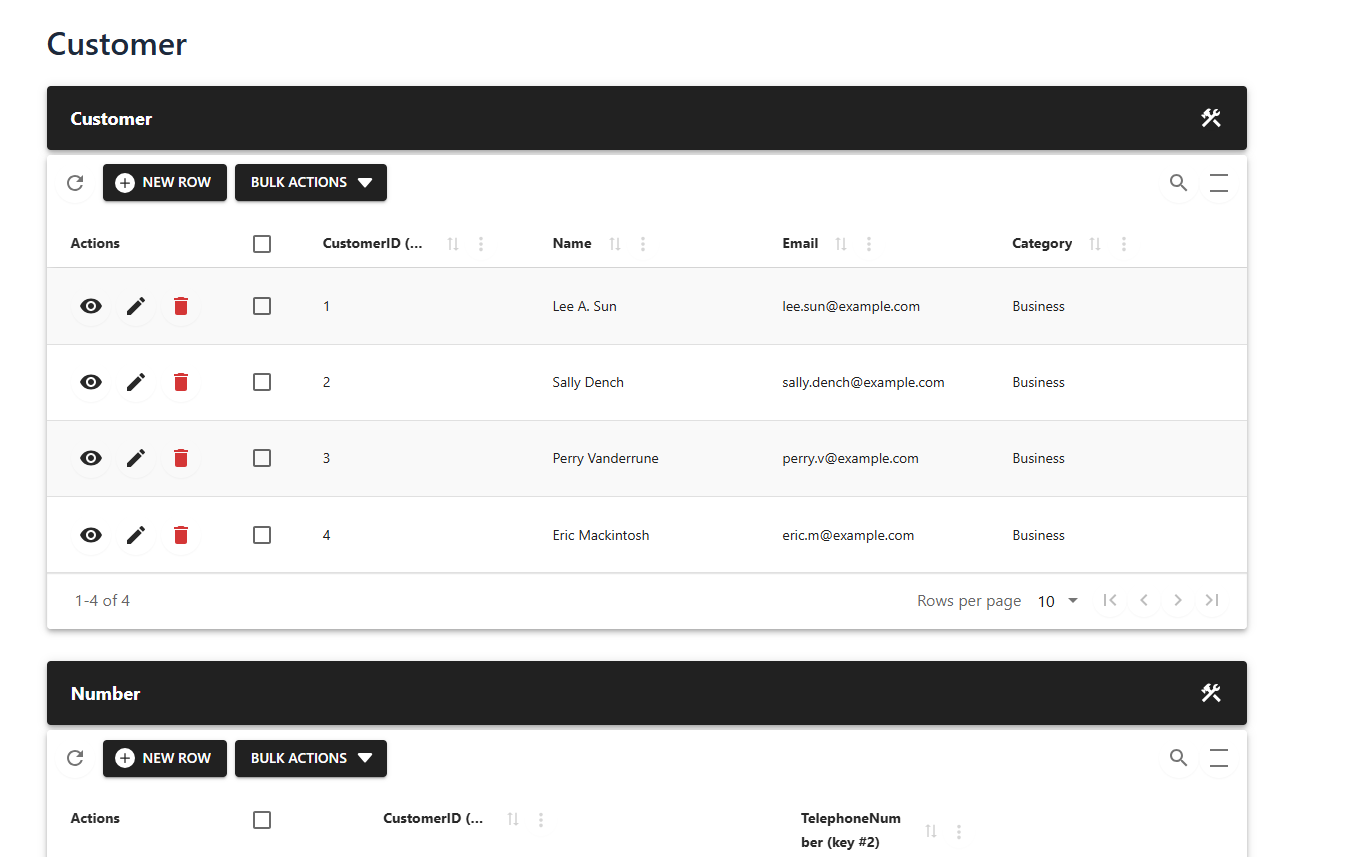
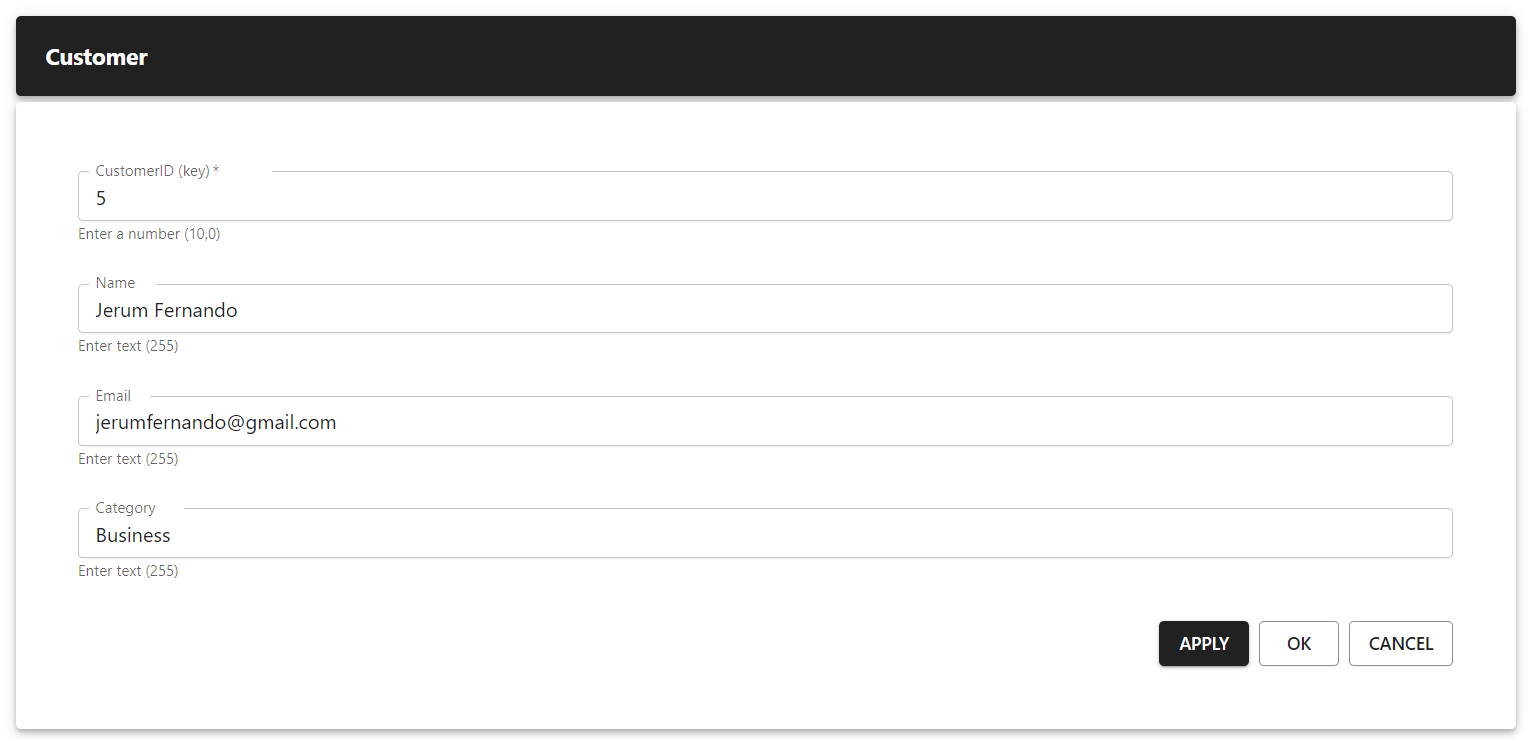


Figure 9:Insert Customer

Figure 10:Customer page

## Supply page

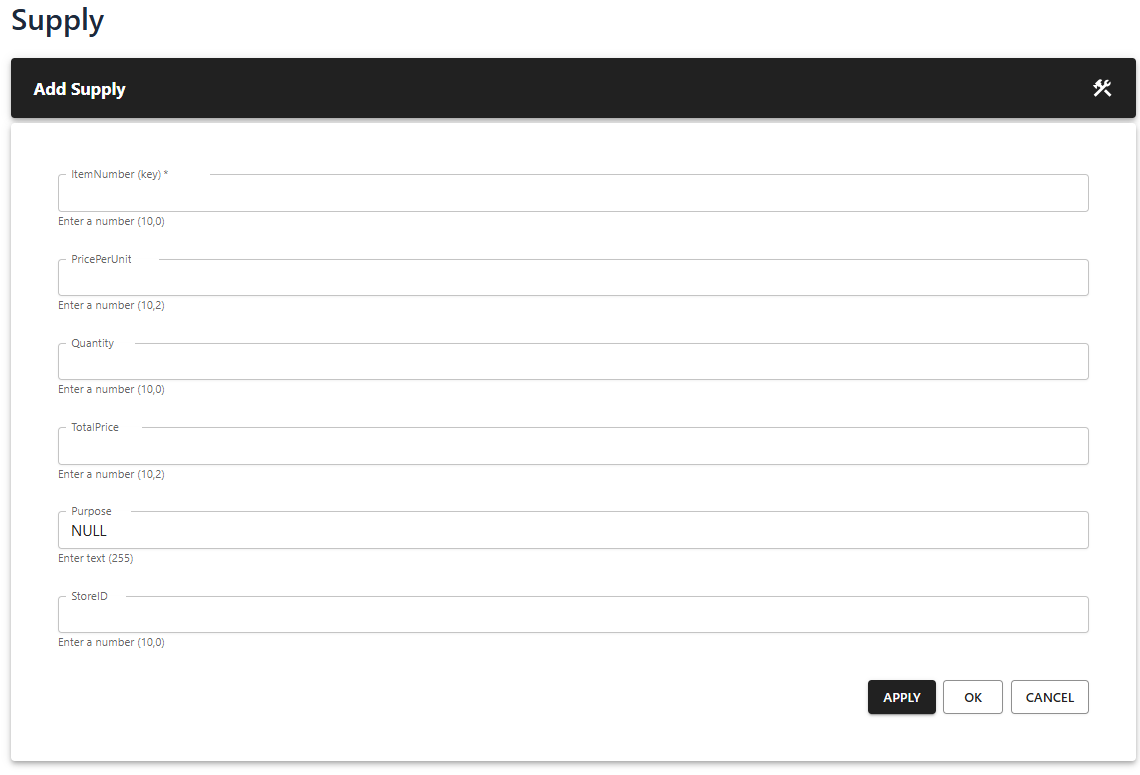


Figure 11:New Supply page

Payments page

Figure 12:New Payment Page

## Store page

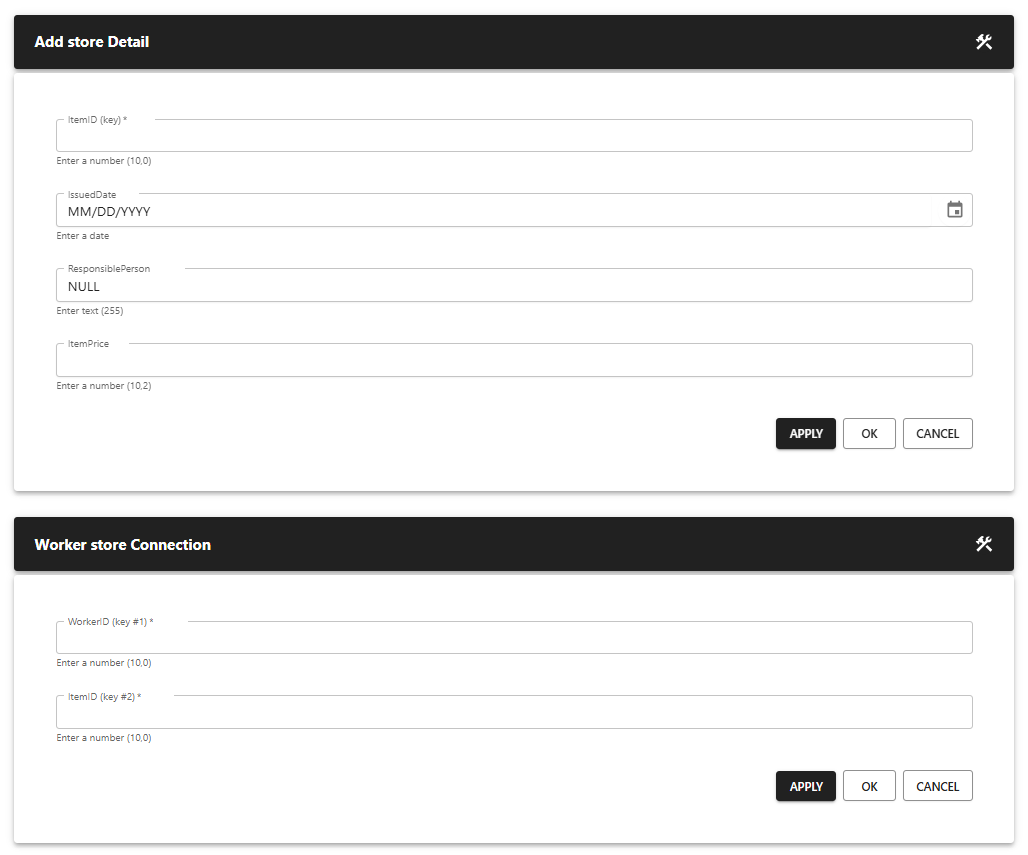


Figure 13:New store Page

## Worker page

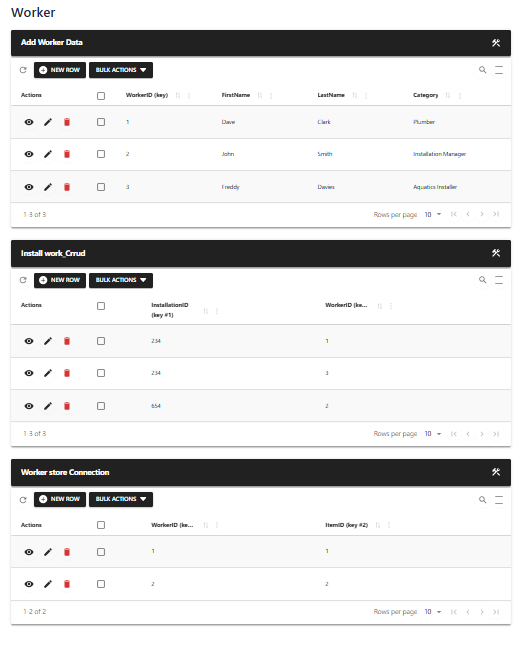


Figure 14:Worker Form Page

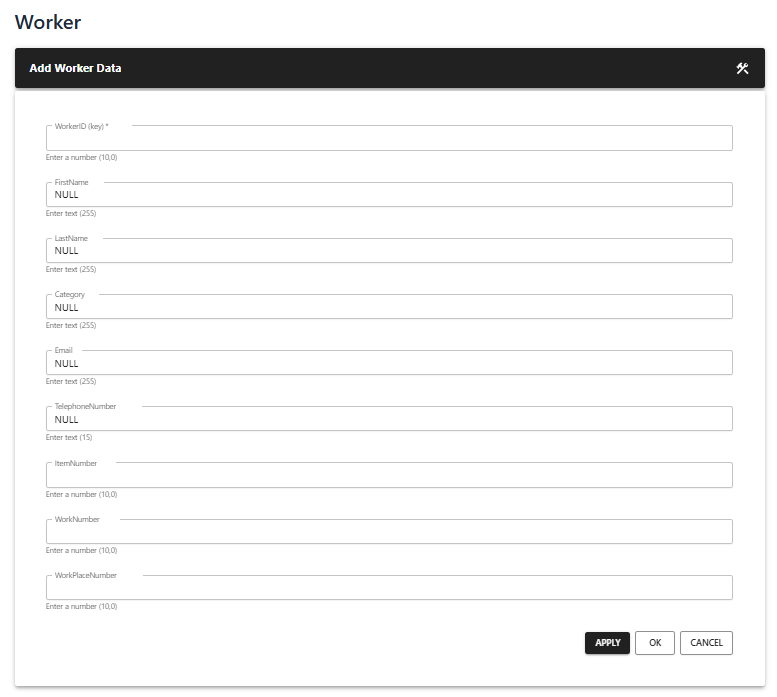


Figure 15:New Worker page

# Validation

Data validation is the process of verifying that the data collected is correct and will be suitable for importation and/or further processing. The nature of validation is categorized by destination constraints or objectives. Data validation is useful in data cleaning and the reduction of errors, cautiously.

Why perform data validation?

Data validation is significant to the processing of data and to proper management of data. There are several types of data validation:There are several types of data validation:

* Constraint validation: Makes sure that data provided must fulfill the constraints like minimum value, maximum value, steps, patterns and required validations.
* Structured validation: Reminds about the legal or valid nature of data.
* Data range validation: Counts the total data and also their numbers and size.
* Code validation: It confirms start with the code of software or websites to check if it runs as required.
* Data type validation: Checked that justified data types which include string, integer, varchar are indeed appropriate.

How validation mechanisms were used for Poly Pipe Company

To ensure a high-quality database system for Poly Pipe Company, various validation types were applied:To ensure a high-quality database system for Poly Pipe Company, various validation types were applied:

* Data Types: For all the data that consisted of alphabets and numbers, the string data type was used while for the numbers; integer data type was used and for the text data varchar type was used.
* Combo Boxes: Applied when defining the option to select, creating a check for fields, which should not contain null values.
* Calculation Methods: Used in recording correct price and other measurable factors in the business environment.

These forms of validation facilitate the achievement of data quality and accuracy within the Poly Pipe Company’s database system, therefore helping in eradicating errors in data processing.

## Examples

### Data Type Validation

* Example: Check that the field called CustomerID in the Customers table is of INTEGER type.
* Implementation: As for the CustomerID column, make it an INT data type in the database schema so that it will recognize only integer values.

## Constraint Validation

* Example: Guarantee that an Email field is specified in the Customers table and is different for each customer.
* Implementation: Remove the duplicate customer entries in the Customers table and then add a CONSTRAINT named UNIQUE to the Email column to avoid this.

## Structured Validation

* Example: Sanitize the format of telephone numbers entered for customers to fit the format required by the company.
* Implementation: After the collection of the telephone number use Regex or custom validation functions to ensure the format is correct (example +1234567890, (123) 456-7890).

## Data Range Validation

* Example: Ensure that the function GetPricePerUnit in the Supplies table range from a given to the other specific one.
* Implementation: Make use of CHECK constraints in the database schema in a way that would force compliance with a Minimum price per unit and a maximum price per unit.

## Code Validation

* Example: This must check and confirm the validity of the installation codes input to the list of allowable codes to be used.
* Implementation: Those installation codes that are valid are put into a lookup table and then a foreign key is to be created to prevent the entry of a code that is not in the lookup table.

## Required Validation

* Example: Therefore, guarantee that the CustomerID is not null while creating a record in the CustomerTelephoneNumbers table.
* Implementation: Modify the CustomerTelephoneNumbers table setting the CustomerID column as NOT NULL in order to ensure that each record to be processed would have a necessary ID of a customer.

## Business Rule Validation

* Example: Make sure that any date mentioned in regard to the installations is not prior to today.
* Implementation: The data type prevents non-anglicized characters from being used for the InstallationDate field, unfortunately an InstallationDate cannot be less than the InstallationDate, so for logic use a trigger or stored procedure to check InstallationDate is greater than or equal to today before inserting or updating records in the Installations table.

When these validation techniques are applied, it eliminates data entry mistakes and improves the efficiency of the Poly Pipe Company’s database system for enabling accurate business operations and customer satisfaction.

## Validation and Verification (PP.Pankaj, 2020)

Table 12:Validation and verification according to various aspects

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Validation** | **Verification** |
| Type | Dynamic testing | Static testing |
| Execution | Includes execution of the code | Does not include execution of the code |
| Methods | Black Box Testing, White Box Testing, | Reviews, walkthroughs, inspections, desk-checking |
|  | Non-functional testing |  |
| Purpose | Checks if software meets specifications | Checks if software conforms to specifications |
| Bugs Found | Finds bugs that couldn't be found by | Finds bugs in early stages of development |
|  | Verification |  |
| Goal | Ensures actual product functionality | Ensures software architecture and specifications |
| Sequence | Comes after verification | Comes before validation |

## Relationship of User and System

The user communicates with the system which is the software or electronic means that satisfy the user’s requirement. This is important relationship when it comes to the optimal running of an organization. Specifically, system developers introduced a database system to perform the management of day-to-day operations in the existence of this company. I have also explained with much detail now how this system meets the user side needs, while also satisfying the system side needs

# Assess the Quality of System and User Requirements

## User Requirements

* **Identify various categories that could be unique in the records. :**Composed from the “Menu” page. The writer managed to identify all the unique categories and distribute the sections therefore.
* **Include identification numbers suitable for the customers’ identification and their needs (IDs, Telephone numbers or Email address). :** Finished from each page with the help of “Primary keys”. IDs, TP numbers or Email was declared as a primary key for uniqueness in a writer.
* **It is also necessary record some particulars about the needs of the customers.: Establishing** a record register for the database. Writer typed all the records required for the later assessment and services into the database tables.
* **Customers can request installations :**Emails and call are some ways that can be used by customers to request installations. One TP call or one TP email message can book a service for a timeframe that is appropriate for the subscriber.
* **That isolating clients for individual installations is essential.:** Service option was split into categories for better and relevant service in the Installation section. Writer segregated the installation interface and the functions in a proper way.
* **There is a need to record other requirements and services. :** Concerning other requirements and services of users, they can maintain records using the Database System.

## System Requirements:

* **Should be user friendly :** This has a simple intended interface and is built such that the general public can utilize it easily. A user interacts with GUIs to input, modify delete and or search data.
* **It should not take much space in a server A well designed and implemented system should not consume a large amount of space in the servers. :** System size is only 150Mb, although it will run on any Media, Music or Cinema PC.
* **Should be attractive :** Task images which are readily associated with the sub tasks need to be displayed and the interfaces to be made as beautiful and as simple as possible.
* **Should be low cost :** Measures toward the enhancement of system costs were adequately implemented.

# Database

## Creating database

### Quarries to create

CREATE DATABASE PollyPipeDB;

USE PollyPipeDB;

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(255),

Email VARCHAR(255),

Category VARCHAR(255)

);

CREATE TABLE CustomerTelephoneNumbers (

CustomerID INT,

TelephoneNumber VARCHAR(15),

PRIMARY KEY (CustomerID, TelephoneNumber),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Installations (

InstallationID INT PRIMARY KEY,

NameOfPlace VARCHAR(255),

PlaceAddress VARCHAR(255),

Type VARCHAR(255),

CustomerID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Workers (

WorkerID INT PRIMARY KEY,

FirstName VARCHAR(255),

LastName VARCHAR(255),

Category VARCHAR(255),

Email VARCHAR(255),

TelephoneNumber VARCHAR(15),

ItemNumber INT,

WorkNumber INT,

WorkPlaceNumber INT

);

CREATE TABLE Stores (

ItemID INT PRIMARY KEY,

IssuedDate DATE,

ResponsiblePerson VARCHAR(255),

ItemPrice DECIMAL(10, 2)

);

CREATE TABLE Supplies (

ItemNumber INT PRIMARY KEY,

PricePerUnit DECIMAL(10, 2),

Quantity INT,

TotalPrice DECIMAL(10, 2) AS (PricePerUnit \* Quantity) STORED,

Purpose VARCHAR(255),

StoreID INT

);

CREATE TABLE Payments (

BillNumber INT PRIMARY KEY,

BuyerName VARCHAR(255),

Category VARCHAR(255),

DatePay DATE,

CustomerID INT

);

CREATE TABLE InstallationWorkers (

InstallationID INT,

WorkerID INT,

PRIMARY KEY (InstallationID, WorkerID)

);

CREATE TABLE WorkerStores (

WorkerID INT,

ItemID INT,

PRIMARY KEY (WorkerID, ItemID)

);

### Quarries to realationships

-- Bellow here shows the relationships

ALTER TABLE CustomerTelephoneNumbers

ADD CONSTRAINT FK\_CustomerTelephones\_Customers

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);

ALTER TABLE Installations

ADD CONSTRAINT FK\_Installations\_Customers

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);

ALTER TABLE Workers

ADD CONSTRAINT FK\_Workers\_Supplies

FOREIGN KEY (ItemNumber) REFERENCES Supplies(ItemNumber),

ADD CONSTRAINT FK\_Workers\_Installations

FOREIGN KEY (WorkNumber) REFERENCES Installations(InstallationID);

ALTER TABLE Supplies

ADD CONSTRAINT FK\_Supplies\_Stores

FOREIGN KEY (StoreID) REFERENCES Stores(ItemID);

ALTER TABLE Payments

ADD CONSTRAINT FK\_Payments\_Customers

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);

ALTER TABLE InstallationWorkers

ADD CONSTRAINT FK\_InstallationWorkers\_Installations

FOREIGN KEY (InstallationID) REFERENCES Installations(InstallationID),

ADD CONSTRAINT FK\_InstallationWorkers\_Workers

FOREIGN KEY (WorkerID) REFERENCES Workers(WorkerID);

ALTER TABLE WorkerStores

ADD CONSTRAINT FK\_WorkerStores\_Workers

FOREIGN KEY (WorkerID) REFERENCES Workers(WorkerID),

ADD CONSTRAINT FK\_WorkerStores\_Stores

FOREIGN KEY (ItemID) REFERENCES Stores(ItemID);

### Cascade operation Quarries

USE PollyPipeDB;

--

ALTER TABLE CustomerTelephoneNumbers

DROP CONSTRAINT FK\_CustomerTelephones\_Customers;

ALTER TABLE CustomerTelephoneNumbers

ADD CONSTRAINT FK\_CustomerTelephones\_Customers

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

ON DELETE CASCADE

ON UPDATE CASCADE;

--

ALTER TABLE Installations

DROP CONSTRAINT FK\_Installations\_Customers;

ALTER TABLE Installations

ADD CONSTRAINT FK\_Installations\_Customers

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

ON DELETE CASCADE

ON UPDATE CASCADE;

--

ALTER TABLE Workers

DROP CONSTRAINT FK\_Workers\_Supplies,

DROP CONSTRAINT FK\_Workers\_Installations;

ALTER TABLE Workers

ADD CONSTRAINT FK\_Workers\_Supplies

FOREIGN KEY (ItemNumber) REFERENCES Supplies(ItemNumber)

ON DELETE CASCADE

ON UPDATE CASCADE,

ADD CONSTRAINT FK\_Workers\_Installations

FOREIGN KEY (WorkNumber) REFERENCES Installations(InstallationID)

ON DELETE CASCADE

ON UPDATE CASCADE;

--

ALTER TABLE Supplies

DROP CONSTRAINT FK\_Supplies\_Stores;

ALTER TABLE Supplies

ADD CONSTRAINT FK\_Supplies\_Stores

FOREIGN KEY (StoreID) REFERENCES Stores(ItemID)

ON DELETE CASCADE

ON UPDATE CASCADE;

--

ALTER TABLE Payments

DROP CONSTRAINT FK\_Payments\_Customers;

ALTER TABLE Payments

ADD CONSTRAINT FK\_Payments\_Customers

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

ON DELETE CASCADE

ON UPDATE CASCADE;

--

ALTER TABLE InstallationWorkers

DROP CONSTRAINT FK\_InstallationWorkers\_Installations,

DROP CONSTRAINT FK\_InstallationWorkers\_Workers;

ALTER TABLE InstallationWorkers

ADD CONSTRAINT FK\_InstallationWorkers\_Installations

FOREIGN KEY (InstallationID) REFERENCES Installations(InstallationID)

ON DELETE CASCADE

ON UPDATE CASCADE,

ADD CONSTRAINT FK\_InstallationWorkers\_Workers

FOREIGN KEY (WorkerID) REFERENCES Workers(WorkerID)

ON DELETE CASCADE

ON UPDATE CASCADE;

--

ALTER TABLE WorkerStores

DROP CONSTRAINT FK\_WorkerStores\_Workers,

DROP CONSTRAINT FK\_WorkerStores\_Stores;

-- cascade

ALTER TABLE WorkerStores

ADD CONSTRAINT FK\_WorkerStores\_Workers

FOREIGN KEY (WorkerID) REFERENCES Workers(WorkerID)

ON DELETE CASCADE

ON UPDATE CASCADE,

ADD CONSTRAINT FK\_WorkerStores\_Stores

FOREIGN KEY (ItemID) REFERENCES Stores(ItemID)

ON DELETE CASCADE

ON UPDATE CASCADE;

### Data insertion

#### Customer data

USE PollyPipeDB;

INSERT INTO Customers (CustomerID, Name, Email, Category) VALUES

(1, 'Lee A. Sun', 'lee.sun@example.com', 'Business'),

(2, 'Sally Dench', 'sally.dench@example.com', 'Business'),

(3, 'Perry Vanderrune', 'perry.v@example.com', 'Business'),

(4, 'Eric Mackintosh', 'eric.m@example.com', 'Business');

#### Installation Data

USE PollyPipeDB;

INSERT INTO Installations (InstallationID, NameOfPlace, PlaceAddress, Type, CustomerID) VALUES

(234, 'Oak House', '17 Wroxton Road, Hertfordshire, H5 667', 'Freshwater Tropical', 1),

(654, 'Bayliss House', 'Orange Street, Kent, K7 988', 'Freshwater Cold', 2),

(767, 'Eaglestone Castle', 'Eaglestone, Kent', 'Marine', 3),

(943, '23 Sackville Street', 'Wilts. W55', 'Marine', 4),

(157, 'Humbertson Castle', 'Kent, K8', 'Freshwater Tropical', 3);

#### Installation Workers Data

USE PollyPipeDB;

INSERT INTO InstallationWorkers (InstallationID, WorkerID) VALUES

(234, 1),

(654, 2),

(234, 3);

#### Payments Data

USE PollyPipeDB;

INSERT INTO Payments (BillNumber, BuyerName, Category, DatePay, CustomerID) VALUES

(1, 'Lee A. Sun', 'Installation', '2024-01-01', 1),

(2, 'Sally Dench', 'Installation', '2024-01-02', 2);

#### Stores Data

USE PollyPipeDB;

INSERT INTO Stores (ItemID, IssuedDate, ResponsiblePerson, ItemPrice) VALUES

(1, '2024-01-01', 'John Doe', 100.00),

(2, '2024-01-02', 'Jane Doe', 200.00);

#### Supplies Data

USE PollyPipeDB;

INSERT INTO Supplies (ItemNumber, PricePerUnit, Quantity, Purpose, StoreID) VALUES

(1, 50.00, 6, 'Installation', 1),

(2, 100.00, 4, 'Installation', 2);

#### Workers Data

USE PollyPipeDB;

INSERT INTO Workers (WorkerID, FirstName, LastName, Category, Email, TelephoneNumber, ItemNumber, WorkNumber, WorkPlaceNumber) VALUES

(1, 'Dave', 'Clark', 'Plumber', 'dave.clark@example.com', '0123456793', NULL, 234, 1),

(2, 'John', 'Smith', 'Installation Manager', 'john.smith@example.com', '0123456794', NULL, 654, 2),

(3, 'Freddy', 'Davies', 'Aquatics Installer', 'freddy.davies@example.com', '0123456795', NULL, 234, 1);

#### Worker Store data

USE PollyPipeDB;

INSERT INTO WorkerStores (WorkerID, ItemID) VALUES

(1, 1),

(2, 2);

## Checking for Proper operation

### Customer tables

Figure 16:Customer table database

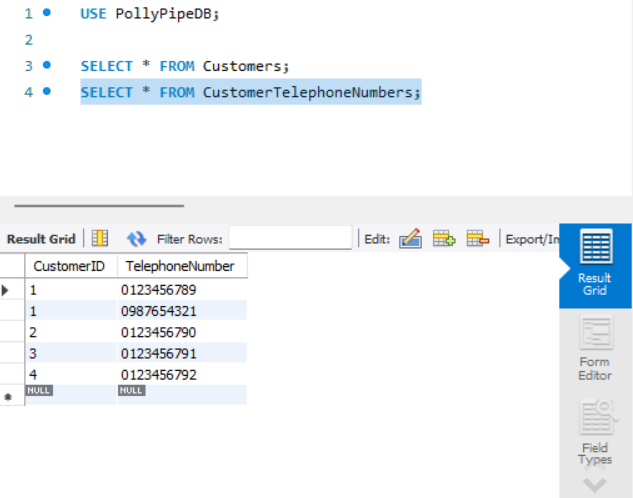


Figure 17:Customer Number table check

### Installation Table

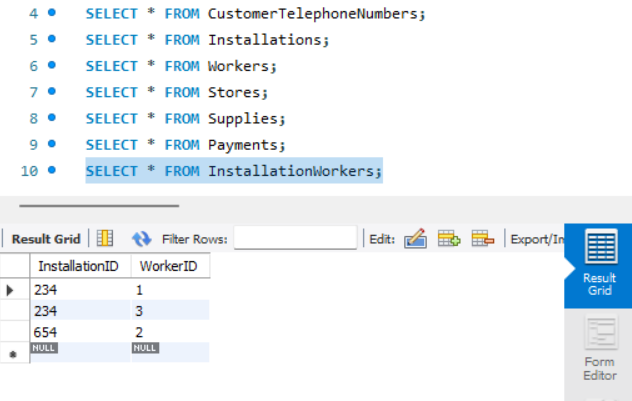


Figure 18:Check Installation workers Table

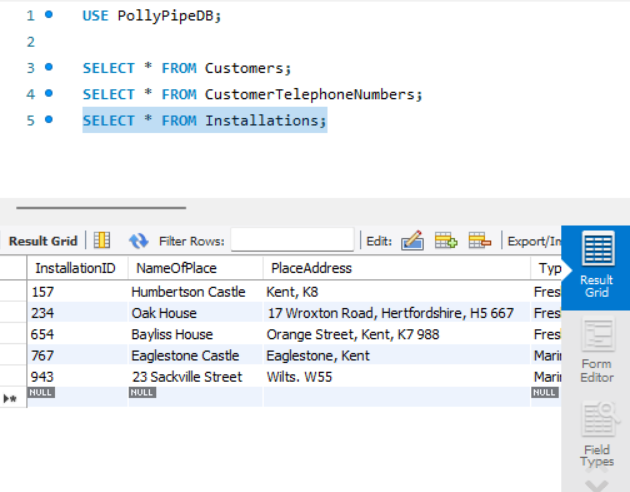


Figure 19:Check installation Table

### Payment Table

Figure 20:Check Payment table

### Stores Table

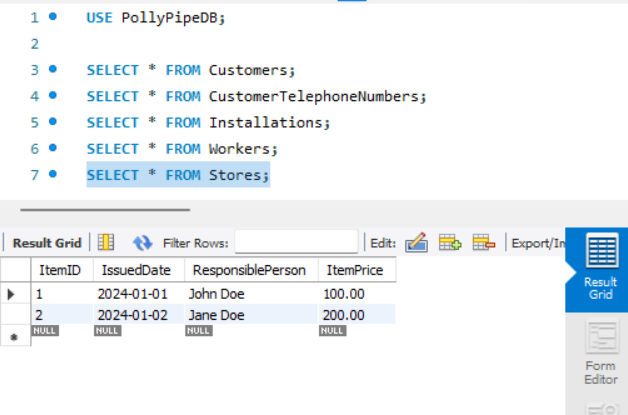


Figure 21:Check store table

### Supplies Table

Figure 22:Check Supplies Table

### Workers Table

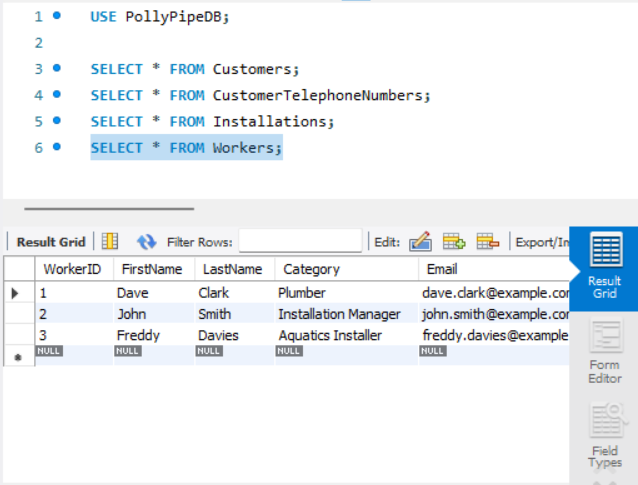


Figure 23:Check Workers Table

## Database operations

### Select Operation

Figure 24:Select Operation

Retrieve all records from the Customers table.

### Update Operation

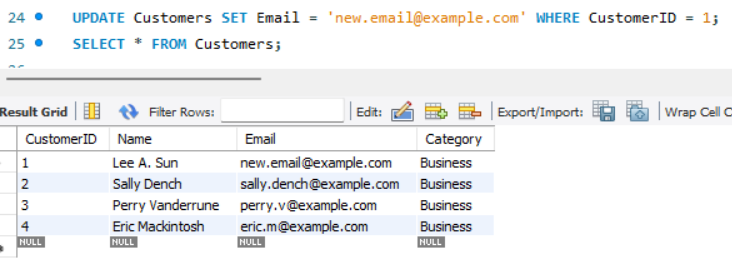
Update the email of a customer with CustomerID 1.

Figure 25:Update Operation

### Where Operation

Select all workers who are 'Aquatics installers'

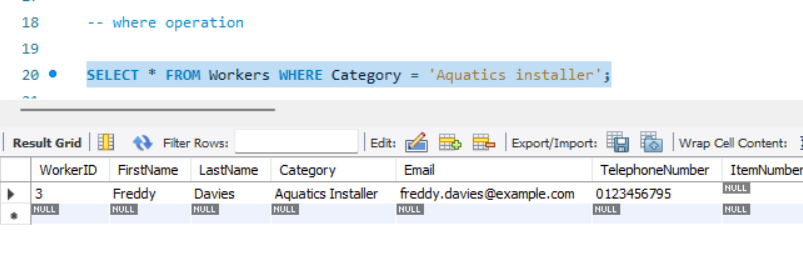


Figure 26:Where Operation

### Between Operation

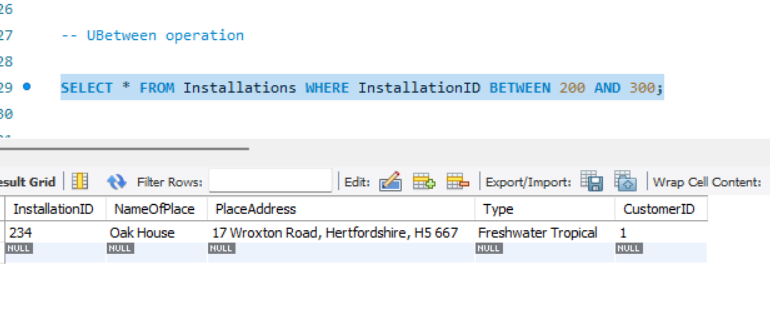
Select all installations with InstallationID between 200 and 300.

Figure 27:Between Operation

### IN Operation

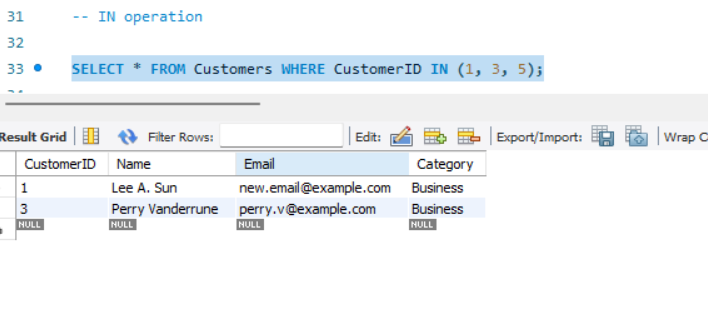
Select all customers whose CustomerID is either 1, 3, or 5.

Figure 28:IN operation

### GROUP BY Operation

Group supplies by Purpose and count the number of supplies for each purpose

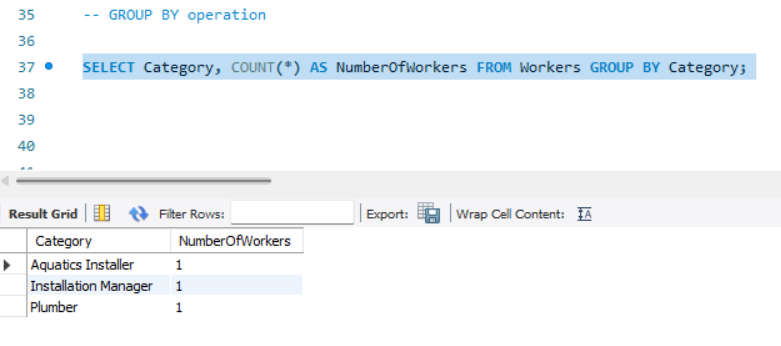


Figure 29:Group By Operation

### ORDER BY Operation

Select all installations ordered by PlaceAddress in ascending order.

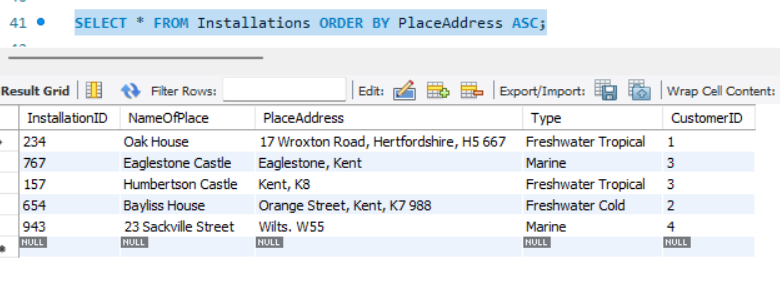


Figure 30:Order By Operation

### Having Operation

Select purposes from the Supplies table where the total quantity is greater than 100, using GROUP BY and HAVING.

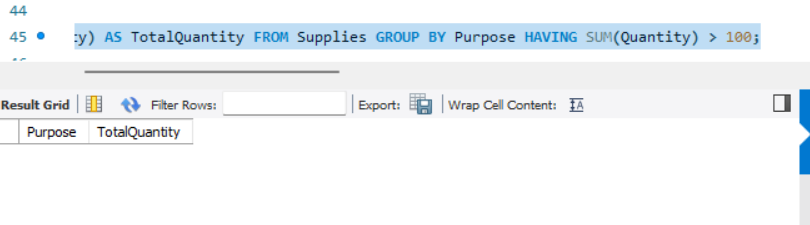


Figure 31:Having Operation

# Test Plan

The test plan is to Add delete and update a customer to see whether the intended operations are to be implemented successfully

Hardware and software testing means checking all the parameters within the existing system DB testing confirms that nothing is wrong with the database within the enclosed testing environment known as back-end or data testing. It entails checking the consistency of any database structures, tables, triggers, and assessing whether a system fulfills the customer’s needs profusely (reqtest, 2017). Testers build up SQL commands to assert durability, organization, and characteristics that affect the application’s performance for the tests. Working in tight cooperation with developers, database testers are responsible for assuring compliance of business rules with the database structure and functioning.

The necessity of database testing is justified for the following reasons. First and foremost, it protects the data from being altered, which usually could lead to financial losses. It also guards its users against security attacks, network themselves, power outage and data corruptions thus maintaining reliable data storage (reqtest, 2017). While conducting the database testing, mistakes and inconsistencies are acknowledged and eliminated, in this way the data integrity is preserved and the formats of data and associations between the entities in the database get verified.

Key areas to test in database testing include the CRUD operations: It is an acronym for the operations that can be carried out on the database which are; Creation, Reading, Updating, and Deletion. It checks that instances of records create, read, update and delete are correct to ensure data integrity in its multiple uses (reqtest, 2017). The performance of these tests is usually carried out either through a GUI layer of the application or directly through SQL queries in a DBMS to evaluate the correctness of all data operations relative to defined specifications and requirements.

Conclusively, database testing is an essential technique in ensuring the quality, correctness, and security of the database. Therefore, it facilitates accuracy and integrity of stored data so as to enhance the functionality and operation of applications that rely on the database.

Testing databases is, therefore, a process of the assessment of operation, structure, and interaction of data within the database. Here’s a structured approach to testing databases effectively:Here’s a structured approach to testing databases effectively:

## Manual Database Testing Process

### Prepare Your Test Environment:

Organize a specific environment for testing simulating a production environment as much as possible. Solving various tasks in structuring the database instances, tables, schema, and other useful data are also included here.

### Craft Test Cases:

Create test cases which would effectively address all aspects of the database usage, CRUD operations, data integrity and validation rules, as well as business logics that should be performed on the database. Finally, each test case should be specific, devoted to different situations and the program’s margins.

### Execute Test Cases:

Manually run through the test cases that were planned and documented earlier. This includes working with SQL commands on the database or by using database utilities in tasks such as inserting data, modifying records, querying database and deleting records among others.

### Verify Results:

Ensure that the results obtained on completion of each test case correspond to the expected results. Make certain that the data is inserted correctly, and updated records have the correct change and that the queries will present the correct result without errors and inconsistencies.

### Document and Report:

After testing provides detailed records with the areas of concern or problems that were met during testing. Prepare reports that highlight them: steps that were taken, actual results, expected results, and any variations that have been noted.

## Automated Database Testing Process

DBMS testing can be made more automated which has benefits of denying the difficulties of manual testing, and improving the possibilities of repeatability and reliability. The steps involved are similar to automating tests for application code.

### Set Up Automated Test Environment

Install test scripts and arrangements which are compatible with database testing. Selenium, TestNG and there are specific tools for testing databases including the type of database and the vendor who provides the database.

### Develop Automated Test Scripts:

Record the targets for automated testing with SQL queries or any specific automation scripts of the database. These scripts should include as many test scenarios as possible ranging from data manipulation, stored procedures, triggers and transactions on the database.

### Execute Tests:

Execute the automated test scripts with the help of a test runner or the used automation framework. Supervise the execution phases to check whether all related tests have been accomplished in the proper manner.

### Validate Results:

Compare the results of automatized testing with original success criteria and expected results. Log files and reports which are created by the automation tool should be reviewed to determine such failure or problem.

### Review and Report:

Check the entire outcome of the automated tests and record all the mistakes and faults that you come across. Produce accurate coverage, status and results of each tests executed, as well as, the areas that need improvement.

## Database Testing Tools

Therefore, the selection of the DBT tool depends on the approach whether manual, automated, or both, DBMS used- SQL or No-SQL, and DBMS vendor like MySQL, MSSQL, ORACLE etc. Some popular database testing tools include:Some popular database testing tools include:

* DBUnit: The effective unit testing framework for any Java application that came under the category of JUnit extension for the database-driven unit testing.
* JUnit: Java oriented unit testing framework which can be used to writing database tests.
* Selenium: Mainly for GUI tests but if connected to databases it can be used in end-to-end testing.
* Postman: Good for API testing for instance testing of database APIs or API integration.
* Apache JMeter: Includes the options for carrying out load testing such as the database load testing and performance testing.

## Types of Database Testing

Apart from functional testing (CRUD operations, data integrity checks), other types of database testing include:Apart from functional testing (CRUD operations, data integrity checks), other types of database testing include:

* Performance Testing: Evaluating the database performance which is essential while under performance testing to determine its scalability and response time.
* Security Testing: Testing of access controls, encryption and evaluation of the security vulnerabilities of the database and the system.
* Concurrency Testing: The last criterion means that the database must work properly when multiple transactions are performed at the same time without data loss or data corruption.
* Backup and Recovery Testing: Invoking the backup and restore strategies of the database to check on data recoverability in the event of failures or disasters.

If the database testers adhere to structured testing processes and use the right tools, it would be possible to guarantee reliably secured and efficient database systems that would in a way complement the quality of those applications which depend on the databases.

## System Testing Overview

### System Testing Definition:

This type of testing is also referred to as system-level testing or system integration testing, it is the last phase of the software development process and one of the most important because the entire integrated system or application is tested to verify that it will work correctly as a whole. It checks the system against certain requirements and evaluates the system’s response in various operations.

### Types of System Testing

#### Scalability Testing:

* Purpose: This type of testing shows how the database system can grow with load, data, and transactions added to the system.
* Poly Pipe Application: Making sure that the future needs for the number of records, installations, and transactions of the customers can be served optimally without compromising the performance or stability of the system.

#### Reliability Testing:

* Purpose: The main concept of reliability testing is to check how the system behaves and functions under diverse conditions during a long time.
* Poly Pipe Application: Ensuring that the database system consistently saves and efficiently retrieves customers’ information as well as safeguarding the information from either deliberate or accidental alteration.

#### Documentation Testing:

* Purpose: Documentation testing checks to ascertain whether all the documentation about a system including the manuals which the users will have to use, those used for installation of the system, and any technical manuals are up to date and include all details adequately.
* Poly Pipe Application: Using the formatting process by revisiting the documentation to assess whether the database has been described in the right manner; including its schema, data relations and the usage instructions for the administrators and other users.

#### Security Testing:

* Purpose: Security testing involves the discovery of flaws in the measures put in place to secure the database system; the access rights, encryption, authentication, and protection from malicious activities.
* Poly Pipe Application: Performing system checks that prove that the customer data is well protected from unauthorized access, only their personnel is allowed access to the database and the company database system is safe from data breaches.

#### Graphical User Interface (GUI) Testing

* Purpose: GUI testing focuses on the functionality of the overall look and feel of the interface features as well as their efficiency in meeting the set requirements as well as users’ predicted impressions.
* Poly Pipe Application: You will want to confirm whether or not the database management interface is easy to use so that CRUD and data query/retrieval tasks can be performed by administrators and users in an efficient manner.

#### Exception Handling Testing:

* Purpose: Test exception focuses on determining how well the database system can respond to events and conditions that are unexpected and can corrupt data or create instability in the system.
* Poly Pipe Application: Including checking the database system’s behavior in case of input of erroneous data, network or database connectivity problems etc, as to avoid data corruptions and to determine the stability of the system.

#### Volume Testing:

* Purpose: Volume test assesses the abilities of the database system to handle huge populations of data, number of transactions, and the total number of users concurrently.
* Poly Pipe Application: Evaluating how optimally the database system handles the large number customers’ record, installation requests and transactions data without lag or slow system processing.

#### Stress Testing:

* Purpose: Stress testing involves exposing the database system to conditions of high loads, concurrency or limited resources to determine areas of poor performance or sign’s of system failure.
* Poly Pipe Application: Challenging the target database system to various high I/O conditions to check for its reliability and stability when under testing pressure.

### Justification for System Testing

The techniques used by the writer for testing Poly Pipe Company’ database system includes systematic testing of the system. Specifically, when implementing these testing types, it was aimed at the condition when the database system satisfies all the functional and non-functional requirements, works stably in different conditions, and provides secure and convenient work for administrators and users.

Thus, every kind of system testing is aimed at checking and verifying various aspects of the database system, such as functionality, performance, security, and usability. Thus, in the course of systematic testing of these components, potential problems and risks are not only detected at the stage of application development but also corrected and improved.

Thus, system testing can be seen as crucial for both improving user satisfaction and increasing the level of reliability and performance of the database system as a whole, which in turn decreases operational risks and proves the system’s efficiency in the context of Poly Pipe Company’s business processes execution.

## Create Operation using GUI

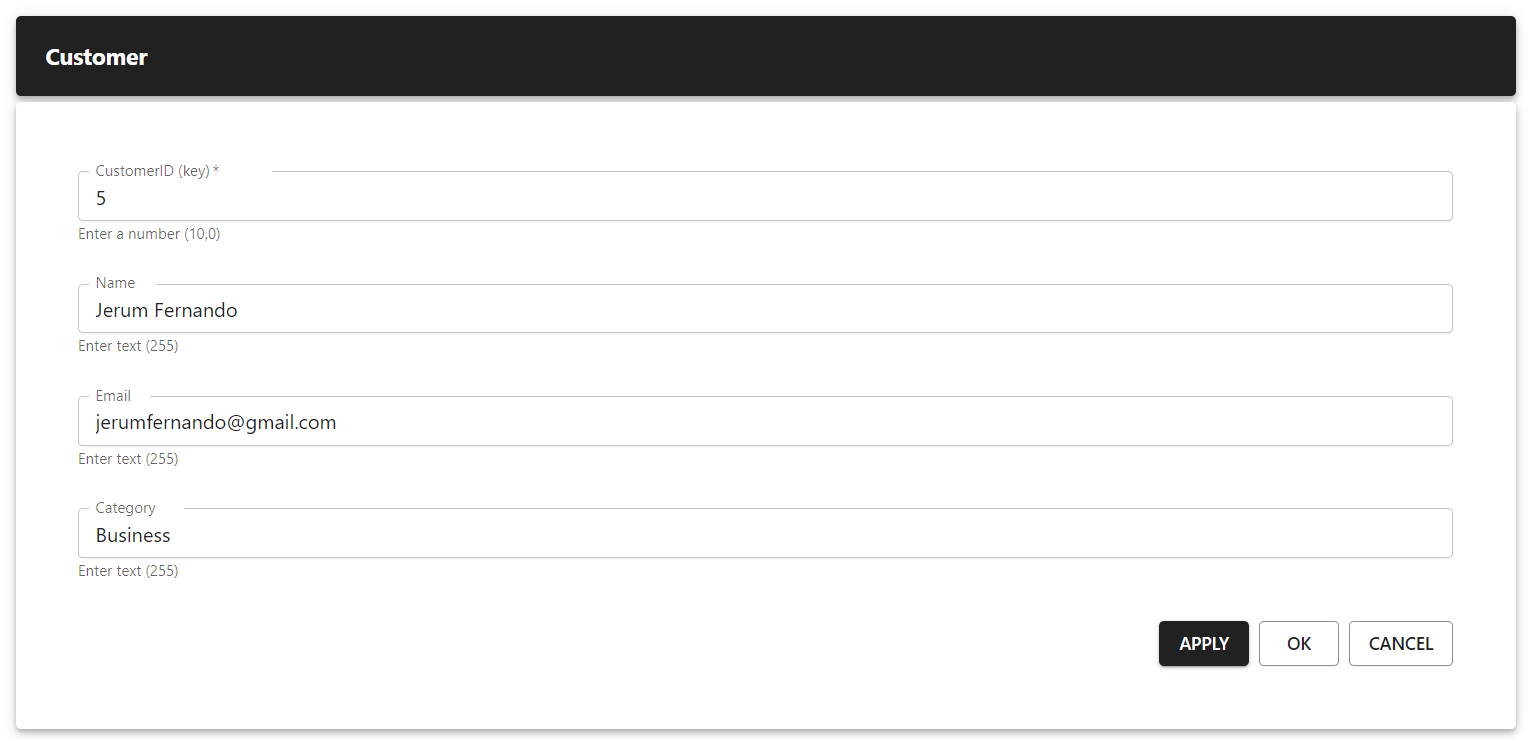


Figure 32:Add a customer



Figure 33:Add a number to the same customer

### Check For Database

As here customer is updated

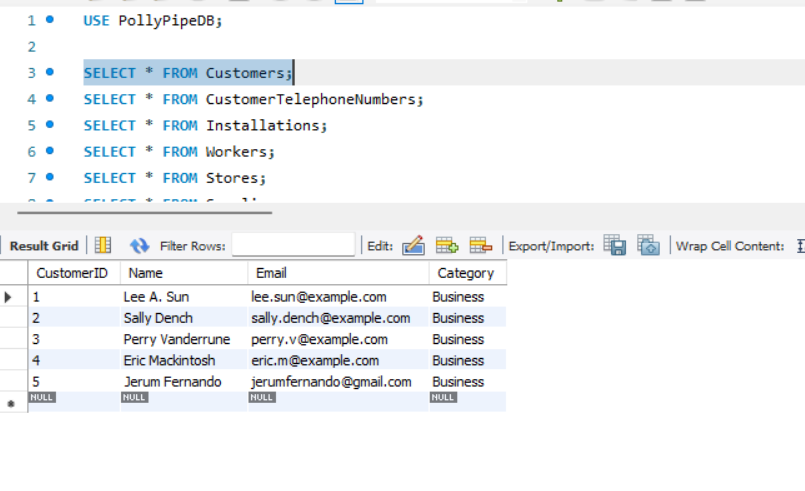


Figure 34:Updated Customer

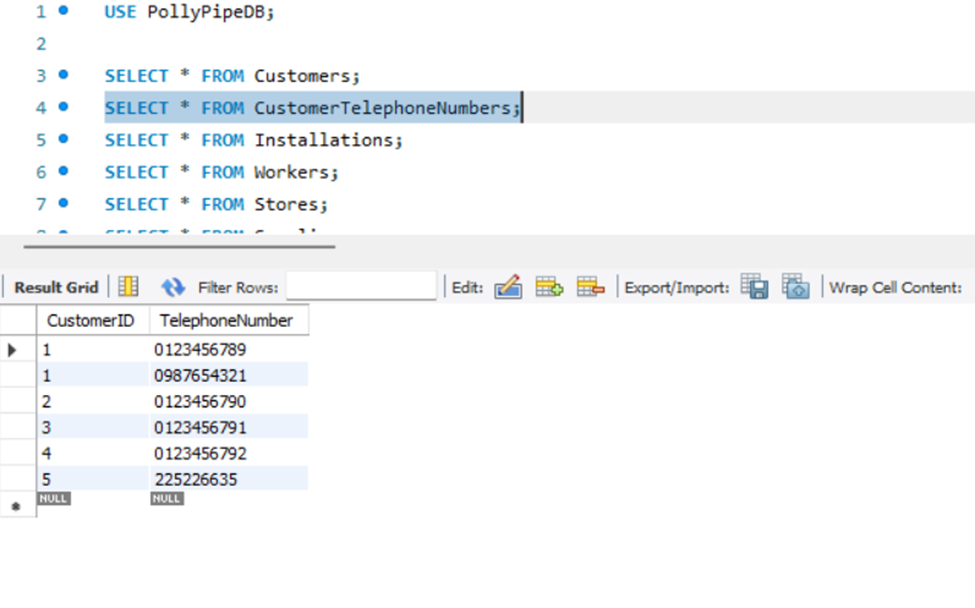
As here Number is also Updated

Figure 35:Updated Number

## Delete Operation

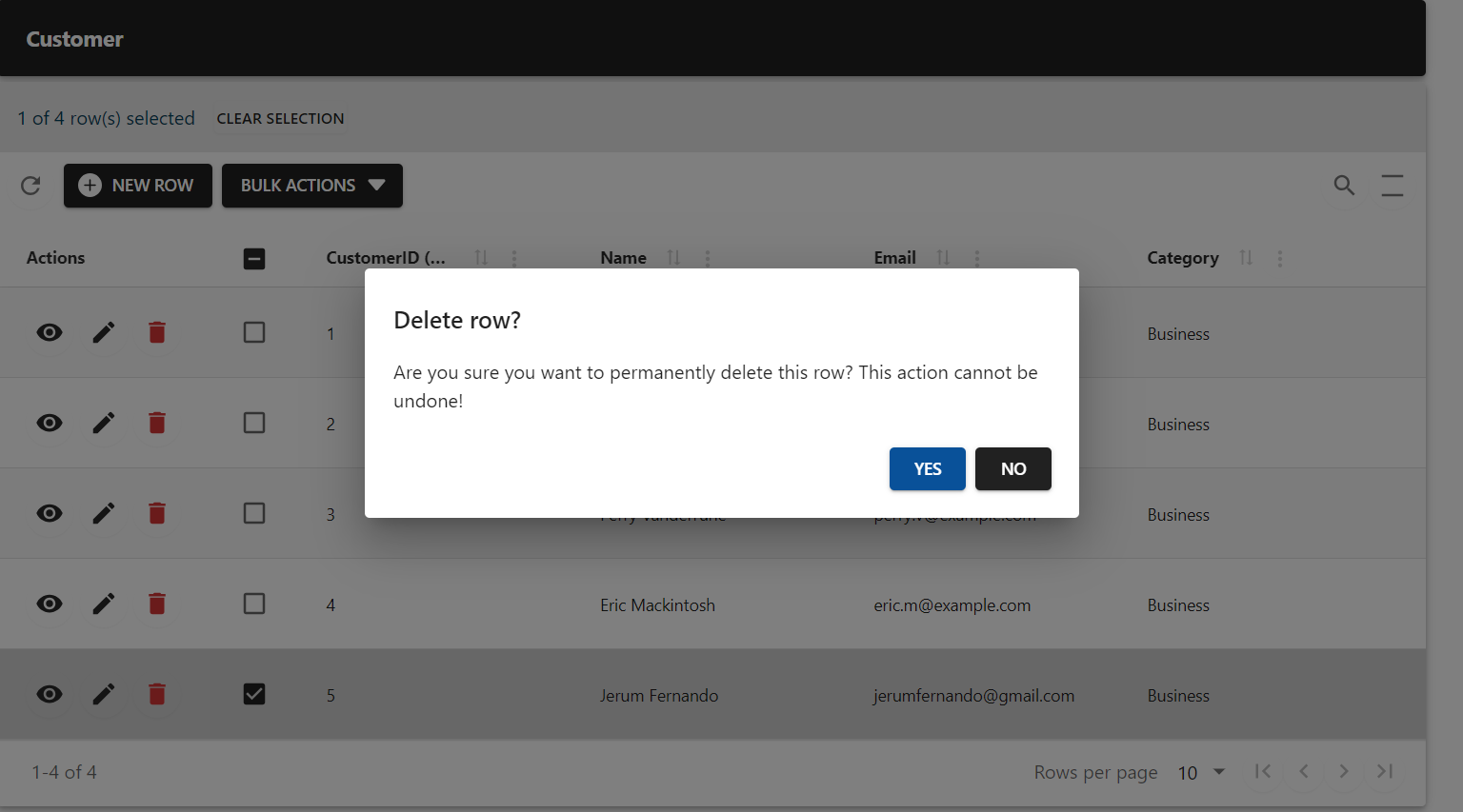
Here we Delete a Customer to see Weather the Database entry is deleted or not.

Figure 36:Delete a Customer

### Check For Database

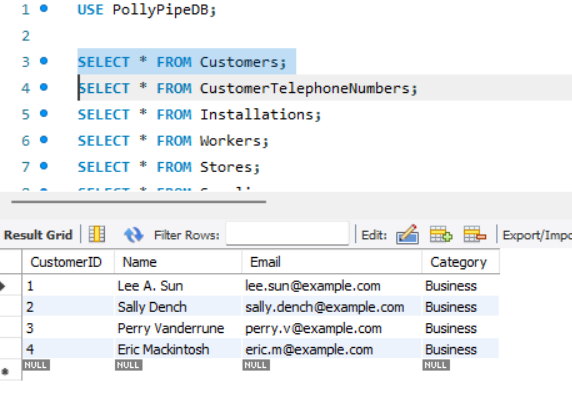
Here database Entry is also deleted .

Figure 37:Deleted Database entry

And Also Corresponding Number Should Also Be Deleted.

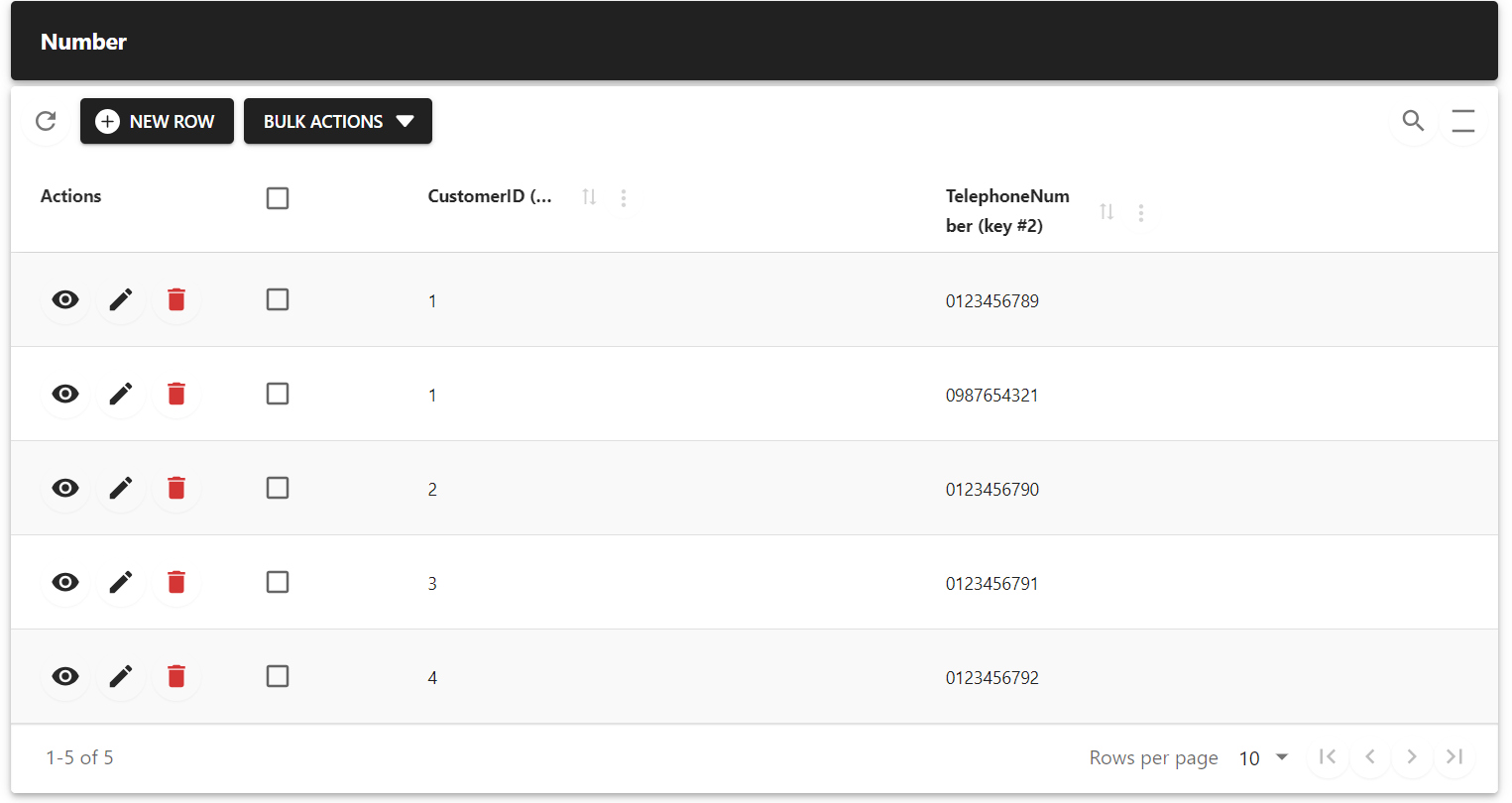
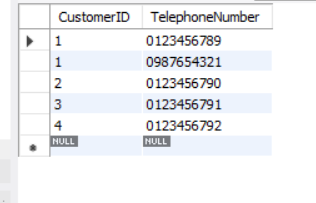


Figure 38:Cascadely Deleted Number In the Database

Figure 39:Deleted Entry Shoun In GUI

## Edit Operation (Update)

### Before Update

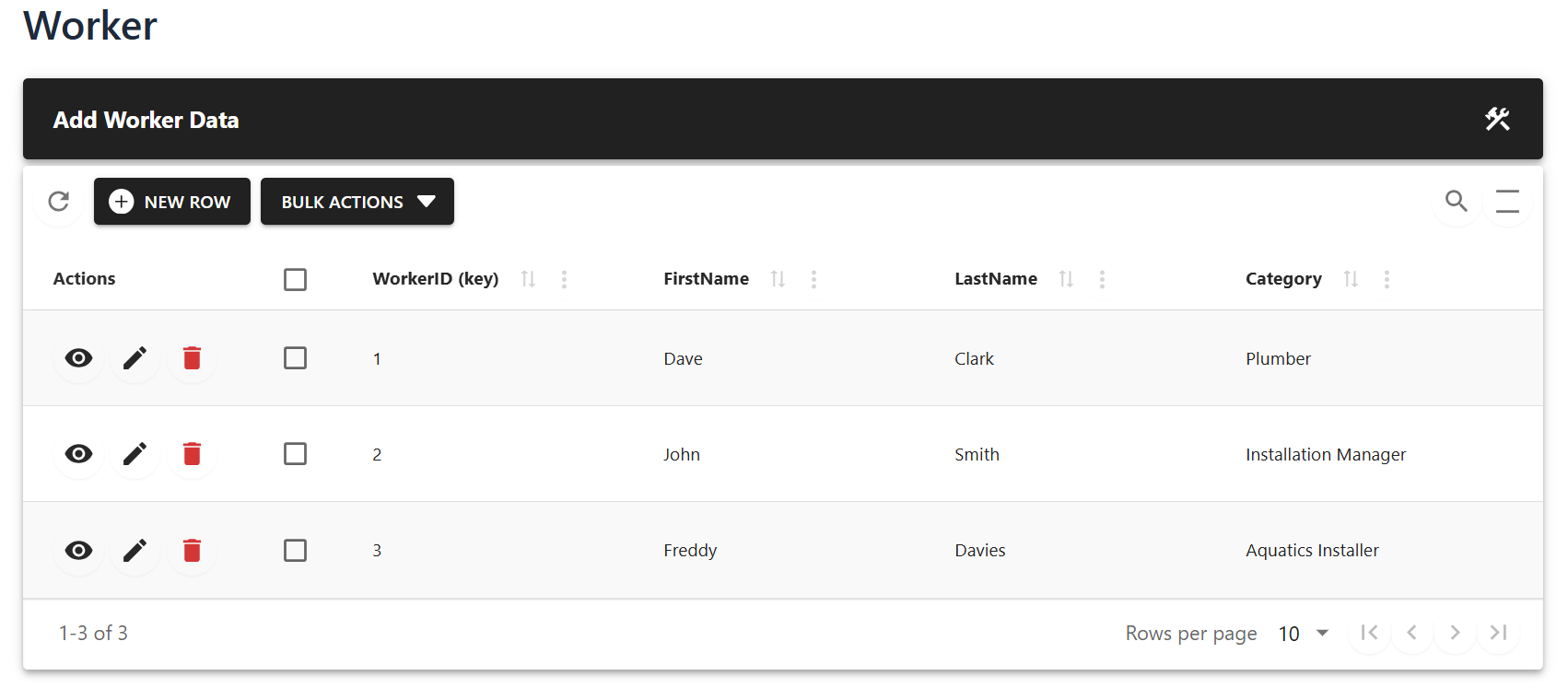


Figure 40:Worker Table before Update

### Update Operation

Last name is updated for KUger.

Figure 41:Updating Last Name

### Implementation

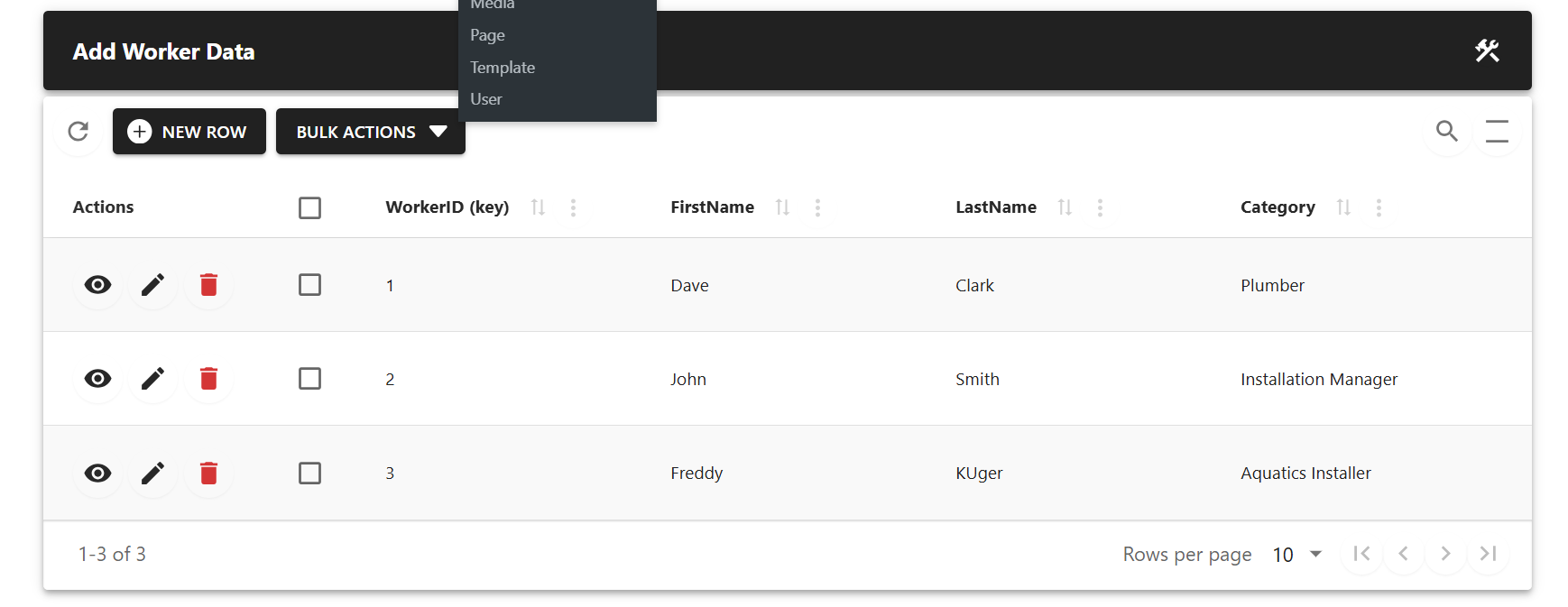
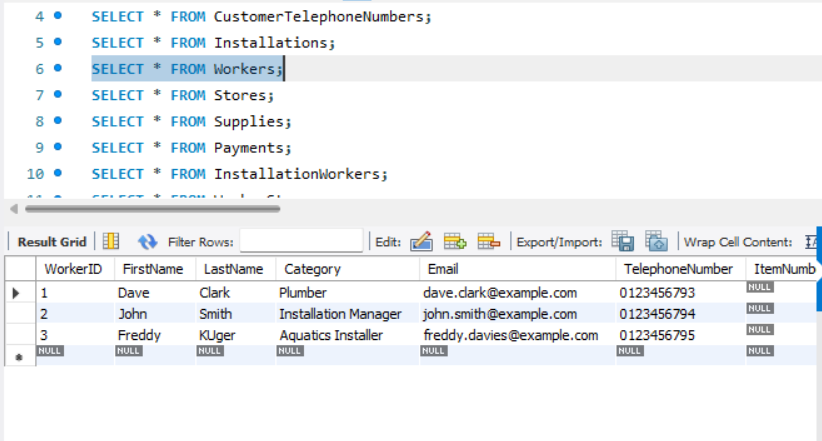


Figure 42:Updated Database

Figure 43:Updated GUI Fields

# Feedback

## User Feedback:

### User Interface and Usability:

* Feedback: ”Firstly, the format of the interface is quite simple and uncluttered; however, it can be confusing how to get from one section to another. While there are tabs, it would be better if the graphic icons were more easily distinguishable to know which section of the application one is opening. ”
* Recommendation: Improve the interactivity of the tool with cleaner navigation paths and popup tips to help the users easily.

### Data Entry and Retrieval:

* Feedback: Exporting is simple, but searching for specific records through the database seems somewhat slow at times, mainly when many users are active at the same time.
* Recommendation: Improve the work of search options and use such techniques as caching to enhance the speed of work.

### Feature Requests:

* Feedback: Currently; there is a search bar in the site that is normally effective in helping users to look for what they need We would however; like to have even more specific filters in the search engine so that users can easily search according to certain parameters.
* Recommendation: Incorporate enhanced elements into the search interface in order to satisfy the various requirements of the users and increase efficiency.

### Security Concerns:

* Feedback: Considering the fact that most of the information accumulated from customers is highly sensitive, measures have to be taken to enhance security of information How can we increase our access controls and encrypt sensitive information?
* Recommendation: Some of the protective measures that can be implemented are tightening up with security and proper identification systems and encryptions to secure the more confidential information.

### Training and Support:

* Feedback: An advantage of the system is that it cannot be easily hacked, however, an employee that is new to their job complains of on boarding without sufficient documentation and training materials.
* Recommendation: Include procedures and educate the users on how to better manage and utilize the available database system.

## Developer Feedback:

### Performance and Scalability:

* Feedback: We noticed some periods of instability in the database response within the course of the stress testing, increasing during the peak loads Though there is no significant growth now, options for scaling and optimization of the queries are vital for further development.
* Recommendation: Add query optimizations, contemplate about the database scaling and integrate caching for the performance.

### Integration Challenges:

* Feedback: There were thus issues in integrating the database with external systems as this was compounded by issues to do with compatibility and also the format of data.
* Recommendation: Introduce improved data massaging tools to seamless integration and make sure the system can interoperate with imported systems.

### Technical Documentation:

* Feedback: ‘Although they provide enough information in the initial documentation, they do not give enough solutions or references with examples on how to handle complicated problems. ’
* Recommendation: Revise the documentation with more detailed instructions on how to troubleshoot a given issue, examples of success and/or failure of the system, and more recommendations which can be of help to developers.

### Feedback Mechanism:

* Feedback: Basically, there is a call for a clearer feedback loop by which developers can continue to get feedback from users on what must be enhanced next.
* Recommendation: Organize a feedback system or user community for refining the concept over time and for prioritizing enhancements with the users’ perspective in mind during the iteration cycles.

### Security Architecture:

* Feedback: Enhancing other aspects of security like RBAC and compound security audits would work to increase the surety in data safety. ”
* Recommendation: Improve the security infrastructure by implementing Role Based Access Control, using regular security assessments, and having measures against possible threats.

In light of the feedback points above, Poly Pipe Company should improve its database system in several ways: This part of the system should be able to meet the users’ expectations of performance, security, and efficiency for growth and development, while encouraging the reception of feedback to address this issue continually. This way, the development process guarantees that the required improvements are incorporated hence enhancing the usability, security and reliability of the company’s database system.

# User Guide

## User Guide Database

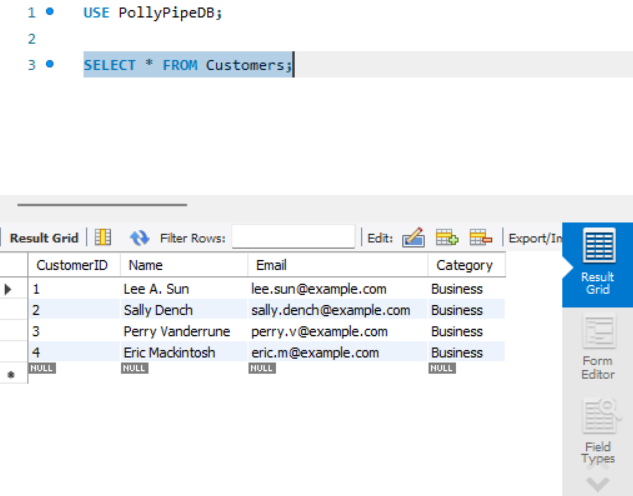
The PollyPipeDB system is designed to manage customer data, installations, workers, stores, supplies, and payments for Polly Pipe, a water sports provider and installer. This documentation provides a detailed technical overview, including database design, implementation, security measures, and future enhancements

Figure 44:Database

### Logical Design

**Customers**: Stores customer details.

* CustomerID (Primary Key)
* Name
* Email
* Category

**CustomerTelephoneNumbers**: Stores customer phone numbers.

* CustomerID (Foreign Key)
* TelephoneNumber (Primary Key)

**Installations**: Stores installation details.

* InstallationID (Primary Key)
* NameOfPlace
* PlaceAddress
* Type
* CustomerID (Foreign Key)

**Workers**: Stores worker details.

* WorkerID (Primary Key)
* FirstName
* LastName
* Category
* Email
* TelephoneNumber
* ItemNumber (Foreign Key)
* WorkNumber (Foreign Key)

**Stores**: Stores information about stores.

* ItemID (Primary Key)
* IssuedDate
* ResponsiblePerson
* ItemPrice

**Supplies**: Stores supply details.

* ItemNumber (Primary Key)
* PricePerUnit
* Quantity
* TotalPrice (Derived)
* Purpose
* StoreID (Foreign Key)

**Payments**: Stores payment details.

* BillNumber (Primary Key)
* BuyerName
* Category
* DatePay
* CustomerID (Foreign Key)

**InstallationWorkers**: Stores the relationship between installations and workers.

* + InstallationID (Foreign Key)
  + WorkerID (Foreign Key)

**WorkerStores**: Stores the relationship between workers and stores.

* + WorkerID (Foreign Key)
  + ItemID (Foreign Key)

### Security Mechanisms

* User roles and permissions.
* Strong password policies.
* Encrypted connections using SSL/TLS.
* Regular audits and monitoring.

### **Operations**

* Open MySQL Workbench or PHPMyAdmin.
* Connect to the PollyPipeDB database using your credentials.

#### **Using PHPMyAdmin**

* Navigate to the table (e.g., Customers).
* Click the “Insert” tab and fill in the form fields.
* Click “Go” to add the data.

#### **Using MySQL Workbench**

* Open a new SQL script window.
* Execute an INSERT statement to add data. For example:

INSERT INTO Customers (CustomerID, Name, Email, Category) VALUES  
(1, 'Lee A. Sun', 'lee.sun@example.com', 'Business');

#### **Using MySQL Workbench**

* Open a new SQL script window.
* Execute an UPDATE statement to modify data. For example:

UPDATE Customers SET Email = 'new.email@example.com' WHERE CustomerID = 1;

#### **Using PHPMyAdmin**

* + Navigate to the table (e.g., Customers).
  + Click the “Browse” tab and then the “Delete” icon next to the record.
  + Confirm the deletion.

#### **Using PHPMyAdmin**

* Navigate to the table (e.g., Customers).
* Click the “Browse” tab and then the “Edit” icon next to the record.
* Modify the data and click “Go”.

#### **using MySQL Workbench**

* Open a new SQL script window.
* Execute a DELETE statement to remove data. For example:

DELETE FROM Customers WHERE CustomerID = 4;

## User Guide For GUI

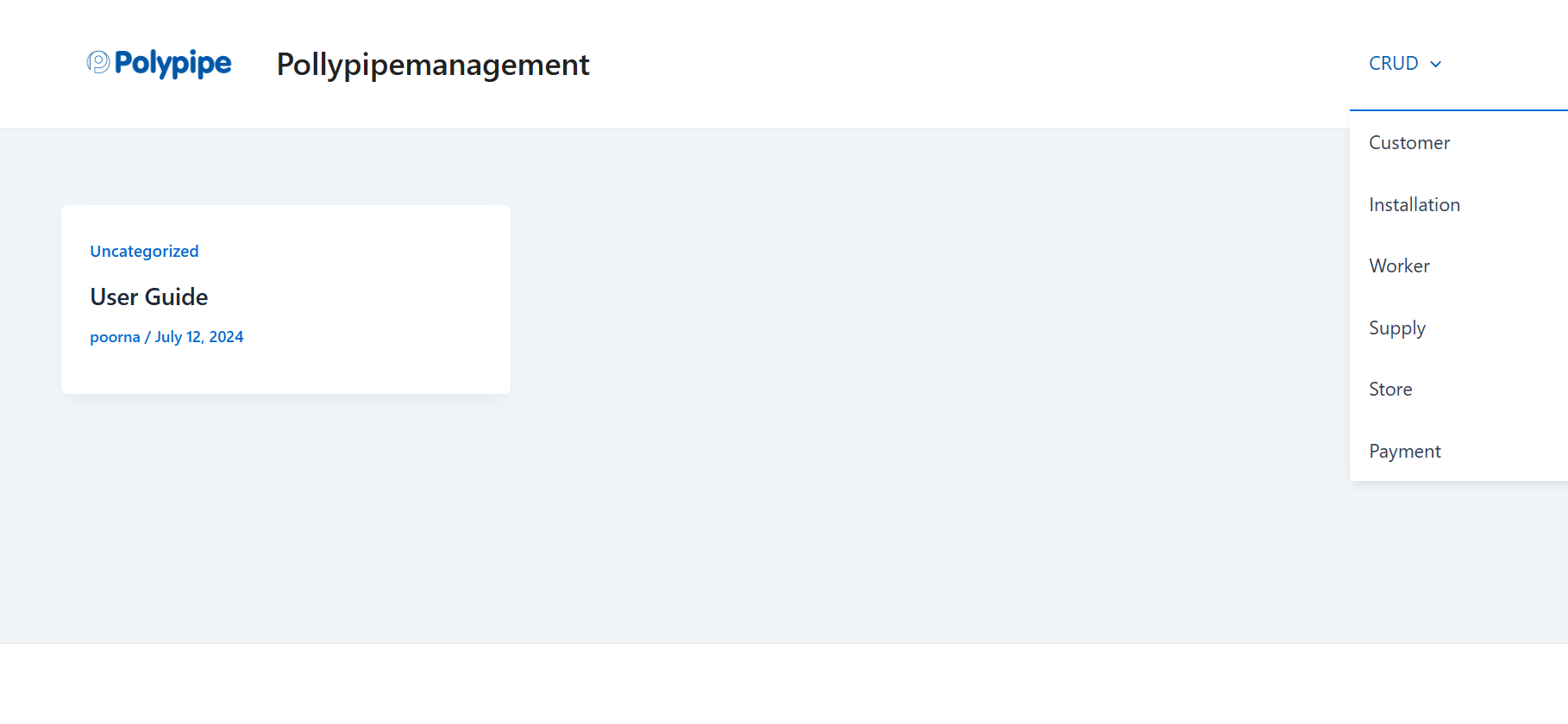


Figure 45:Web GUI

### Functions

* Sign in sign up
* Crud operation on any Table
* Cascade delete operation
* Cascade Update Operation
* Direct Connection to the Database

### Important

* Web application Using PHP
* WordPress application
* when Signed in no sign in button shows
* Step By step DataBase Guide is given in the Forms

### Customer form and Create operation Windows

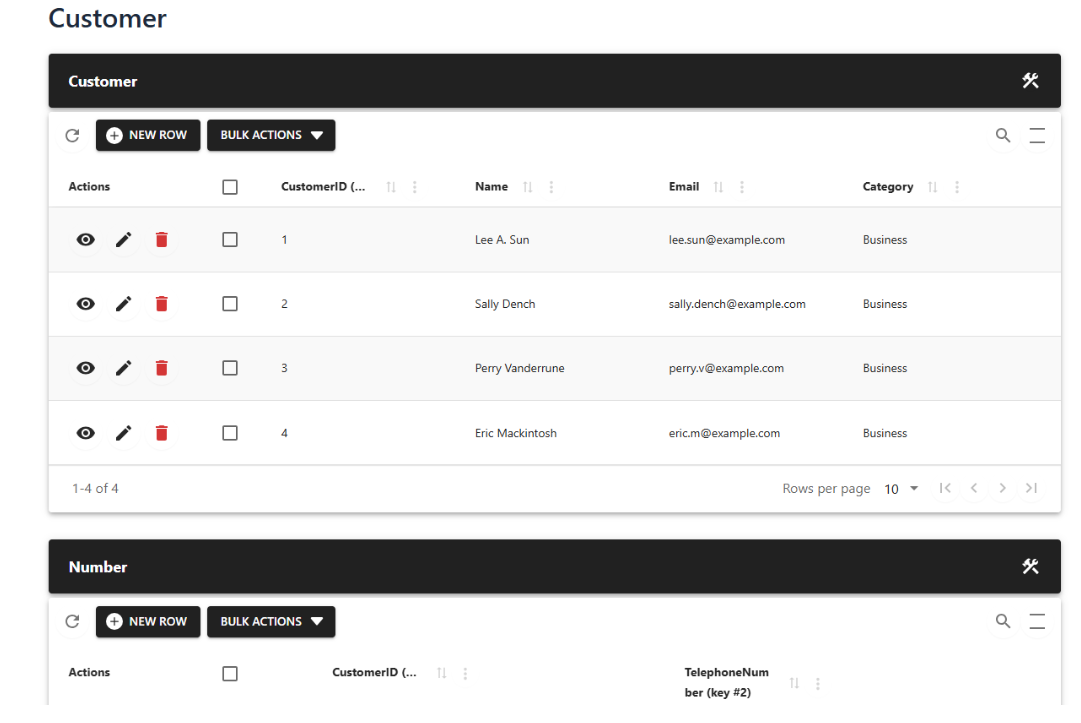
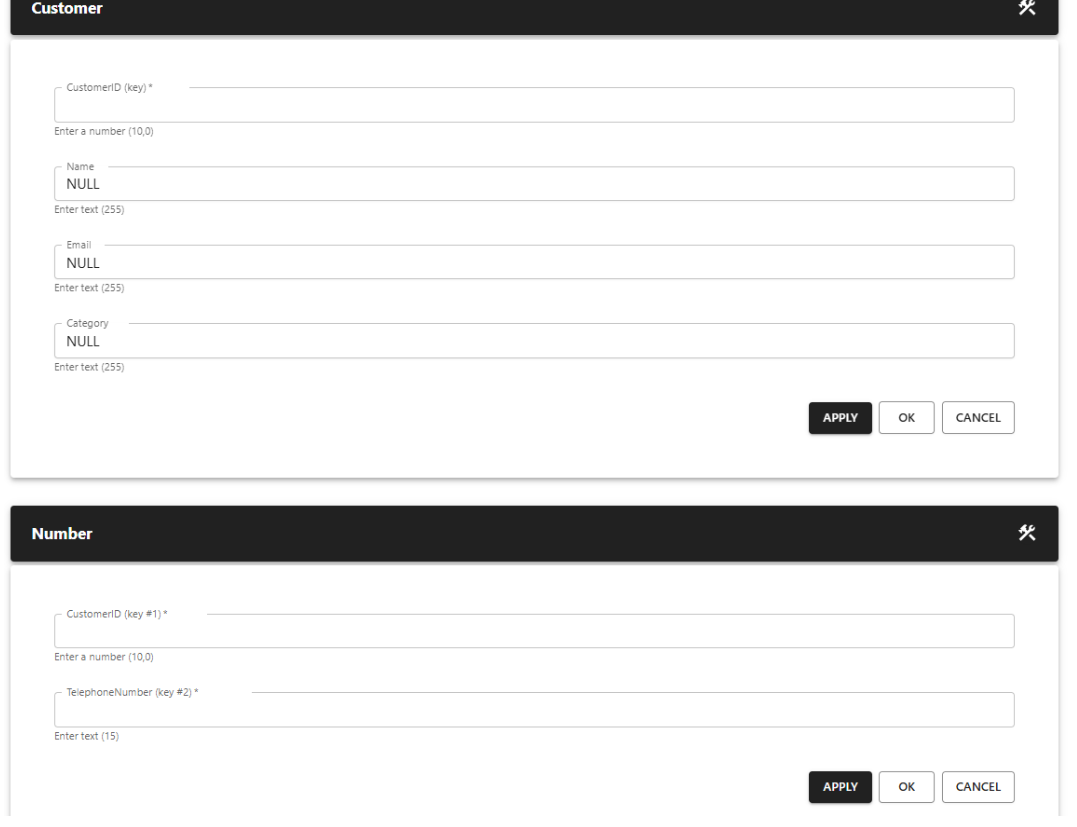


Figure 46:Insert Forms

Figure 47:Forms In the GUI

### Guide

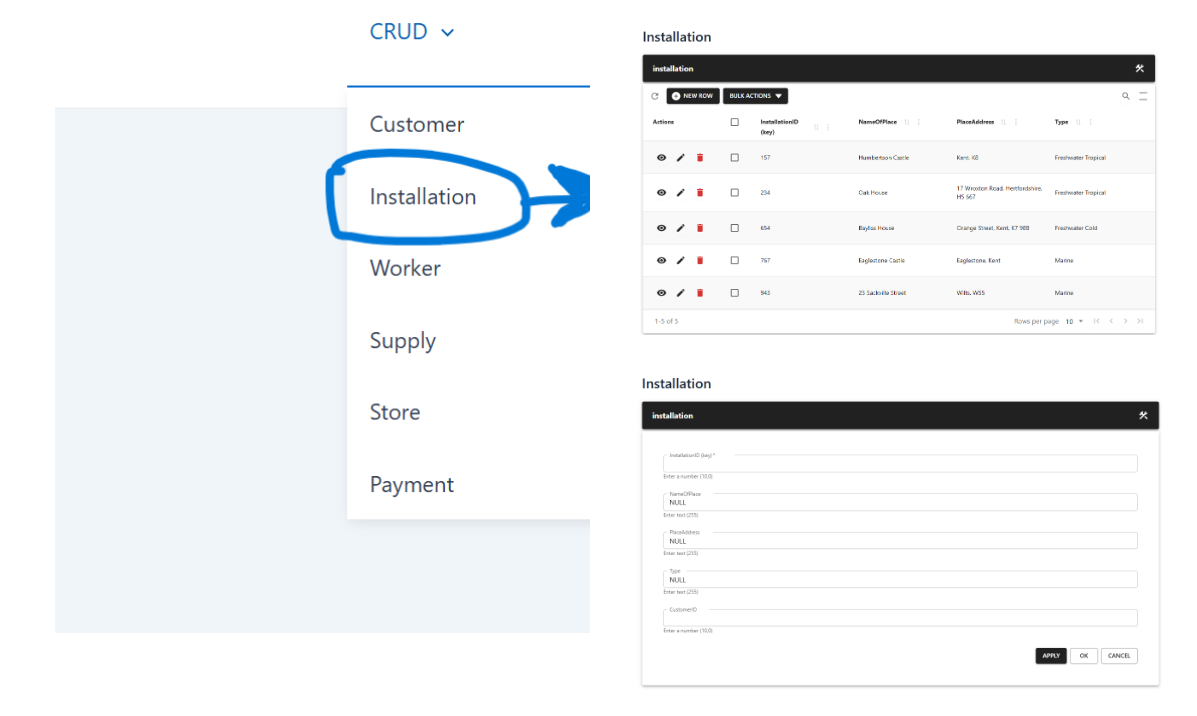
On the Gui there is a Catagory called CRUD on the right upper corner When howering your arrow head over that you can see operation List From that you can Choos whatever you want to Change .

Figure 48:Guide For CRUD

### CRUD Operation

Here is the explanation how to Do crud operations from the the Form table Display.

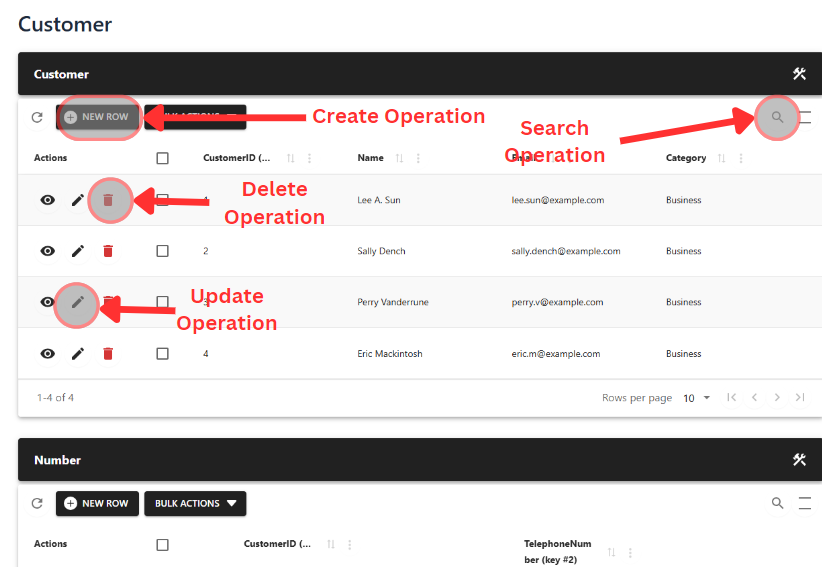


Figure 49:Crud Operation

### Create a Customer

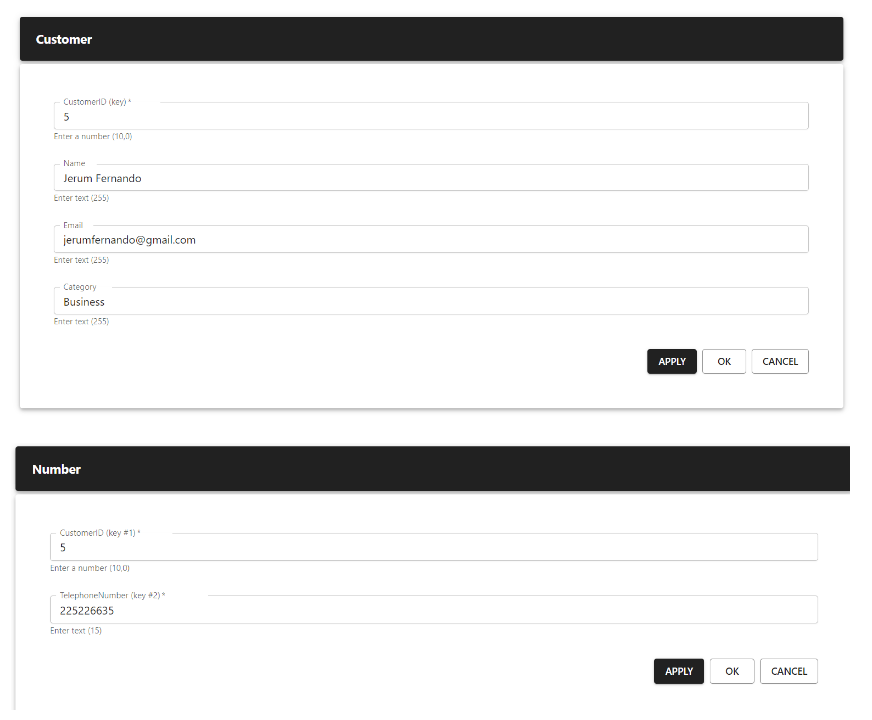


Figure 50:Creating a Customer Using GUI

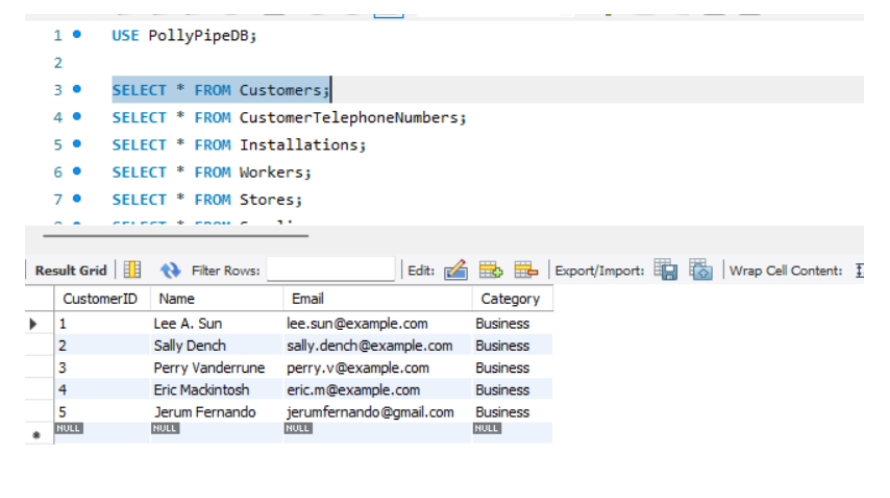
when creatring a customer there should be entry in the database also here number is also should define

Figure 51:Updated Database Entry

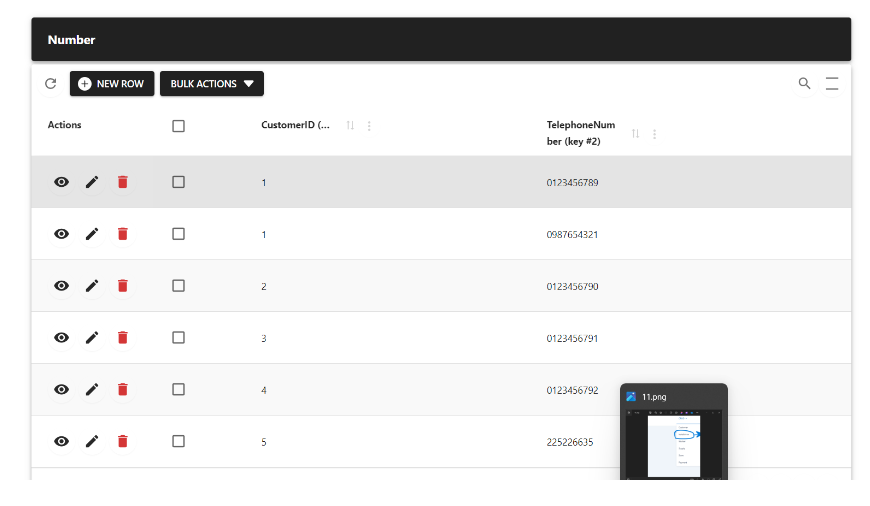
It is also Updated in the Number table and it is shown in the GUI also.

Figure 52:GUI Update

### GUI posts

#### Database GUIDE

In the Web application also you can see the step by step guide on tow to Operate.



Figure 53:Inbuild User Guide For Database

#### GUI GUIDE

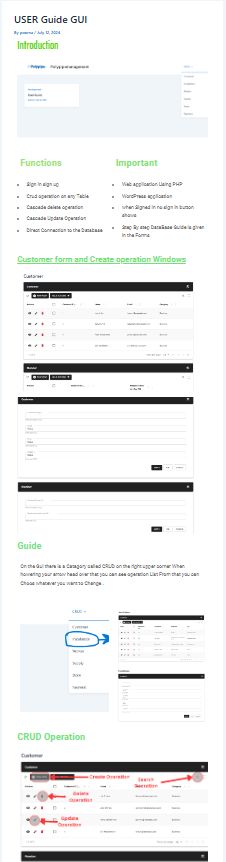
In the Web application also you can see the step by step guide on how to Operate GUI.

Figure 54:Inbuild USer GUIDE GUI

# Conclusions

The creation of the database system for Poly Pipe Company as well as its integration in the organization represent a major achievement in contributing to their effectiveness and methods of organization, record keeping. This all-encompassing process has achieved two objectives for them: it has helped them convert their conventional record-storage habits into a more efficient, computerized database method; and it has made certain that those methods address the needs of their business.

Thus, much attention was devoted to the collection and satisfaction of user and system needs throughout the project. When working on the first phases of the development, the end-users and stakeholders are integrated, thus, the proposed solution reflects the real needs of the company. There is no doubt that the integration of users’ responses in the development stage played a crucial role in enabling the development of an Interface that employs simplicity and easy navigability in order to create a seamless transition to employees at Poly Pipe Company.

The critical functions include create, read, update, and delete operations, extended search capabilities, as well as automatic notifications, which all are designed to optimize the company’s operations. However, the way of organizing the database: entities, their association, relations, and the usage of primary and foreign keys safeguard data from getting damaged or inconsistent and this is very important given that the database requires accurate data to make informed decisions.

To ensure that the database and the whole system was functioning efficiently, optimally, and securely, unit, integration, and system testing was performed. These tests were important in the diagnosis of any problem that may be there hence correct them to enable the system work properly under different situations. Some of the findings included recommendations on how to improve on the system from the user interface to query performance as proposed by users and developers who were non-technical.

The following measures were instituted so to give protection to customer information pertinent to security: Data validation, restriction in access and control, and encryption so as to answer for data protection regulations and to extend trust from the end-users touching the system’s reliability aspect.

To prevent data redundancy, normalization methods were used and these methods helped not only in efficient data storage but also in efficient extraction of the same data. The use of these techniques assisted in eradicating the growth of unwanted database anomalies and made it possible to organize the database in a presentable format as well as ensuring that the database remained accurate and up to date.

Therefore, the new database system for Poly Pipe Company means a valuable enhancement over the previous manual records. It offers the company a solution, which is affordable, secure and would suit the organization’s needs in case of expanded in the future. The engagement of the users in the development cycle, testing as well as integration of proper database development techniques and security measures have given the company a system that is efficient in support of the business operations of the Company. In order to sustain the process of enhancement and stay relevant to the future changes of the company’s needs and demands of the IT field, prompt feedback and frequent revisions will be necessary in the future.

# REFERENCES

* Adepoju, S. (2021). Database Testing Tools and Techniques. Retrieved from <https://www.example.com>
* Informatica. (2021). Data Validation: Techniques and Best Practices. Retrieved from <https://www.example.com>
* Pankaj, P. (2020). Validation and Verification in Software Testing. Retrieved from <https://www.example.com>
* ReQtest. (2017). Database Testing: All You Need to Know. Retrieved from <https://www.example.com>
* Telend. (2021). Data Redundancy: Definition, Examples, and Solutions. Retrieved from <https://www.example.com>
* **Note:** Please replace "<https://www.example.com>" with the actual URLs of the sources used