Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS				
Programme title	**BTEC HND in Computing**			
Assessor		**Internal Verifier**		
Unit(s)	**Unit 04: Database Design & Development**			
Assignment title	**Database Solution for Polly Pipe**			
Student's name	Ryan Wickramaratne			
List which assessment criteria the Assessor has awarded.	**Pass**	**Merit**	**Distinction**	

INTERNAL VERIFIER CHECKLIST		
Do the assessment criteria awarded match those shown in the assignment brief?	Y/N	
Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work?	Y/N	
Has the work been assessed accurately?	Y/N	
Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions?	 Y/N Y/N Y/N Y/N	
Does the assessment decision need amending?	Y/N	

Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)		**Date**	

Confirm action completed			
Remedial action taken Give details:			
Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)		**Date**	

Higher Nationals - Summative Assignment Feedback Form

Student Name/ID			
Unit Title	Unit 04: Database Design & Development		
Assignment Number	1	**Assessor**	
Submission Date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	

Assessor Feedback:

LO1 Use an appropriate design tool to design a relational database system for a substantial problem

Pass, Merit & Distinction Descripts **P1** ☐ **M1** ☐ D1 ☐

LO2 Develop a fully functional relational database system, based on an existing system design

Pass, Merit & Distinction Descripts **P2** ☐ **P3** ☐ **M2** ☐ **M3** ☐ **D2** ☐

LO3 Test the system against user and system requirements.

Pass, Merit & Distinction Descripts **P4** ☐ **M4** ☐ **D2** ☐

LO4 Produce technical and user documentation.

Pass, Merit & Distinction Descripts **P5** ☐ **M5** ☐ **D3** ☐

Grade:	**Assessor Signature:**	**Date:**
Resubmission Feedback:		
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
Signature & Date:		

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.

Assignment Feedback

Formative Feedback: Assessor to Student			
Action Plan			
Summative feedback			
Feedback: Student to Assessor			
Assessor signature		**Date**	
Student signature		**Date**	

Pearson Higher Nationals in

Computing

Unit 04: Database Design & Development

Assignment 01

General Guidelines

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Time New Roman**.
2. **Use 1.5 line spacing**. Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each pag**e. This is useful if individual sheets become detached for any reason.
5. Use word processing application spell check and grammar check function to help editing your assignment.

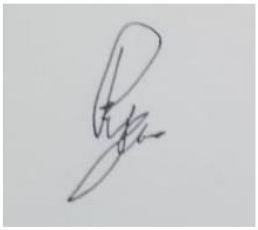
Important Points:

1. It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.
2. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.
6. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.
10. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct form. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of Edexcel UK.
3. I know what the consequences will be if I plagiarise or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspect of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Pearson, UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the assignment.



ryandilthusha@gmail.com

28/02/2022

Student's Signature:
(*Provide E-mail ID*)

Date:
(*Provide Submission Date*)

Higher National Diploma in Computing

Assignment Brief

Student Name /ID Number	Ryan Wickramaratne (COL 00081762)
Unit Number and Title	**Unit 4: Database Design & Development**
Academic Year	2021/22
Unit Tutor	Mr. Praveen Croos
Assignment Title	**Data base system for Polly Pipe**
Issue Date	
Submission Date	
IV Name & Date	

Submission format	
Part 1: The submission should be in the form of an individual written report written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using Harvard referencing system. Please also provide in-text citation and bibliography using Harvard referencing system. The recommended word limit is 3,000–3,500 words, although you will not be penalised for exceeding the total word limit. Part 2: The submission should be in the form of a fully functional relational database system demonstrated to the Tutor; and an individual written report (please see details in Part 1 above). Part 3: The submission should be in the form of a witness statement of the testing completed by the Tutor; technical documentation; and a written report (please see details in Part 1 above).	
	Unit Learning Outcomes:
	LO1 Use an appropriate design tool to design a relational database system for a substantial problem. LO2 Develop a fully functional relational database system, based on an existing system design. LO3 Test the system against user and system requirements. LO4 Produce technical and user documentation.
	Assignment Brief and Guidance:

Assignment brief

Polly Pipe is a water sports provider and installer based in Braintree, England. They need you to design and implement a database that meets the data requirements. These necessities are defined in this scenario and below are samples of the paper records that the Polly Pipe preserves.

Polly Pipe is focused in placing aquariums at business customers. Customers can request several installations, but each installation is tailor-made for a specific customer. Facilities are classified by type. One or more employees are assigned to each facility. Because these facilities are often very large, they can include carpenters and masons as well as water installers. The facilities use equipment such as aquariums, air pumps and thermostats. There can be multiple computers in a facility.

Below are examples of paper records that Polly Pipe currently maintains.

Staff Management Record

Staff Number	**Name**	**Type**
SHA1	Dave Clark	Plumber
SHA8	John Smith	Installation Manager
SHA2	Freddy Davies	Aquatics installer
SHA11	McCloud	Aquatics installer
SHA23	Satpal Singh	Plumber
SHA66	Winstn Kodogo	Aquatics installer
SHA55	Alison Smith	Brick Layer

Equipment Type Table

Type	Equipment
Tanks	20 gallon tank, 50 gallon tank, 100 gallon tank, 200 gallon tank
Thermostats	Standard, Super
Air Pumps	Standard, Super
Filters	Air driven, Undergravel

Instillation Management Form

Installation ID	Installation Type	Installation Name and Address	Customer	Equipment	Types of Staff Required	Period of Staff assignment
234	Freshwater Tropical	Oak House, 17 Wroxton Road, Hertfordshire, H5 667	Lee A. sun	2 air pumps 200 gallons fish tank 1 x standard thermostat	1 x Carpenter 1 x Aquatics installer 1 x Electrician	From 1st September 2012
654	Freshwater Cold	Bayliss House, Orange Street, Kent, K7 988	Sally Dench	2 air pumps 200 gallons fish tank Large Gravel Bag 2 x standard thermostats	5 x Carpenters 1 x Installation Manager 1 x Aquatics installer 1 x Plumber 3 x Labourers	1st June 2005 – 1st June 2011
767	Marine	Eaglestone Castle, Eaglestone, Kent	Perry Vanderrune	2 x 200 gallons fish tanks 500 Wood panels	10 x Carpenters 2 x Installation Manager 1 x Aquatics installer 1 x Plumber 3 x Labourers	From 30th June 2012
943	Marine	23 Sackville Street, Wilts. W55	Eric Mackintosh	2 air pumps 200 gallons fish tank 1 x standard thermostat	No staff required	
157	Freshwater Tropical	Humbertson Castle, Kent, K8	Perry Vanderrune	2 air pumps 400 gallons fish tank 3 x standard thermostat	1 x Aquatics installer	1st September 2005 – 1st September 2012

Activity 1

1.1. Identify the user and system requirements to design a database for the above scenario and design a relational database system using conceptual design (ER Model) by including identifiers (primary Key) of entities and cardinalities, participations of relationships. Convert the ER Model into logical database design using relational database model including primary keys foreign keys and referential Integrities. It should contain at least five interrelated tables. Check whether the provided logical design is normalised. If not, normalize the database by removing the anomalies.

(Note:-It is allowed to have your own assumptions and related attributes within the scope of the case study given)

1.2. Design set of simple interfaces to input and output for the above scenario using Wireframe or any interface-designing tool. Evaluate the effectiveness of the given design (ERD and Logical design) in terms of the identified user and system requirements .

Activity 2

Activity 2.1

a. Develop a relational database system according to the ER diagram you have created (Use SQL DDL statements). Provide evidence of the use of a suitable IDE to create a simple interface to insert, update and delete data in the database. Implement proper security mechanisms in the developed database.

   Evaluate the database solution developed and its effectiveness with relevant to the user and system requirements identified, system security mechanisms (EX: -User groups, access permissions) and the maintenance of the database.

Activity 2.2

a. Explain the usage of DML with below mentioned queries by giving at least one single example per each case from the developed database. Assess the usage of the below SQL statements with the examples from the developed database to prove that the data extracted through them are meaningful and relevant to the given scenario.

   Select/ Where / Update / Between / In / Group by / Order by / Having

Activity 3

Activity 3.1

Provide a suitable test plan to test the system against user and system requirements. provide relevant test cases for the database you have implemented. Assess how the selected test data can be used to improve the effectiveness of testing.

Note:- Learner needs to give expected results in a tabular format and screenshots of the actual results with the conclusion

Activity 3.2

Get independent feedback on your database solution from the non-technical users and some developers (use surveys, questioners, interviews or any other feedback collecting method) and make recommendations and suggestions for improvements in a separate conclusion/recommendations section.

Activity 4

Produce a technical documentation and a user guide for the developed database system. Suitable diagrams diagrams (Use case diagram, class diagram, flow charts, DFD level 0 and 1) should be included in the technical documentation to show data movement in the system. Assess the developed database by suggesting future enhancements to ensure the effectiveness of the system.

Grading Criteria	Achieved	Feedback
LO1 Use an appropriate design tool to design a relational database system for a substantial problem		
P1 Design a relational database system using appropriate design tools and techniques, containing at least four interrelated tables, with clear statements of user and system requirements.		
M1 Produce a comprehensive design for a fully functional system that includes interface and output designs, data validations and data normalization.		
D1 Evaluate the effectiveness of the design in relation to user and system requirements.		

LO2 Develop a fully functional relational database system, based on an existing system design		
P2 Develop the database system with evidence of user interface, output, and data validations, and querying across multiple tables.		
P3 Implement a query language into the relational database system		
M2 Implement a fully functional database system that includes system security and database maintenance.		
M3 Assess whether meaningful data has been extracted using query tools to produce appropriate management information.		

LO3 Test the systems against user and system requirements		
P4 Test the system against user and system requirements.		
M4 Assess the effectiveness of the testing, including an explanation of the choice of test data used.		
LO2 & LO3 **D2** Evaluate the effectiveness of the database solution in relation to user and system requirements, and suggest improvements.		
LO4 Produce technical and user documentation		
P5 Produce technical and user documentation.		
M5 Produce technical and user documentation for a fully functional system, including diagrams showing movement of data through the system, and flowcharts describing how the system works.		
D3 Evaluate the database in terms of improvements needed to ensure the continued effectiveness of the system.		

Acknowledgement

I would like to express my special thanks of gratitude to my programing lecturer Mr. Praveen for providing invaluable guidance and giving immense amount of knowledge to work on this assignment perfectly. I specially thanks him because he helped us in doing a lot of research and I came to know about so many new things about the database designing and developing.

Secondly, I would like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Executive Summery

This entire assignment is based on an implementation of a Water Equipment installation system design for selected company (Polly Pipe). The purpose of this assignment is to improve database developing and management skills.

This report includes basic database software handling, creating designs, understanding data flows, understanding different database designing methods, working with IDEs, Basic system creation using Microsoft SQL and Visual Studio C# windows forms application, database normalization by finding and resolve anomalies, using different SQL query types when creating a database for a system, make a documentation for a system with describing future enhancements for the system.

List of figures

List of Tables

TABLE OF CONTENTS

Activity 1

1.1 ER Diagram

1.1.1 What is ER Diagram

Entity Relationship Diagram (ERD) is a diagram that shows the relationship between entity sets contained in a database. In other words, ER diagrams help in the explanation of database logical structure. Entities, attributes, and relationships are the three main components that ER diagrams are built on.

Rectangles are used to represent entities, ovals are used to define attributes, and diamond shapes are used to show relationships in ER Diagrams.

An ER diagram appears to be quite similar to a flowchart at first glance. The ER Diagram, on the other hand, contains many specific symbols, and the meanings of these symbols distinguish this model. The entity framework infrastructure is represented by the ER Diagram.

1.1.2 What is ER Model

Entity Relationship Model (ER Model) is a high-level conceptual data model diagram. The ER model aids in the methodical analysis of data requirements in order to create a well-designed database. The ER Model is a representation of real-world things and their relationships. Before we implement our database, we should create an ER Model in a database management system.

ER Modelling assists us in doing a systematic analysis of data requirements in order to create a well-designed database. As a result, completing ER modelling before implementing our database is considered best practice.

1.1.3 Why use ER Diagrams?

- ERD supports in the define the concepts used in entity relationship modelling.
- These diagrams show how all of our tables should relate, as well as what fields will be on each table.
- This helps in the definition of entities, attributes, and relationships.
- ER diagrams may be converted into relational tables, allowing us to create databases quickly.
- Database designers can utilize ER diagrams as a roadmap for implementing data in specific software applications.
- With the help of an ERD diagram, the database designer gains a better knowledge of the information that will be stored in the database.
- The ERD Diagram helps us to explain the database's logical structure to users.

1.1.4 ER Diagrams Symbols & Notations

The rectangle, oval, and diamond are the three primary symbols used to represent relationships between elements, entities, and attributes in the Entity Relationship Diagram. Some sub-components in the ERD Diagram are based on the main elements. The ER Diagram is a visual representation of data that shows how different ERD Symbols and Notations are used to explain how data is related to one another.

Main components and its symbols in ER Diagrams:

- Rectangles → This symbol represents entity types
- Ellipses → This symbol represent attributes
- Diamonds → This symbol represents relationship types
- Lines → This links attributes to entity types and entity types with other relationship types
- Primary key → Attributes are underlined when representing.
- Double Ellipses → This symbol represents multi-valued attributes

1.1.5 3 basic components of the ER Diagram

There are 3 basic components in ER Diagram:

1. Entity
2. Attribute
3. Relationship

1) Entity

An ERD entity is a simply definable concept within a system. A database entity can be a place (e.g. School), a person/role (e.g. Student), an object (e.g. Invoice), an event (e.g. Transaction), or a concept (e.g. Profile). In ERD, the term "entity" is frequently used instead of "table," however the two terms are interchangeable.

When determining entities, we must consider them to be nouns. An entity in an ER model is represented by a rectangle. Inside the rectangle is written the Entity's name. Entities must have an attribute and a unique key as part of their characteristics. Every entity is built up of a set of 'attributes' that define it.

Ex: - * Peron – Doctor, Teacher, Student

* Place – School, Office, Building

* Object – Pen, Bus, Machine

* Event – Promotion, Sale, Renewal

* Concept – Account, Course, Profile

An entity set is a collection of entities that are similar in nature. It may contain entities with attribute values that are similar. It's an object for which we'd like to model and store data. Properties, often known as attributes, are used to represent entities.

Each attribute has its own set of values. For example, student entity may have properties such as name, age, and class. If we want to store data about students in School database, the entity in the diagram is represented by rectangles. Inside the rectangle is written the Entity's name.

Student Entity

2) Attribute

The term "attributes" is also used to refer to a column. A property or characteristic of the entity that holds it is called an attribute. An attribute has a name that identifies the property and a type that describes the type of attribute, such as varchar for a string and int for an integer. When creating an ERD for physical database development, it's critical to choose data types that are supported.

The attributes are represented by an oval shape. The attribute's name is written inside the oval shape, and a line connects it to its entity. An entity with some attributes is shown in the ER diagram below.

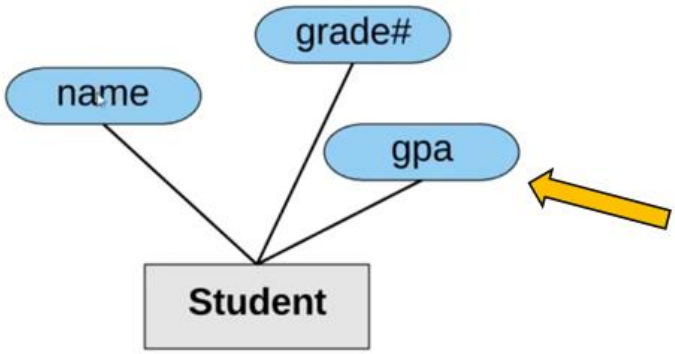


Figure 1. 1 Relevant attributes for student entity

3) Relationship

The term "relationship" refers to the relationship that exists between two entities. A relationship is nothing more than a link between two or more entities. A relationship between two entities shows that the two entities are linked in some way. For example, a student might take in a class. As a result, the entities Student and Course are linked, and a relationship is shown as a connector joining them.

The relationship type is represented as a diamond in the ER diagram, with lines linking the items. For example, when we combine the Student Entity and the Class Entity, Student can take a class. Class can be taken by student.

1.1.6 Other components of the ER Diagram

1) Primary Key

A primary key, often known as PK, is a type of entity attribute that uniquely identifies a record in a database table. To put it another way, there can't be two (or more) records with the identical primary key attribute value.

The primary key is represented by an underline in the attribute name.

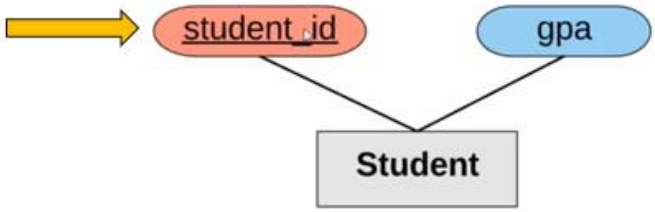


Figure 1. 2 Student entity's primary key

2) Foreign Key

A foreign key, sometimes known as an FK, is a reference to a table's main key. It's used to figure out what entities are related to each other. Foreign keys do not have to be unique.

The ER Diagram below shows an entity with one column, one attribute of which contains a foreign key for referencing another entity.

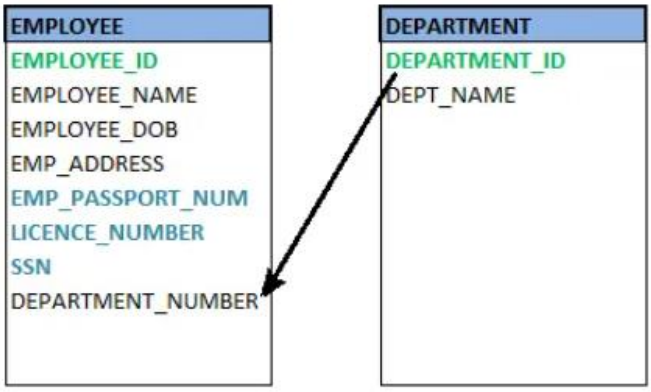


Figure 1. 3 How Foreign key being used with example tables

3) Composite Attribute

A composite attribute is an attribute that is made up of several other attributes. It simply means it's an attribute that can be broken up into sub-attributes.For example the address element of the student entity type includes Street, City, State, and Country. The composite attribute is represented as an oval made up of ovals in the ER diagram.

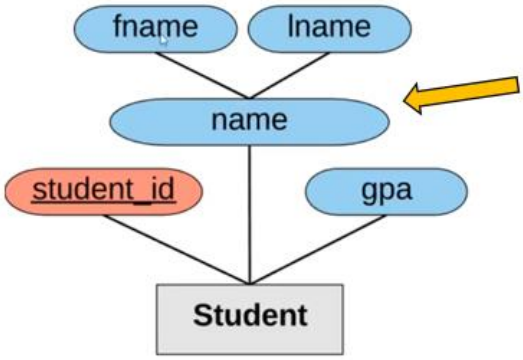


Figure 1. 4 Student Entity's composite attribute

4) Multi- Valued Attribute

Multi- Valued Attribute is an attribute that can have several values. For a given entity, it's an attribute with several values. For instance, Phone No (can be more than one for a given student).

Another example is students may participate in a few clubs (Dexter may participate in Scouts, Hockey, Drama), but they only have one GPA and one name. A multivalued attribute is represented by a double oval in the ER diagram.

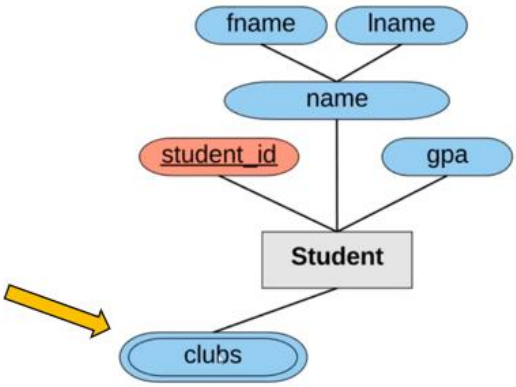


Figure 1. 5 Student Entity's multivalued attribute

5) Derived Attribute

A derived attribute is an attribute that can be derived from other attributes of the entity type. Age, for example (can be derived from DOB). Another example is GPA can be used to determine whether or not someone has Honours (Anyone with a 3.5 or higher receives honours).

Derived attribute is represented by a dashed oval in the ER diagram.

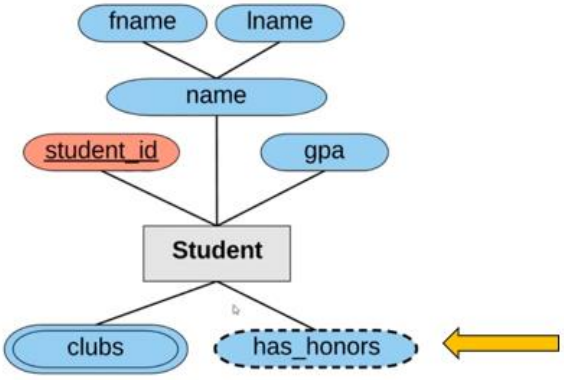


Figure 1. 6 Student Entity's derieved attribute

6) Relationship Attribute

Attributes can be related with relationships as well. This is a attribute of the relationship. For example, students that enroll in a class will receive a specific grade for that class. The only way for a student to get that grade is to enroll in that class. When converting an ER model to a Relational model, it is not suggested to assign attributes to relationships if they are not required. This is because things might become complicated, and we may need to create a separate table to describe the relationship.

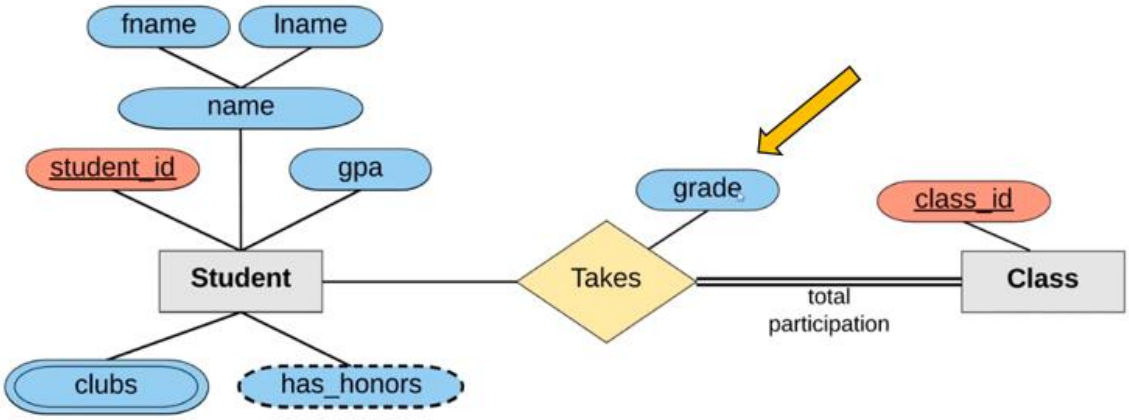


Figure 1. 7 Student Entity relationship with Class entity

7) **Weak Entity**

Weak entity sets are those with insufficient attributes to create a primary key, while strong entity sets are those with a primary key. Because weak entities lacking a primary key, they can't be identified on their own and must rely on another entity which is known as owner entity. In the ER Diagram, weak entities are represented by a double rectangular box, while identifying relationships are represented by a double diamond.

Ex:- Exam Entity, can't exist without a class. For Exam to exist it has been associated with a class.

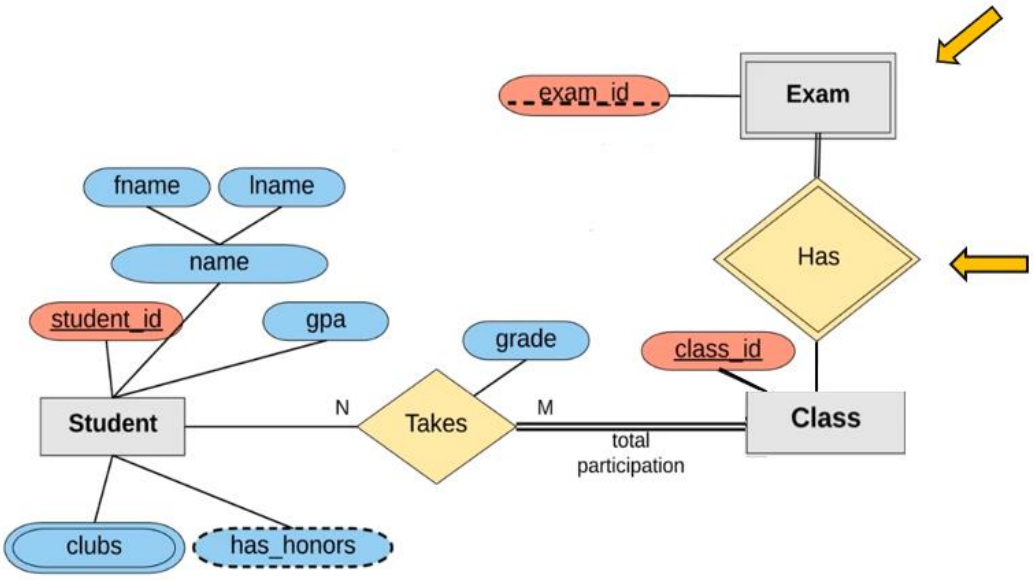


Figure 1. 8 Class entity's weak entity exam

1.2 Relationship Cardinality and Ordinality

Cardinality referred to as the maximum number of times an instance of one entity can be related to instances of another entity's instance.

Ordinality, on the other hand, is the minimum number of times an instance in one entity can be related to another entity's instance.

Cardinality, in other terms, indicates a fundamental relationship between two entities or objects. For example, many Students can take many Classes.

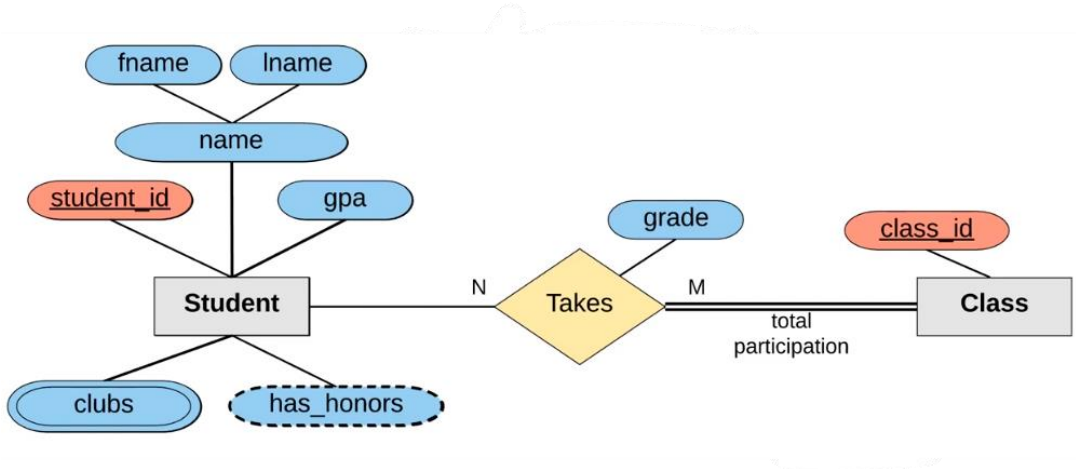


Figure 1. 9 Student Entity relationship with Class entity

Cardinality is represented by a crow's foot at the connector's ends in an ER diagram. One-to-one, one-to-many, and many-to-many are the three most common cardinal relationships.

1.2.1 One-to-One (1:1)

A one-to-one relationship is commonly used to separate an entity into two parts in order to provide information in a simple and understandable manner. A one-to-one relationship is represented in the figure below.

Ex:- 1 User can have 1 user profile

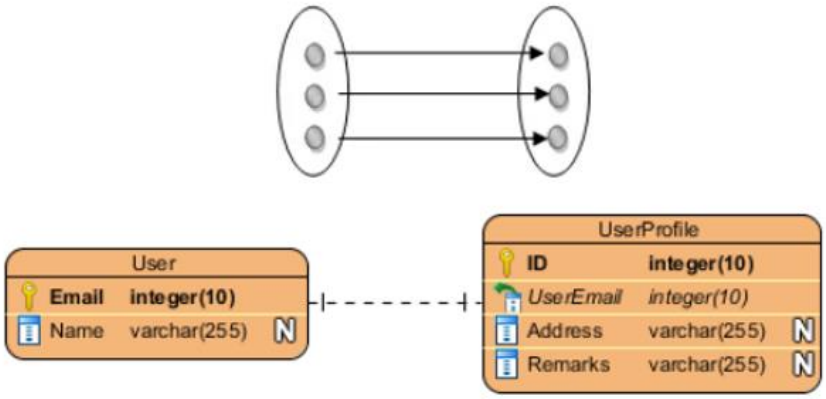


Figure 1. 10 One-to-One relationship cardinality

The main field in each table, Student ID, is designed to hold unique values in the example below. The Student ID field is the primary key in the students table, but it is a foreign key in the Contact Info table.

When the Student ID field in the Contact Info table and the Student ID field in the students table have the same value, this connection returns related records.

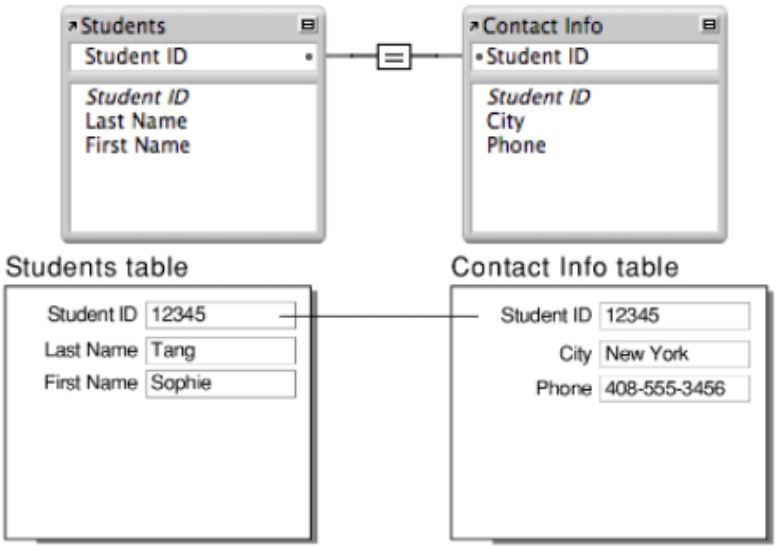


Figure 1. 11 One-to-One relationship cardinality explained with logical tables

1.2.2 One-to-Many (1: M)

One record in one table can be linked to one or more records in another table in a one-to-many relationship. For example, each customer, may have a large number of sales orders. A one-to-many relationship is represented in the figure below.

Ex:- 1 Department can have many Students

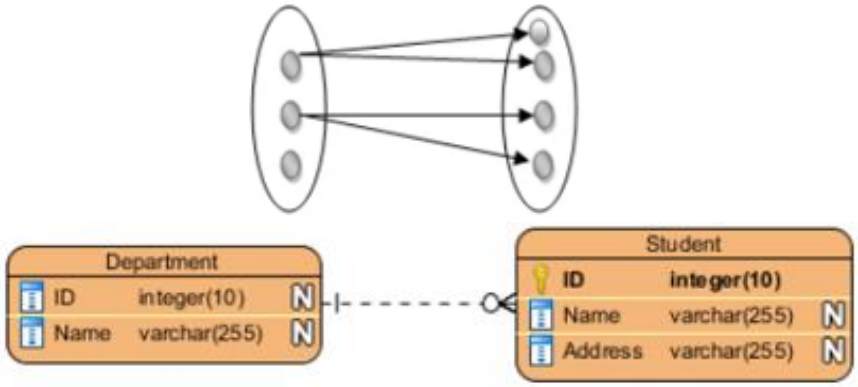


Figure 1. 12 One-to-Many relationship cardinality

As in the below example Customer ID, the primary key field in the Customers table, is designed to contain unique values in this example. Customer ID is a foreign key column in the Orders table that allows multiple instances of the same value. When the Customer ID field in the Orders table and the Customer ID field in the Customers table have the same value, this relationship returns related records.

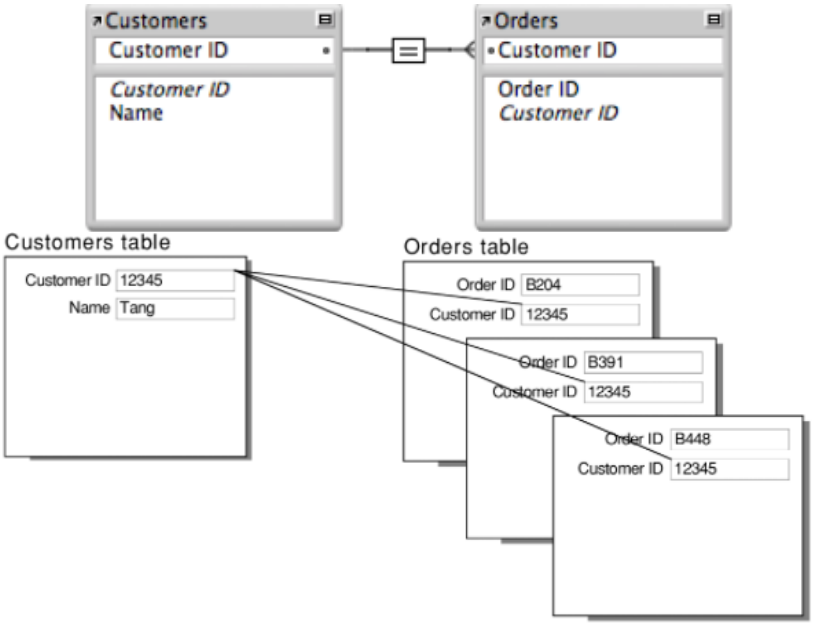


Figure 1. 13 One-to-Many relationship cardinality explained with logical tables

1.2.3 Many-to-Many (M: N)

When numerous records in one table are linked to several records in another table, this is known as a many-to-many relationship. Customers and products, for example, have a many-to-many relationship. Customers can buy many varieties of products, and products can be bought by a many customer.

In most relational database systems, we can't create a straight many-to-many link between two tables. When it comes to maintaining track of invoices, for example, if there were several invoices with the same invoice number and one of our clients inquired about it, we would have no idea which number they were referring to. This is one of the reasons why each invoice should have its own unique value.

To avoid this issue, we may use a third table called a join table to divide the many-to-many relationship into two separate one-to-many relationships. A matching field is included in each record in a join table, and it holds the value of the primary keys of the two tables it joins. These matching fields are foreign keys in the join table. As records in the join table are created from any table it joins, these foreign key fields are filled with data. For an example following below example representing with the joining table which is “Student Course table” to holds the value of the primary keys of the two both Student table and Course table it joins.

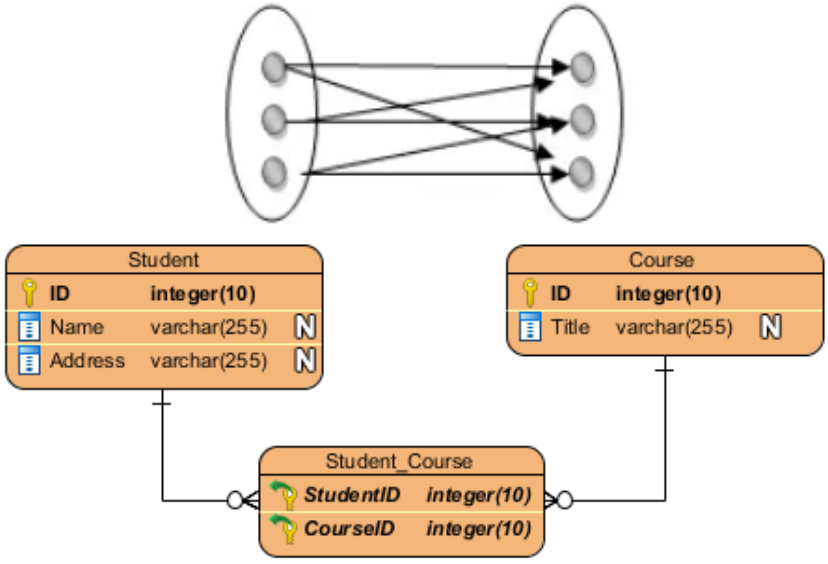


Figure 1. 14 Many-to-Many relationship cardinality

A relationship between students and classes is another example of a many-to-many relationship. A student can enrol in many classes, and a class can have many numbers of students. A Students table contains a record for each student, and a Classes table contains a record for each class in the following example. Enrolments is a join table that splits a many-to-many relationship into two independent one-to-many relationships.

Each student in the students table is individually identified by the primary key Student ID. Each class in the Classes table is uniquely identified by the primary key Class ID. The foreign keys Student ID and Class ID are found in the Enrolments table.

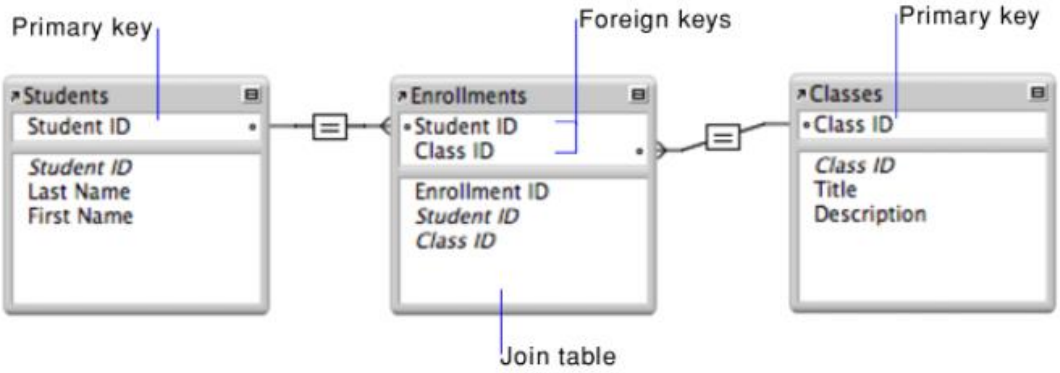


Figure 1. 15 Many-to-Many relationship cardinality explained with logical tables

1.3 User and System requirements to design above Polly Pipe Database

1.3.1 User Requirements

The term "user requirements" refers to a set of user standards that inform software about what the device should be able to do. The user specifications specification should be free of solution-oriented bias and use terms that are relevant to the user's problem area.

According to this assignment Polly Pipe is a company, situated in Braintree, England. Polly Pipe is a water sports provider and installer. They have a requirement to design and implement a database with a system that satisfies the information needs as mentioned in below.

According to the assignment brief Polly Pipe focuses on placing aquariums with business clients. Hence, they can have their own customer base which doing business. Business clients can request many installations, but each one is customized for a specific consumer. Hence customer should have ability to select manpower they need for per installation and they should have adjusted a time period for per installation.

The Polly Pipe user requirement clearly state types of facilities Polly Pipe serves are categorised. And every facility has one or more staff assigned to it. Hence these each facility can be served by hired employees such as carpenters, electricians, water installers and masons. Equipment such as Aquariums, air pumps, and thermostats, computers are used in the facility, and they should be Cleary mentioned with each facility according to user requirements.

According to above requirements it’s better to create the system for both admin and sales representatives. They should have the ability to control over the customer details. But it’s better to give authority of controlling employee, equipment, and facility types of details to admin since they are important entities that should be handle carefully. Finally, the sales representative could have to manage the installing according to the data which has provided by facility details and other entities.

The user requirements that applied to the Polly Pipe Aquariums system are summarised below.

1. Creating logging system for both admin and sales representative
2. Admin and Sales Representative could add new customer to the system
3. Admin could Add new employees with employee type to the system
4. Admin could add new equipment to the system
5. Admin could add new facilities to the system
6. Customer could request installation according to the facilities Polly Pipe System has.
7. Staff could assign employees to each installation.
8. Staff could set period of each installation.
9. User should have ability to move through forms easily

In addition, the system will have to adhere to the following user groups:

1. Admin
2. Sales Representative

1.3.2 System Requirements

System requirements are the minimum criteria that a device must meet in order to use specific hardware or software. These are usually provided by the system developer to the consumer. As my knowledge Polly Pipe must meet the following system requirements in order to design and implement a database with a system.

First there should have Entity Relationship diagram tool to get visual starting point for database architecture. And also, ERD determining information system requirements across an organization. For that I'm choosing "draw.io" application. Draw.io is a free diagramming application. It can be used to create flowcharts, network diagram software, ER diagrams, and can use to create database schema.

Then to get better understand of the created ERD there must have Logical database designer tool. For that I'm using "Lucidchart" webapplication. Lucidchart allows users to create and share professional flowchart diagrams for a variety of purposes, including brainstorming and project management. And with this application it's very easy to create logical database rather than using "draw.io" application. Since this is free and cloud-based application, can save work anytime anywhere.

To create a desktop application for the system I'm using Visual Studio. We can create data-centric, modern line of business applications for Windows using.NET and Visual Studio. Since Windows Forms creation with Visual Studio is provide a UI(User Interface) framework, with easy to use drag-and-drop designers, for building Windows desktop apps on the .NET platform.

Then to create database for the system I selected Microsoft SQL Server Management Studio since it can be used to quickly create or modify SQL database. And also, it can use to add database objects including tables, views, and stored procedures.

The system requirements that applied to the Polly Pipe Aquariums system are summarised below.

1. Draw.io application to create ERD.
2. LucidChart application to create logical database.
3. Visual Studio to create desktop system application.
4. Microsoft SQL Server Management Studio to handle database.

1.4 Designing Polly Pipe ERD and Logical Database

1.4.1 ERD with PK, Entities, and Cardinalities(Relationships)

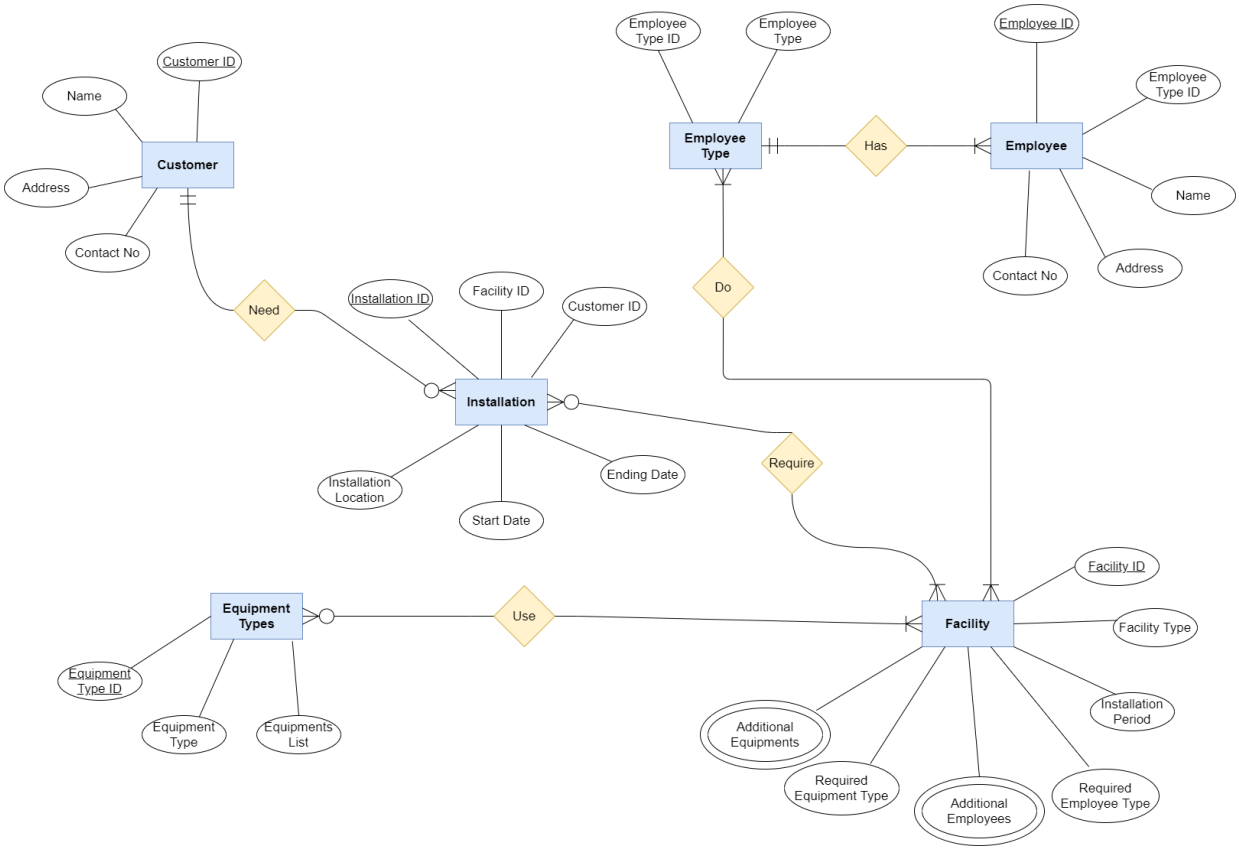


Figure 1. 16 ERD by "Draw.io" software

According to above ERD there are 6 entities being used. Since Polly Pipe to handle customer details there is a "Customer" entity. Then "Employee" entity to handle employee details. Each employee belongs to type of work they do. Hence, "Employee Type" entity created to handle employee type details.

Then I created entity named "Equipment Types" to handle equipment data and to keep track of their list details available. Then the "Facility" entity created to maintain the facilities Polly Pipe serves. This entity created to manage relevant employees, equipment details with approximate days to serve each facility. And finally created "Installation" entity to handle facility installations which customers require.

Facility entity is a is an important entity. Because Facility requires employees, equipment to produce an installation. Hence Facility entity is interrelated with 3 entities at least. Because of this Facility entity is important since its interconnecting with multiple entities. There are several other ways to create this ERD, but I preferred to do this in this simple manner since it's less complicated. By interconnecting several entities to facility entity make this less complicated. Otherwise, this ERD could be filled with bunch of multiple lines and with many relationships like a matrix.

When creating this ERD, to show relationship cardinalities I've chosen more sophisticated way such as "Crow Foot Model" rather than using simple conventional method such "Chen's Model". For example, let's consider relationship with customer and installation. When considering customer side of the relationship, to exist installation there must at least 1 customer. But for an installation there could not have many customers. That means an installation is needed by only 1 customer. Because of that I used "One and Only One" crow foot natation for the customer side of the relationship. Then when considering the Installation side, sometimes none of customers could need an installation. Or 1 customer could need several installations. Hence, I used "Zero or Many" crow foot natation for the Installation side relationship. Likewise for other relationships I continued with this method.

1.4.2 Converting ERD to Logical Database design

The entities and relationships represented in an ERD are focused on the requirements of the business. The necessity to satisfy database design has not yet been considered. The real design of a database is represented by a logical database. It is concerned with the transformation of a logical design into a schema-level design that will be turned into a relational database.

When modelling a logical database, ERD is used as the foundation, and then primary keys, foreign keys, and constraints are defined. Relationships are sometimes resolved by adding new tables, such as a Linked table for a many-to-many relationship.

Because the ERD and logical database represent the business requirement and database schema, comparing them both helps to identify the differences, ensuring that the database follows the initial business's user requirements regardless of modifications.

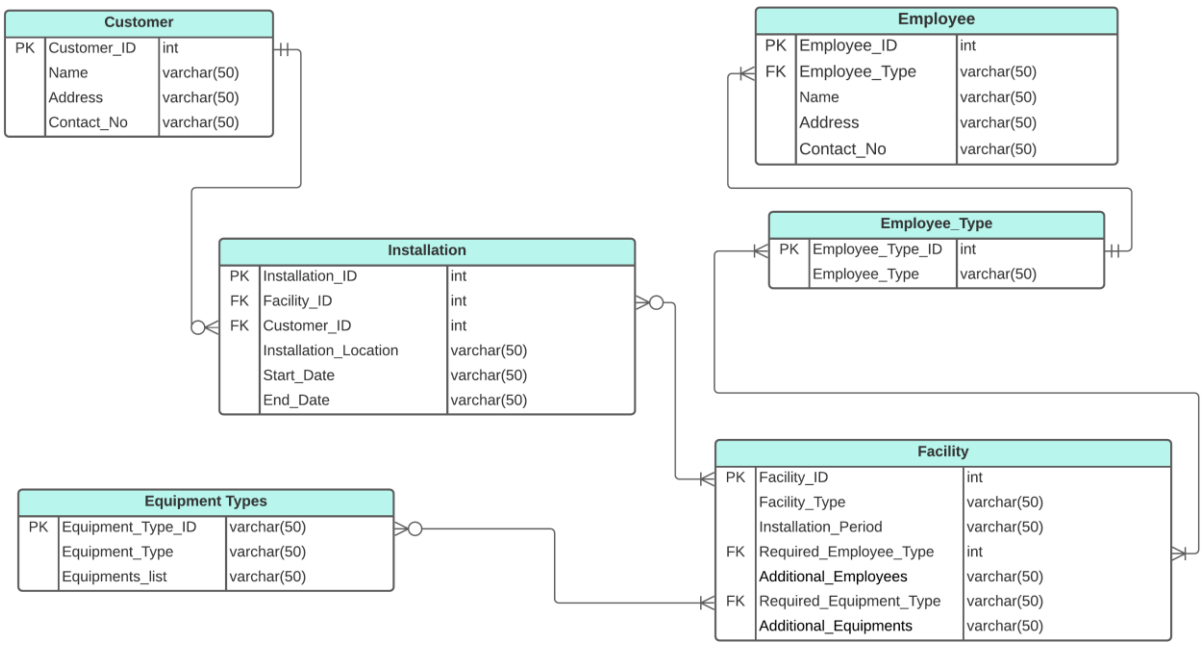


Figure 1. 17 Logical Database by "LucidCharts" web application

1.5 Normalization

1.5.1 What is Normalization

One of the most important considerations when building the schema of a relational database is to ensure that duplication is minimized. This serves two purposes:

- Reducing the amount of storage required for data storage.
- Avoiding unnecessary data conflicts that may arise as a result of storing numerous copies of the same data.

Normalization is a method of separating data into numerous related tables in order to reduce data redundancy. Repetition of same data in different locations is known as data redundancy. The size of the database grows as data is repeated. Database normalization is a technique that supports in the creation of an ideal database structure. Instead of copying data, database normalization divides tables into smaller sub tables and stores pointers to the data.

For a better understanding of DBMS (Data Base Management System) Normalization, let's refer below simple table.

Course	Course Venue	Lecturer	Lecturer Phone No.
Physics	Lab	Sanath	7735434356
Bio	Main Hall	Viani	31 2254354
Chem	Auditorium	Kapila	1234
Analytical Chem	Auditorium	Kapila	1234

Figure 1. 18 Sample table to explain what normalization is

This above table design appears to be fine at first sight. When we need to change information, unfortunately, problems occur. Consider what would happen if Lecturer Kapila's phone number changed. In this case, we'll have to make changes in two areas. Similarly, excessive data repetition expands the database's size.

Data redundancy not only causes database size issues, but it also causes several anomalies also.

1. Insertion Anomaly
2. Deletion Anomaly
3. Updating/Modification Anomaly

1.6 Types of Anomalies

1.6.1 Insertion Anomaly

Let's refer the below table to understand what insertion anomaly is. If we have to add another Chem subject with same Lecturer (Kapila), you have insert Kapila's data again. If you have added 10 Chem subjects with same Lecturer (Kapila), you have repeat insert Kapila's data 10 times. This leads to Insertion Anomaly. As a result, Insertion Anomaly refers to the process of entering redundant data for each new row.

Course	Course Venue	Lecturer	Lecturer Phone No.
Physics	Lab	Sanath	7735434356
Bio	Main Hall	Viani	31 2254354
Chem	Auditorium	Kapila	1234
Analytical Chem	Auditorium	Kapila	1234
Practical Chem	Auditorium	Kapila	1234
Physical Chem	Auditorium	Kapila	1234
Biomedical Chem	Auditorium	Kapila	1234

Figure 1. 19 Sample table to explain what insertion anomaly is

1.6.2 Deletion Anomaly

Let's refer the below table to understand what deletion anomaly is. If we delete the Course Venue information, along with it the Lecturer information will be deleted. When we erase all of the Course Venue information, we mistakenly delete all of the Lecturer's details as well. Deletion Anomaly is the result of this. As a result, Deletion Anomaly refers to the loss of a related dataset as a result of the deletion of another dataset.

Course	Course Venue	Lecturer	Lecturer Phone No.

Figure 1. 20 Sample table to explain what deletion anomaly is

1.6.3 Updating/Modification Anomaly

Let's refer the below table to understand what updating anomaly is. We'll have to update the table multiple times if Chemistry Lecturer is replaced by another Lecturer. If a single row is missed, the data becomes inconsistent. Updating Anomaly is the result of this. Hence, Updating Anomaly refers data inconsistency caused by data redundancy and a partial updating.

Course	Course Venue	Lecturer	Lecturer Phone No.
Physics	Lab	Sanath	7735434356
Bio	Main Hall	Viani	31 2254354
Chem	Auditorium	Kapila -> Amal	1234 -> 0777
Analytical Chem	Auditorium	Kapila -> Amal	1234 -> 0777
Practical Chem	Auditorium	Kapila -> Amal	1234 -> 0777
Physical Chem	Auditorium	Kapila -> Amal	1234 -> 0777
Biomedical Chem	Auditorium	Kapila -> Amal	1234 -> 0777

Figure 1. 21 Sample table to explain what updating/modification anomaly is

1.6.4 The way how Normalization helps to solve this data redundancy issues

As a result of data redundancy, in normalization will divide the existing table into sub tables. We can divide above table into Course Detail Table and Lecturer Detail Table as below.

Course	Course Venue	Lecturer	Lecturer Phone No.
Physics	Lab	Sanath	7735434356
Bio	Main Hall	Viani	31 2254354
Chem	Auditorium	Kapila	1234
Analytical Chem	Auditorium	Kapila	1234
Practical Chem	Auditorium	Kapila	1234
Physical Chem	Auditorium	Kapila	1234
Biomedical Chem	Auditorium	Kapila	1234

Figure 1. 22 In normalization divide the existing table into sub tables

In order to interconnect these two tables, we can add Lecturer ID column as below table illustrate.

Course Details		
Course	Course Venue	Lecturer ID
Physics	Lab	1
Bio	Main Hall	2
Chem	Auditorium	3
Analytical Chem	Auditorium	3
Practical Chem	Auditorium	3
Physical Chem	Auditorium	3
Biomedical Chem	Auditorium	3

Lecturer Details		
Lecturer ID	Lecturer	Lecturer Phone No.
1	Sanath	7735434356
2	Viani	31 2254354
3	Kapila	1234

Figure 1. 23 Interconnecting two sub tables

So according to this solution if we wanted to add another 5 Chemistry Lecturers to Course Details Table, we don't have to repeat lecturer Kapila's data again and again. Only we have to insert No.3 to the table. If we delete all Course Details Table data, Lecturer data will be not deleted.

As a result, normalization does not mean the elimination of data redundancy. Normalization is about minimizing Data Redundancy.

1.7 Normalization Types

As we learned so far, we can break down normalization as below. We study normalization because database complexity rises as data requirements rise. As a result, the necessity for Normalization has grown. The process of structuring data in a database is known as normalization.

Normalization is a technique for reducing the amount of redundancy in a relation or group of relations. It's also used to get rid of annoying attributes like Insertion, Update, and Deletion Anomalies. Normalization breaks the larger table into smaller tables and uses relationships to connect them. The usual form is used to eliminate database table redundancy. The database normalization procedure is further divided into the several types as given in the figure below.

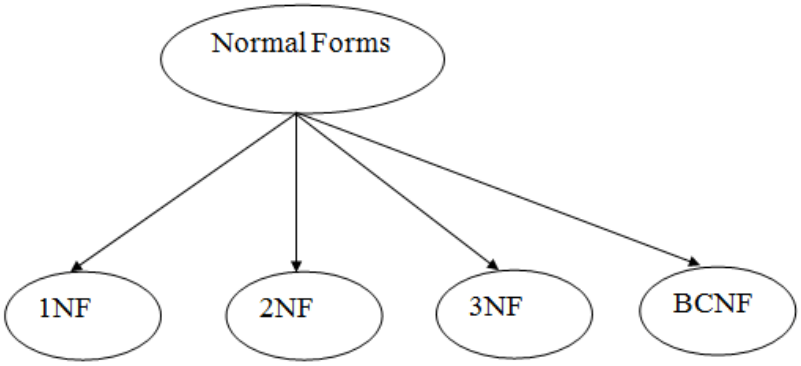


Figure 1. 24 Types of Normalization

The types of normalization are

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (BCNF / 3.5NF)

1.7.1 First Normal Form (1NF)

The first normal form states that each table cell must contain only one value. It is considered poor database design if the database is not even in its first normal form. As we can see, 1NF is a necessity. To be in 1NF, we must follow 4 essential rules.

Rule no 1 → Atomic numbers should be present in each column (single Value). As in the highlighted cell in below figure, this rule is broken when we put Math and Commerce in the same cell.

Anomaly

Student	Subject
Ryan	Bio
Judith	Maths , Commerce

Solution

Student	Subject
Ryan	Bio
Judith	Maths
Judith	Commerce

Figure 1. 25 First Normal Form (1NF) rule no 1 explanation

Rule no 2 → Values of the same kind should be in the same column. As a result, do not mix multiple types of values in the same column. As given below figure only one data type should be present in the subject field. This rule is broken when a variable contains both a string and an integer.

Student	Subject
Ryan	Bio
Judith	2020

Figure 1. 26 First Normal Form (1NF) rule no 2 explanations

Rule no 3 → Each column should be given a distinct name. As a result, similar names cause confusion while retrieving data.

Student ID	Name	Name
1	Ryan	Wickramaratne
2	Judith	Fernando

Student ID	F_Name	L_Name
1	Ryan	Wickramaratne
2	Judith	Fernando

Figure 1. 27 First Normal Form (1NF) rule no 3 explanations

Rule no 4 → It doesn't matter if the data is in sequential manner. Because SQL queries allow us to retrieve data from a table in any sequence.

Student ID	F_Name	L_Name
2	Judith	Fernando
3	K	Nicz
1	Ryan	Wickramaratne

Figure 1. 28 First Normal Form (1NF) rule no 4 explanations

1.7.2 Second Normal Form (2NF)

The following two conditions must be met for a table to be in second normal form:

1. The table should be in the 1NF format.
2. The table should not contain any partial dependencies. (The table's primary key should consist of exactly one column.)

What is Primary Key (PK)?

A primary key is a column or set of columns that uniquely identifies each row in a table. Any data row from the table can be retrieved using the primary key. For example, I may say, "Give me the details of Student ID =1." As a result, I'll be able to get all of the information I need.

Similarly, the Student ID determines every column in the table (PK). This explains the definition of dependency.

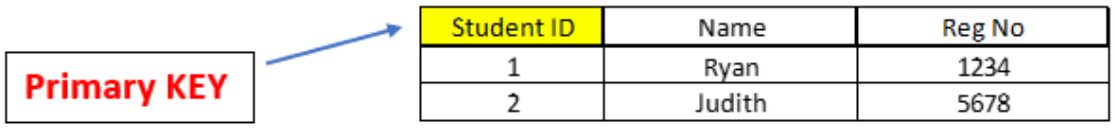


Figure 1. 29 Primary Key of a table

What is Composite Key?

Let's understand Composite Key with an example. When referring below table, none of the Student IDs or Subject IDs in this table are unique. For instance, Student ID 1 appears twice in this table, each with a different Subject ID.

If someone ask to give details of Student ID =1, which row should send? 1 or 2? And if someone ask to get Marks of Subject ID CHEM, which one to get? If someone again ask to give details of Subject ID =CHEM, which column SQL should send? 1 or 3? This makes so much confusion.

Student ID	Subject ID	Marks	Teacher
1	CHEM	70%	Viani
1	BIO	60%	Kapila
2	CHEM	2%	Viani

Student ID	Subject ID	Marks	Teacher
1	CHEM	70%	Viani
1	BIO	60%	Kapila
2	CHEM	2%	Viani

Figure 1. 30 Table without a distinct primary key

However, because a student cannot enrol in the same course twice, the tuple (Student ID, Subject ID) is unique. As a result, the PK for the database is formed by combining these two columns (Student ID + Subject ID). As a result, combining the Student ID and Subject ID columns yields a more meaningful PK. Composite Key is the name given to this combination of PKs. Hence composite key is a primary key made up of many columns that is used to uniquely identify a record.

Student ID	Subject ID	Marks	Teacher
1	CHEM	70%	Viani
1	BIO	60%	Kapila
2	Maths	2%	Viani

Figure 1. 31 Table with highlighted distinct composite key

If someone asks for information about Student ID = 1 and Subject ID = CHEM, SQL will almost certainly send the first row. When there is a M: M relationship, this happens. (This indicates that one student may have few numbers of subjects or 1 subject has a few numbers of students).

What is Partial Dependency?

As seen in the table above, marks are based on both the Student ID and the Subject ID. As a result, Student ID + Subject ID represents PK. When it comes to teacher, however, only the Subject ID matters (1 part of PK). It is not the student's fault. This is what partial dependency is all about. Partial Dependency should not be present in a table.

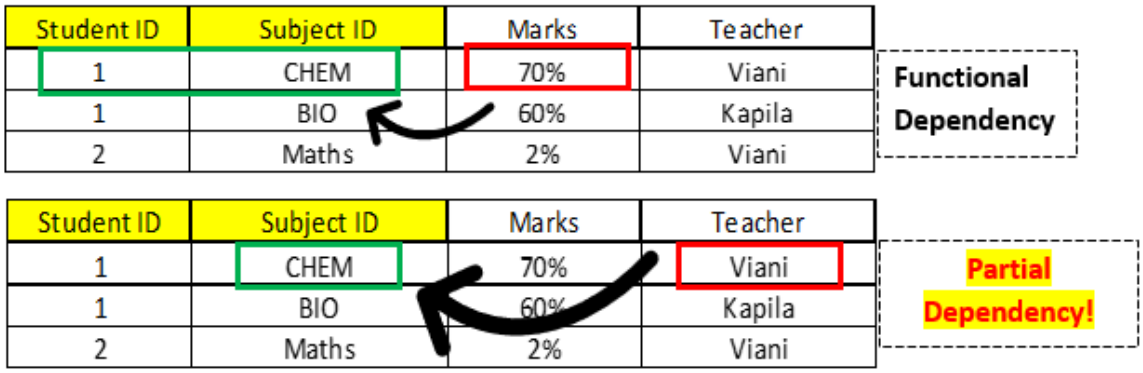


Figure 1. 32 Functional dependency vs partial dependency

To fix this Partial Dependency, the best option is to separate teacher column and add it to separate subject table as below. Then these 2 separate tables should interconnect with a foreign key. For this below table the foreign key is subject ID.

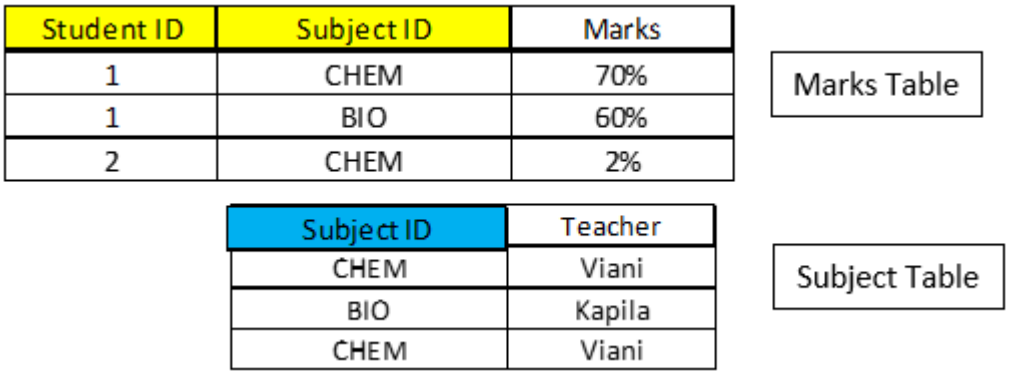
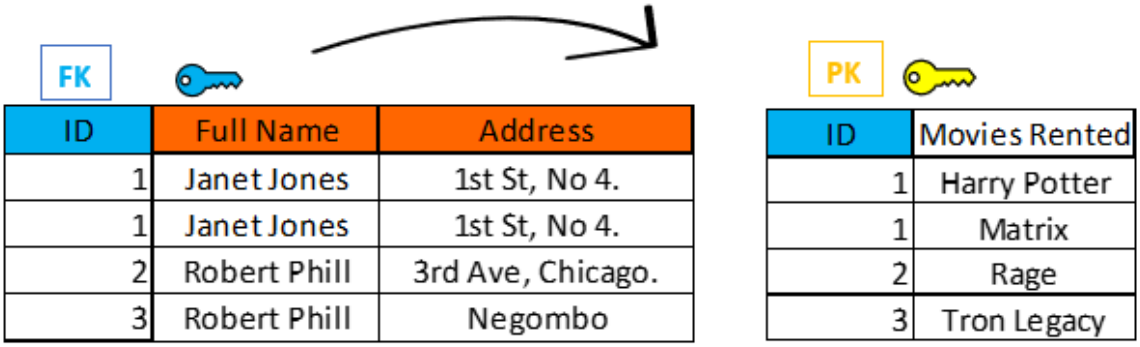


Figure 1. 33 Fixing Partial Dependency

What is Foreign Key (FK)?

In a relational database, a foreign key is a column (or set of columns) that links data between tables. For another table, the Foreign Key becomes the Primary Key.

- Unlike the primary key, a foreign key might have a different name than its primary key.
- It assures that rows in one table have matching rows in another.
- They do not have to be unique, unlike the primary key. They aren't always unique.
- Unlike primary keys, foreign key can be null (without values).



ID	Full Name	Address
1	Janet Jones	1st St, No 4.
1	Janet Jones	1st St, No 4.
2	Robert Phill	3rd Ave, Chicago.
3	Robert Phill	Negombo

ID	Movies Rented
1	Harry Potter
1	Matrix
2	Rage
3	Tron Legacy

Figure 1. 34 How foreign key being a primary key in another table

1.7.3 Third Normal Form (3NF)

The following two conditions must be met for a table to be in third normal form:

1. The table should be in the 2NF format.
2. There should be no Transitive Dependencies.

What is Transitive Dependency?

When a non-key column is changed, it may cause any of the other non-key columns to change, which is known as a transitive functional dependency. Let's understand with an example.

Depend only on non-key column

Student ID	Subject ID	Marks	Exam Name	Maxiumum Marks
1	CHEM	70%	Practical	500
1	BIO	60%	Theory	1000
2	Maths	2%	Attendence	100

Figure 1. 35 Transitive Dependency explaining example table

Marks are dependent on Composite Key, as we discussed in 2NF. Exam Name is also determined by the Composite Key. However, the Maximum Marks for each exam are independent of the Composite Key. Exam Name, which is a non-Key Column, determines the maximum marks.

As an example, only 500 marks maximum are allowed for the Practical Exam. If we alter the name of the exam, we'll need to adjust the maximum marks as well. (The easiest way to check for dependencies is to use this approach). As a result, the value of Maximum Marks varies depending on the exam.

Because of that, the non-key attribute Total Marks is dependent on another non-key attribute. To overcome this, we can create a new table with the Exam name and Total Marks.

Student ID	Subject ID	Marks	Exam Name
1	CHEM	70%	Practical
1	BIO	60%	Theory
2	Maths	2%	Attendence

Student Table

Exam Name	Maxiumum Marks
Practical	500
Theory	1000
Attendence	100

Exam Table

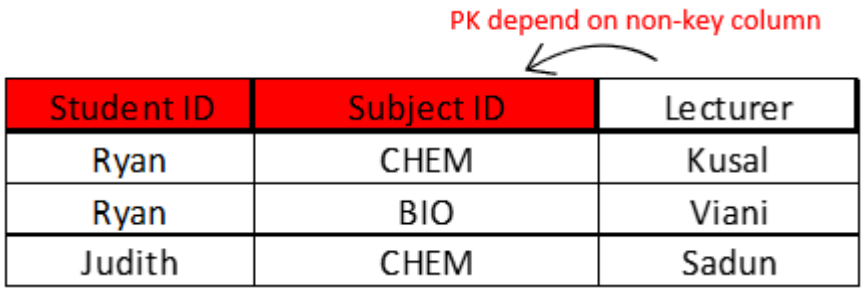
Figure 1. 36 Fixing Transitive Dependency

1.7.4 Boyce-Codd Normal Form (BCNF / 3.5NF)

The Boyce-Codd Normal Form, also known as 3.5 Normal Form, is an extension of the third normal form. The following two conditions must be met for a table to be in Boyce-Codd normal form:

1. The table should be in the 3NF format.
2. "A" should be a super key for any A → B dependent.

The second point sounds a bit tricky but in simple words it means, that for a dependency A → B, "A" cannot be a non-prime attribute, if "B" is a prime attribute. Let's understand this with referring below example table.



Student ID	Subject ID	Lecturer
Ryan	CHEM	Kusal
Ryan	BIO	Viani
Judith	CHEM	Sadun

Figure 1. 37 Boyce-Codd Normal Form explaining example table

Lecturer Kusal depend on both Ryan and CHEM (Kusal is Ryan's Lecturer and Kusal is CHEM Lecturer). But in this table the Subject is dependent on Lecturer also (CHEM is dependent on Kusal.). But subject is a Primary Key. This shouldn't be happened.

So, in order to fix this problem, we have to sperate Subject and Lecturer columns. When separating them, it is best to consider relationship cardinality. When we take relationship cardinality Subject and Lecturer, it is 1:M. Which means 1 subject can have many Lecturers. But 1 Lecturer can teach 1 subject. So, when separating those columns, the best way is creating separate Lecturer table. Then we can add Subject column in Lecturer table. So, the final step is making relationship between those 2 tables with a foreign key as usual. So, this how Boyce-Codd Normal Form used to fix this kind of problems.

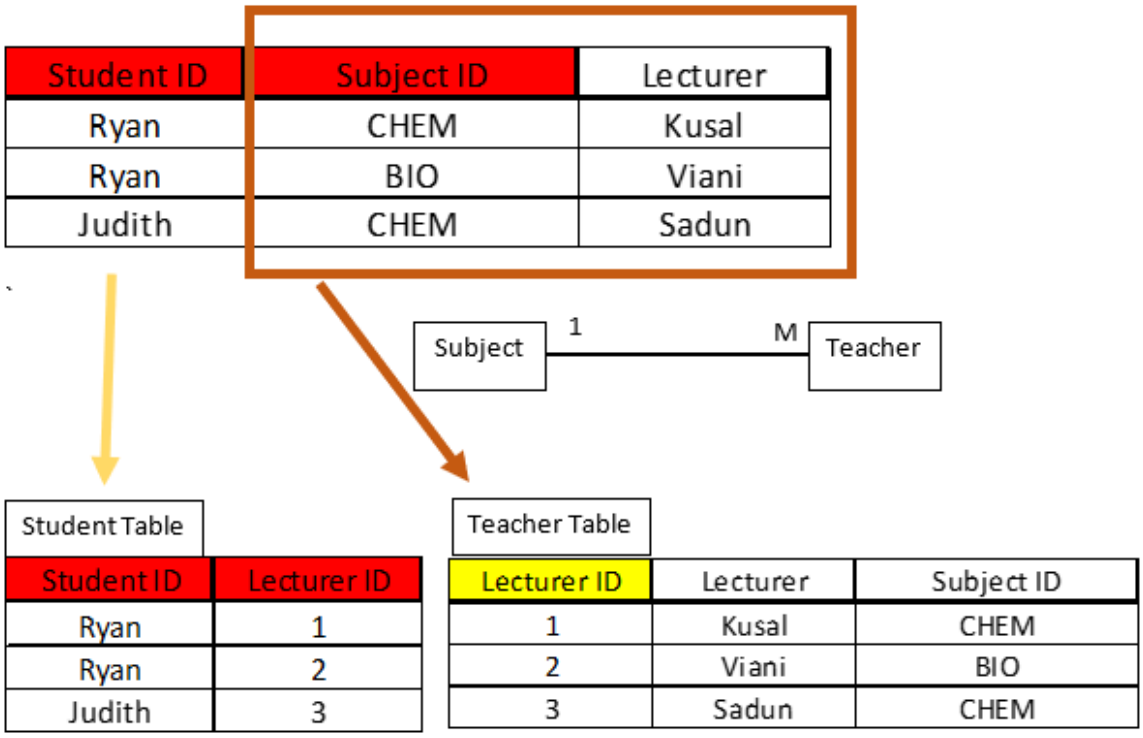


Figure 1. 38 How Boyce-Codd Normal Form being used

1.7.5 How I used First Normal Form (1NF) Normalization for this Polly Pipe project

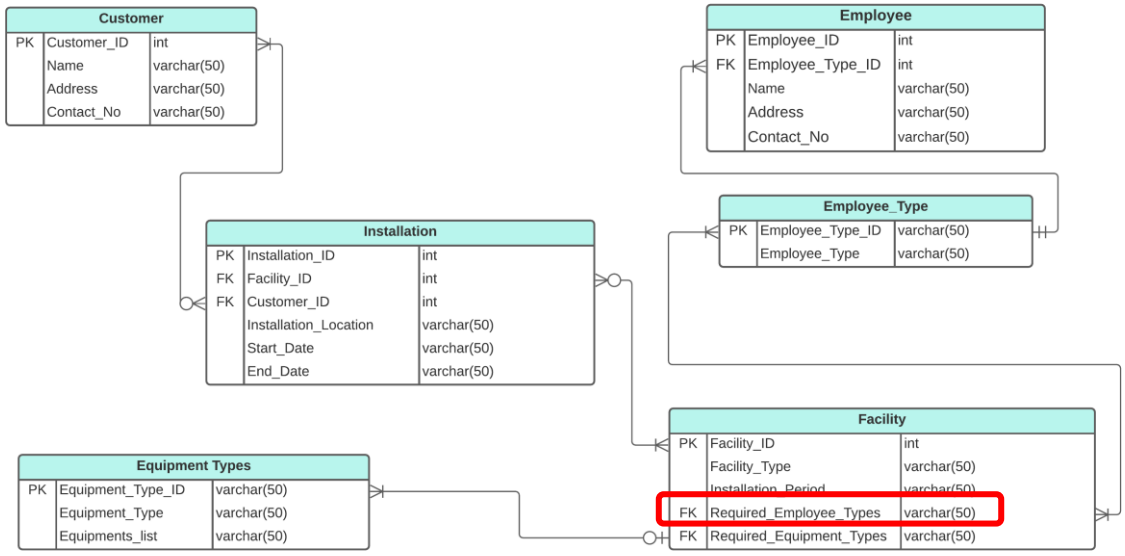


Figure 1. 39 Sample logical database I created later (highlighted area is the problem part)

The above figure illustrates the logical database I created earlier. For each Facility I needed to assign several employees. But according to First Normal Form, in a table each cell should contain 1 value. Since “Required Employee Types” column is a foreign key which referencing to Employee table, entering several employees in “Required Employee Types” column make complications. So likewise, below figure explains I had to enter values 1 by 1 in each cell.

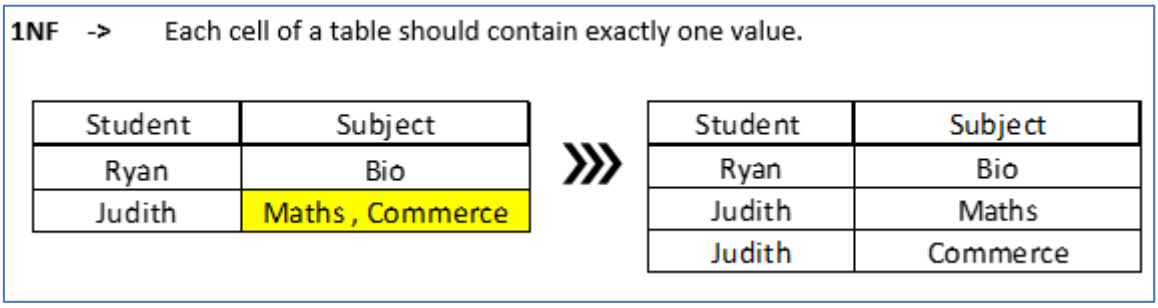


Figure 1. 40 First Normal Form simple summery

Because of that I decided to enter only 1 value to “Required Employee Types” column. But in order to fulfill my expectation, which is to assign several employees in Facility table, I created another column named “Additional Employees”. In this column I’m expected to enter several employee types as a description like entering an address. When we are entering an address, we state several attributes in a single column. Such as number of the house, street name, city name and so on.

Likewise, to enter multiple employee types in a single column, I created another column named “Additional Employees” in Facility Table as in below figure shows. Below figure illustrate the final output of my logical database. Since “Required Employee Types” column is a foreign key referencing to Employee table, I’m only give 1 value to that column. But in the “Additional Employees” column I’m expected to give several values as a description. I used this same method for the “Required Equipment Type” column. To enter multiple equipment into single cell as a description I created another column named “Additional Equipment” in Facility Table .

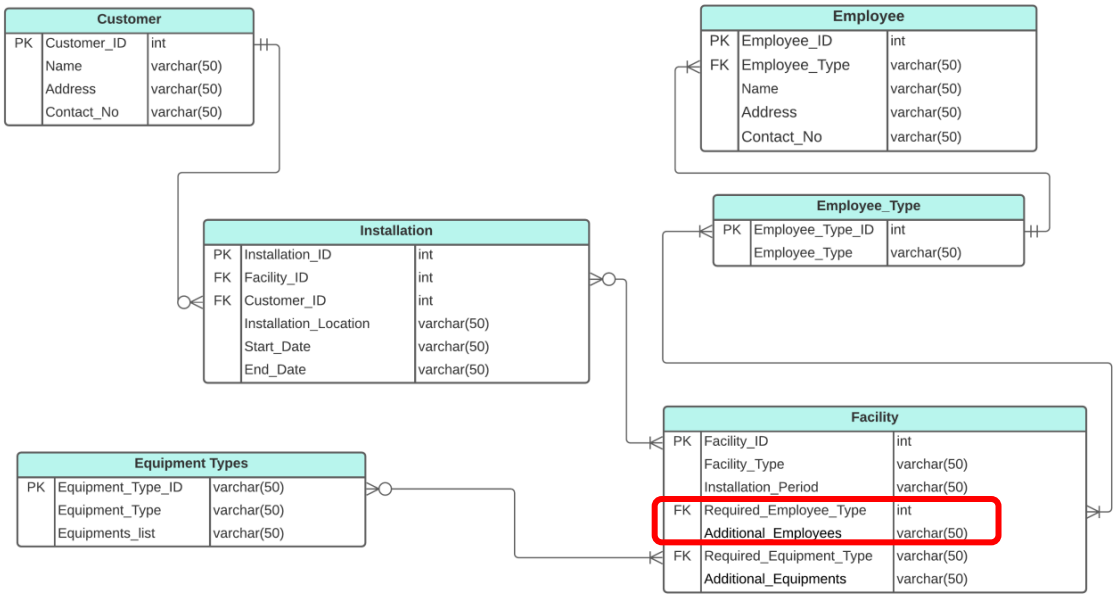


Figure 1. 41 Final output of logical database I created after using 1NF (highlighted area is the problem fixed part)

1.7.6 How I used Third Normal Form (3NF) Normalization for this Polly Pipe project

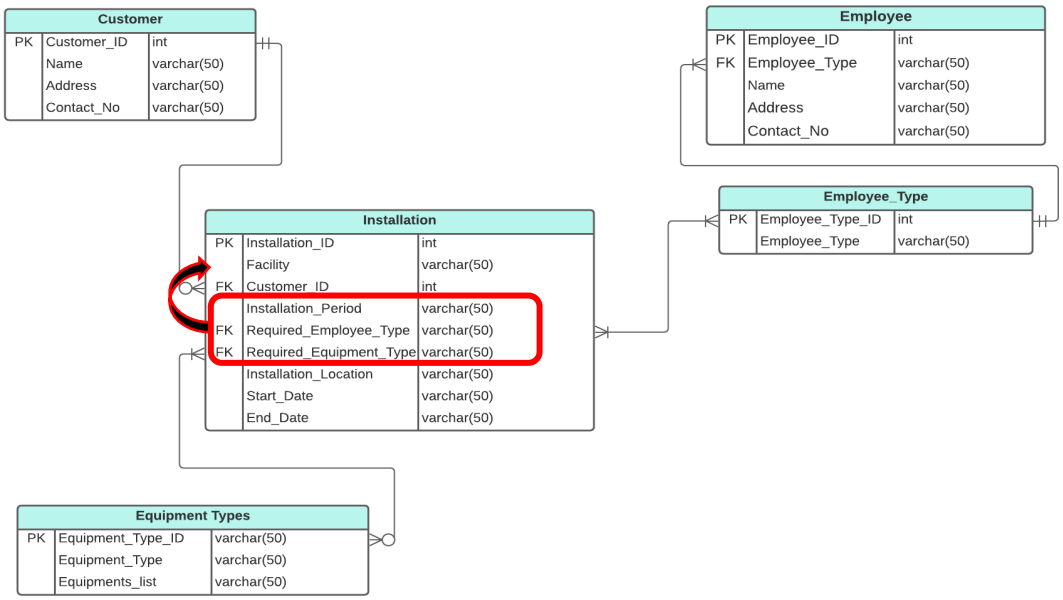


Figure 1. 42 Sample logical database I created first (highlighted area is the problem part)

The above figure illustrates the logical database I created first. According to that in the Installation table the highlighted columns of above figure in only depend on “Facility” column which resulting Transitive Dependency. Which means non-key column depending on another non-key column. According to Third Normal Form there should not have Transitive Dependencies. Below figure simply explains how to use Third Normal Form to fix the Transitive Dependencies problems.

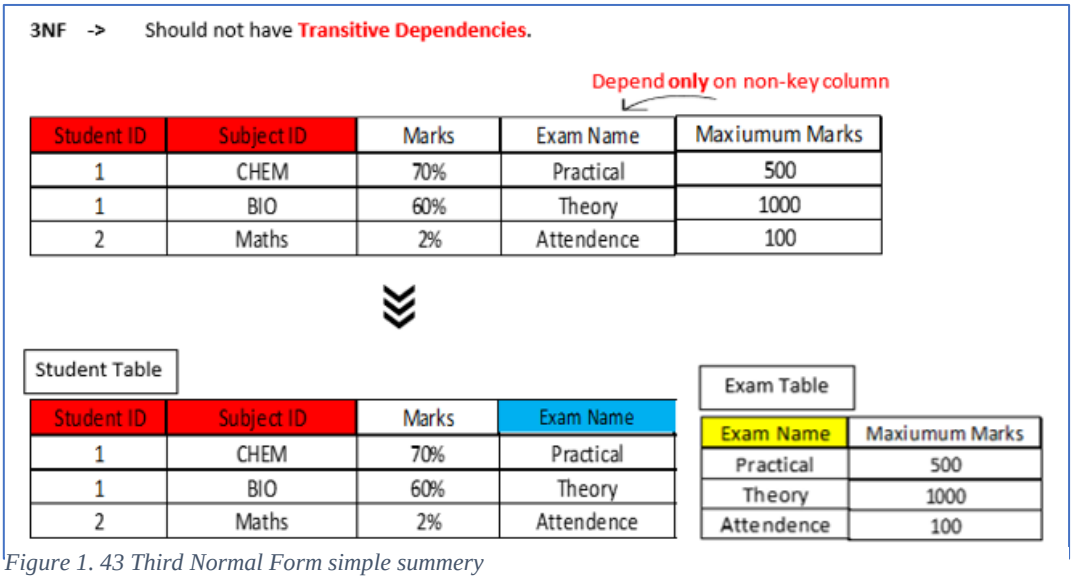


Figure 1. 43 Third Normal Form simple summery

Since 3 non-key columns such as “Installation Period”, ”Required Employee Type”, ” Required Equipment Type” are dependent on non-key column which is “Facility” I simply separated the Installation Table in to 2 parts. One table is “Installation” Table and other one is “Facility” table. As below figure shows in Facility table, I included the 3 non-key columns which influenced to occur Transitive Dependency. Then finally to interconnect these 2 tables I used Facility ID as a foreign key.

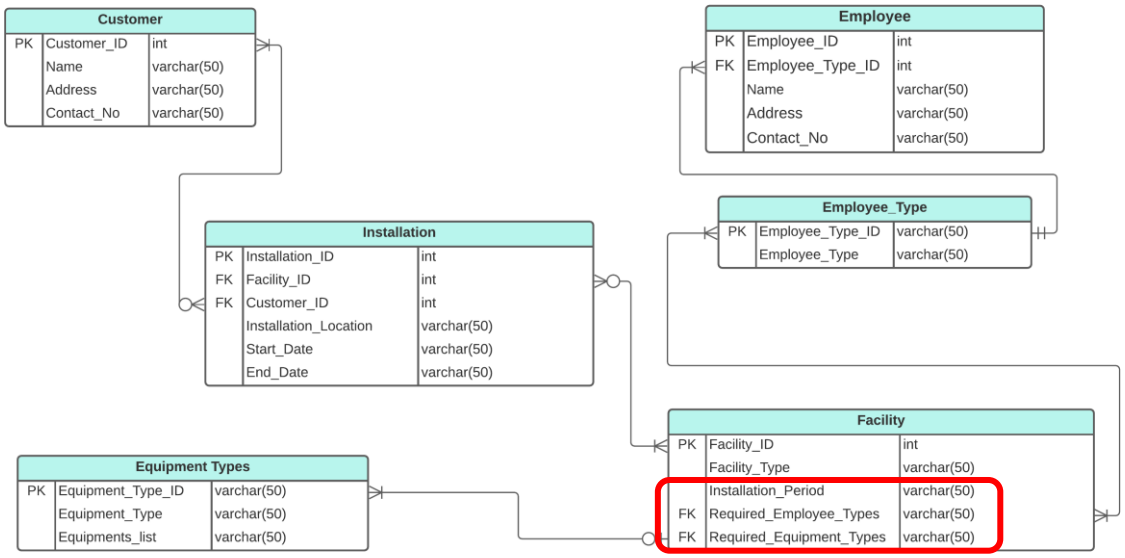


Figure 1. 44 Final output of logical database I created after using 3NF

1.8 Designing the system

1.8.1 Interface of the Systems

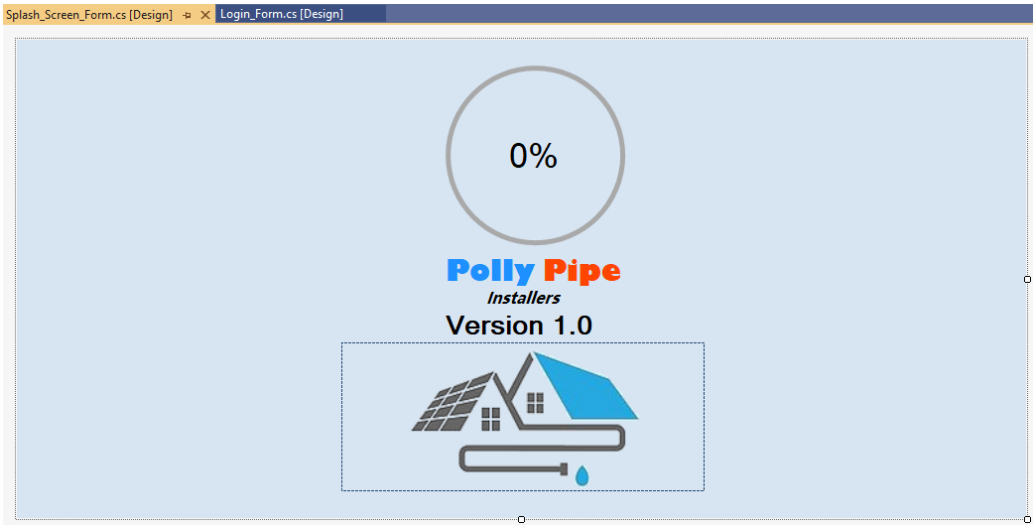


Figure 1. 45 Splash Screen form design

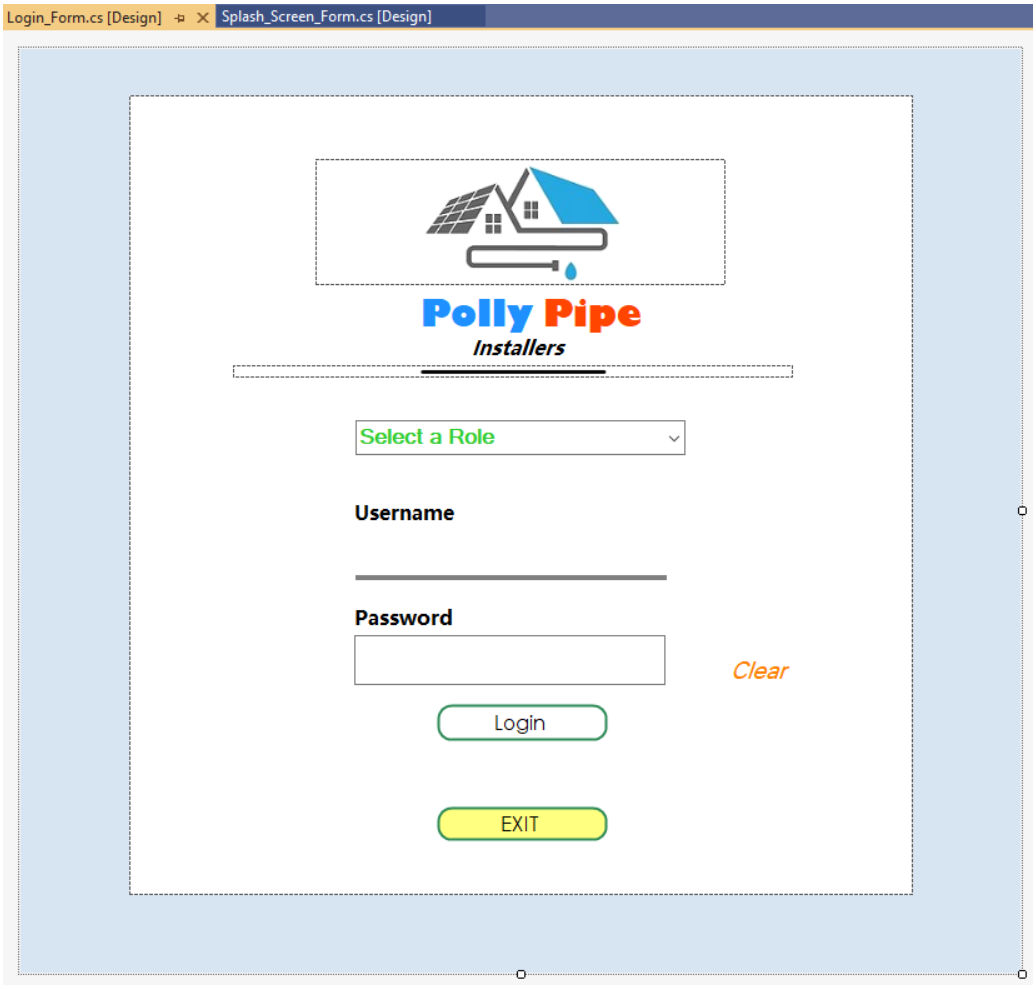


Figure 1. 46 Login form design

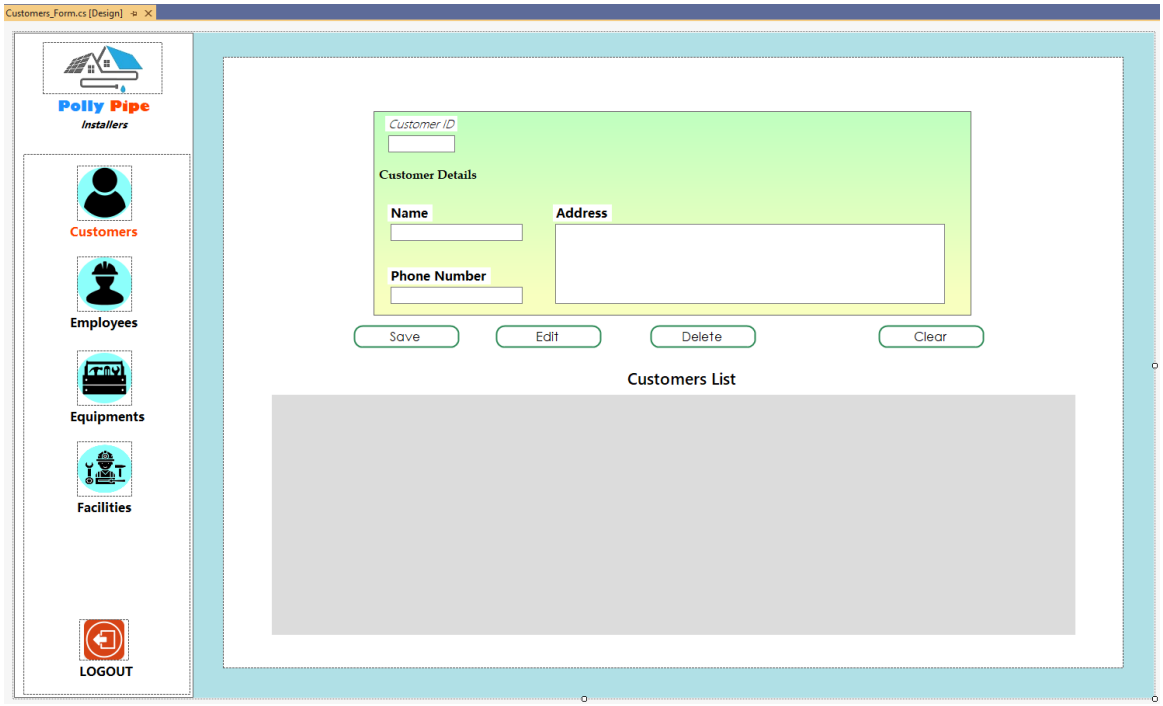


Figure 1. 47 Customers from design (Can view by admin sales representative)

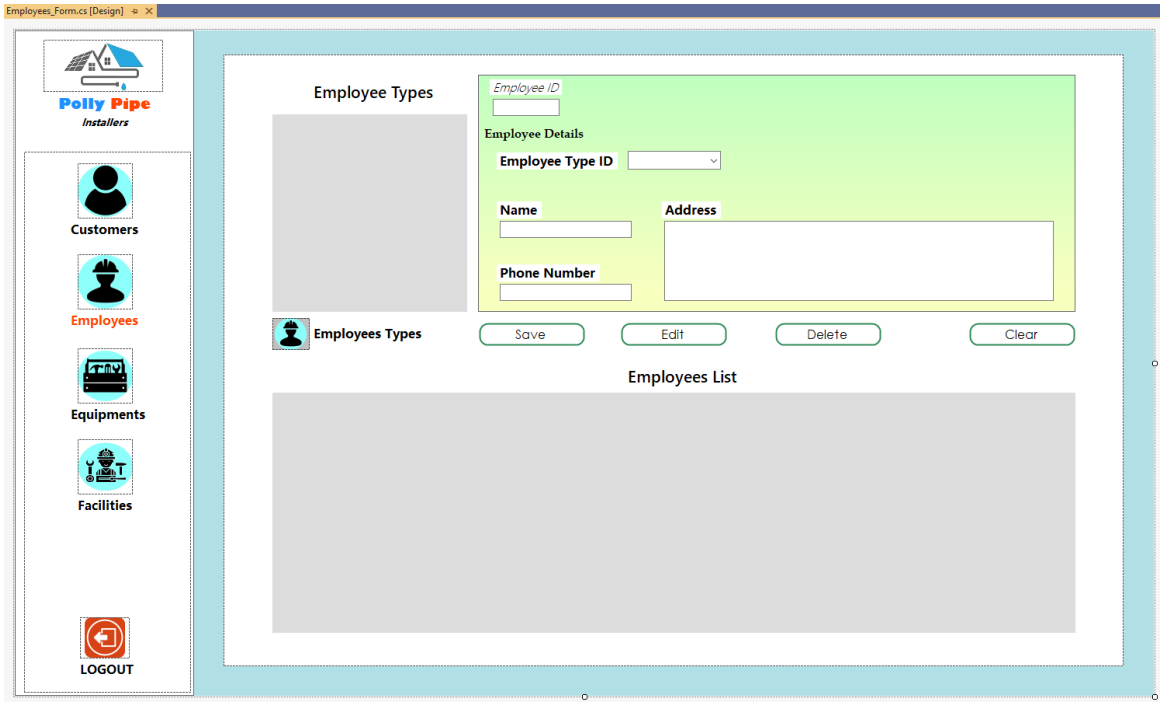


Figure 1. 48 Employees from design (Can view by admin)

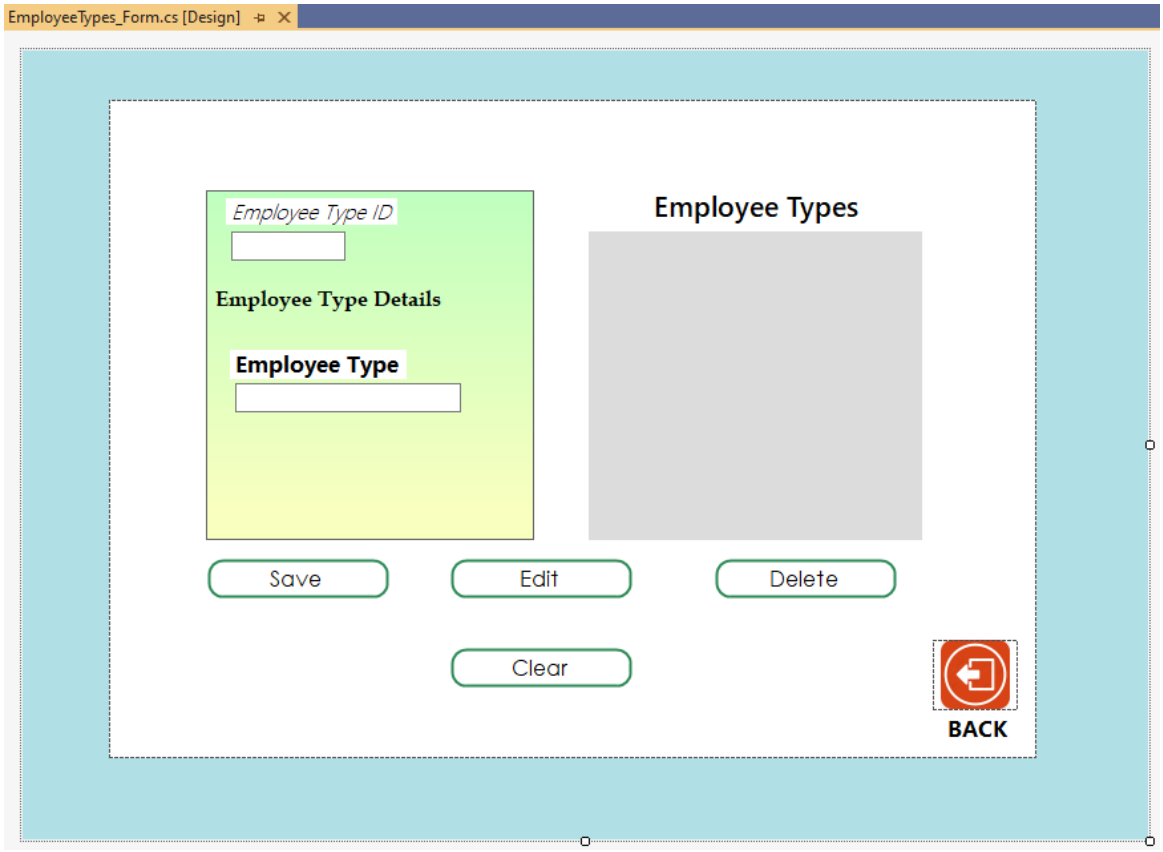


Figure 1. 49 Employees Types from design (Can view by admin)

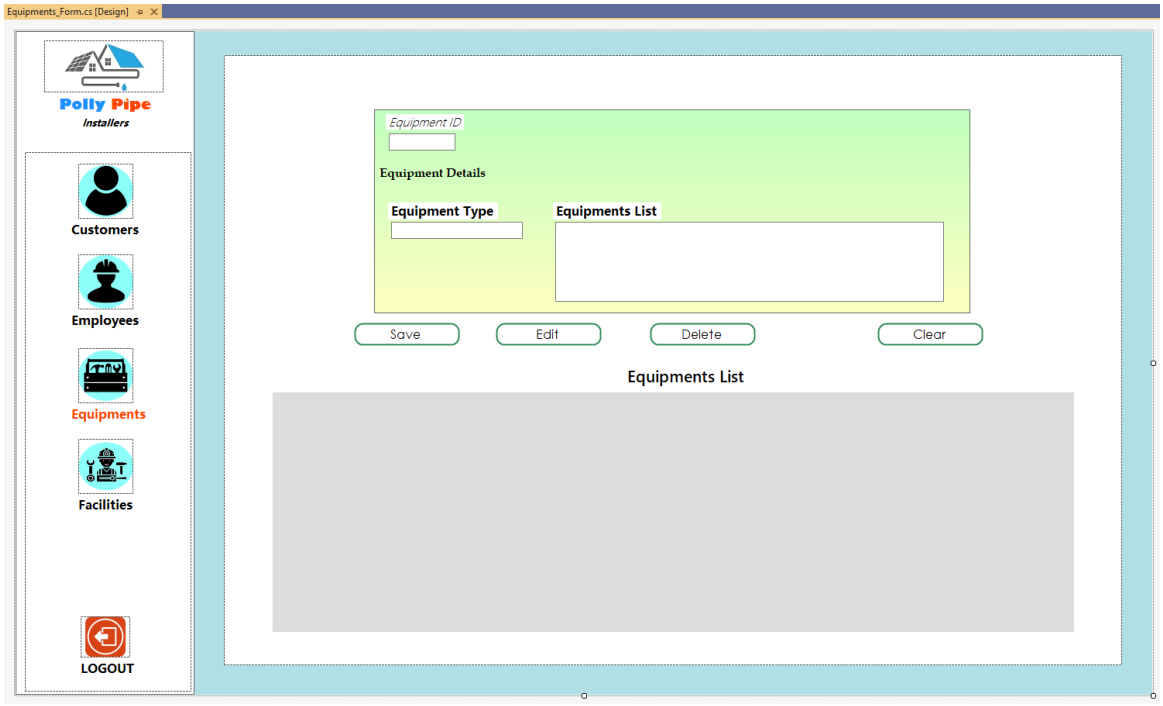


Figure 1. 50 Equipment from design (Can view by admin)

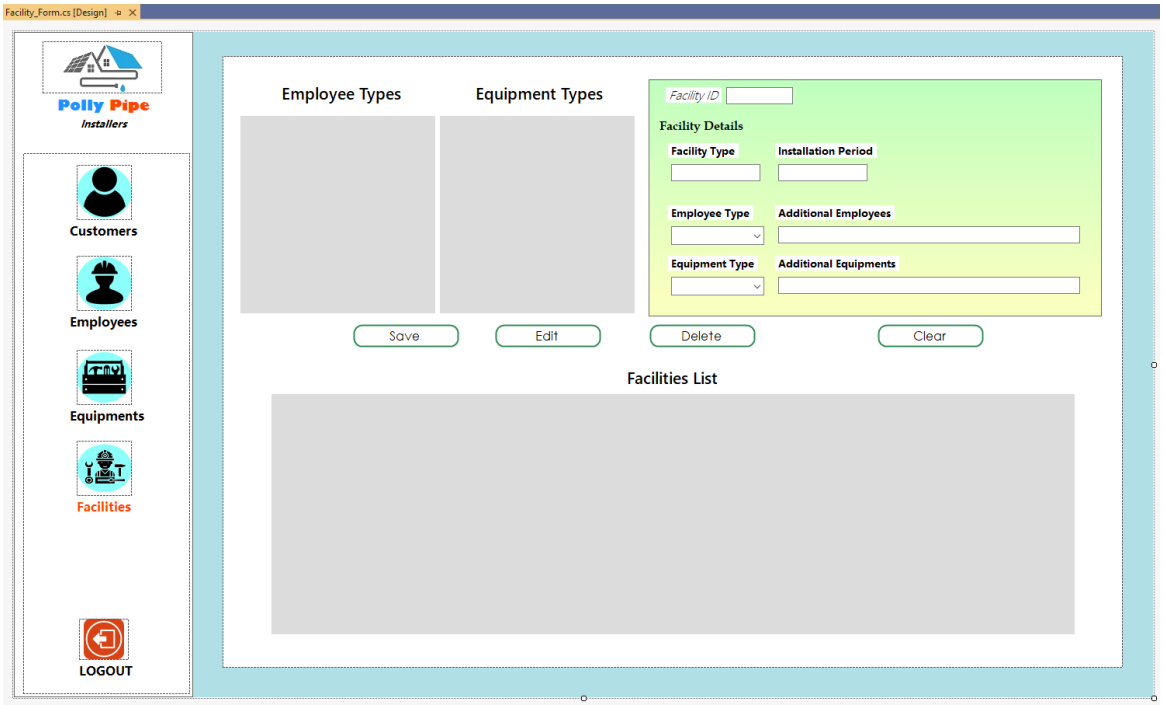


Figure 1. 51 Facilities from design (Can view by admin)

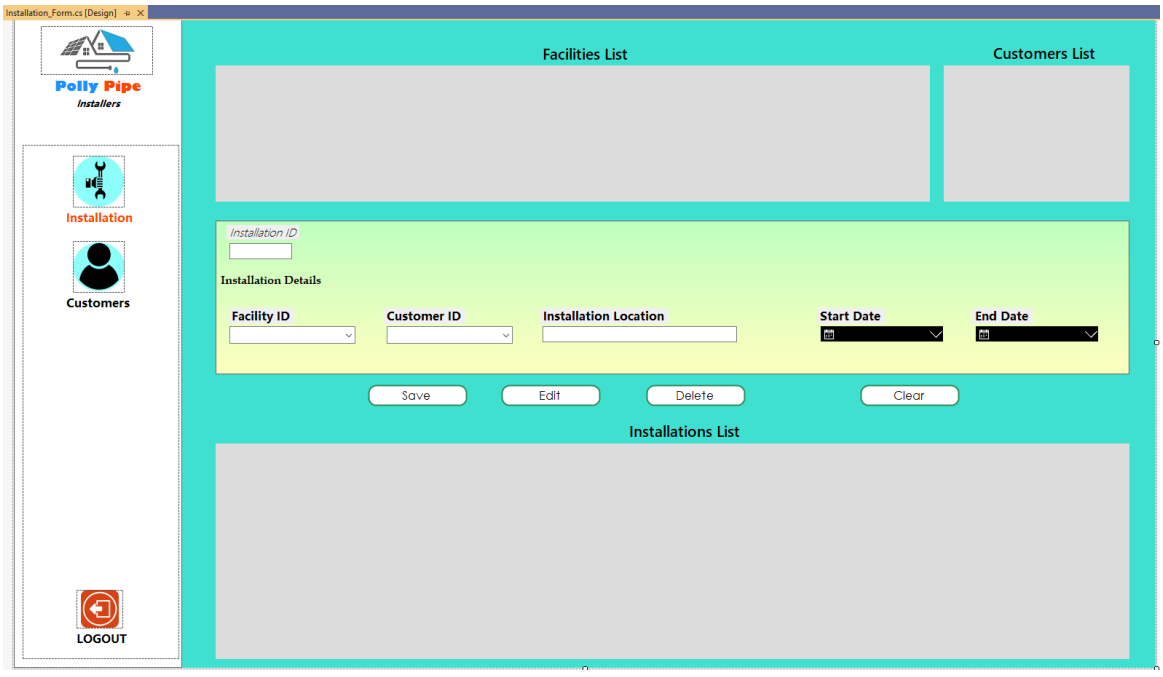


Figure 1. 52 Installation from design (Can view by sales representative)

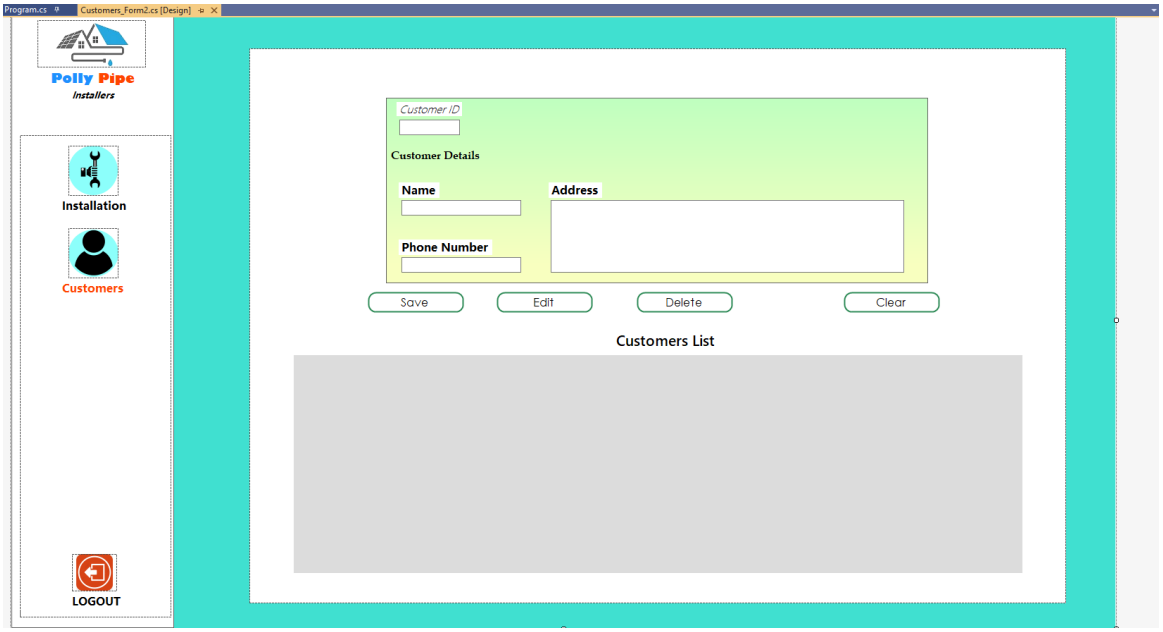


Figure 1. 53 Customer from 2 design (Can view by sales representative)

1.8.2 Evaluate the effectiveness of designed forms with identified user and system requirements before

User Requirements	
User Requirement key point	**Effectiveness of designed forms to fulfill the user requirement**
Creating logging system for both admin and sales representative The system should be followed by following two user groups. 1. Admin 2. Sales Representative	In the Login Form I've added combo box to select and identify whether the user is admin or sales representative. Depending on the position, the forms which they can enter are different, Admin can log into system and can make changes of necessary entities such as customers, employees, equipment, and facilities. But sales representatives have only control over of customers and installations.
Admin and Sales Representative could add new customer to the system	Admin can log into Customer Form and can add or make changes of customers. Even sales representatives do the same because sales representatives are also dealing with customers. If there is a new customer, sales representative has access to the Customers Form also to add the new customer. And also, Admin and sales representative doesn't have to manually enter the Customer ID since the system automatically assign ID value for them.

	Finally, with the bottom DataGrid view, the user can view the data. If he/she want to make changes, by double clicking a cell the data can retrieve into text boxes easily. This makes to update records effectively.
Admin could Add new employees with employee type to the system	Only admin can log into Employees Form and can add or make changes of employees and employee types of details. Because the employees are an important asset of a company which should handle over by superior rather than sales representative. Admin can log into Employee Types Forms only through Employees Form because it's enhanced the effectiveness of the system rather than having multiple buttons on the selection menu panel. This making this system so simple to use. And also, in the Employee Form there is a small DataGrid view to see employee types of details. Which it is showing the data inserted through Employee Types Form. When filling employee details in Employees Form, that DataGrid view can be used effectively to retrieve employee type details into text boxes to save more time. I've included a combo box for employee type id in Employees Form. My intention was to fill the combo box with employee types of id in the combo box which retrieved by the Employee Types Form.

	And also, Admin doesn't have to manually enter the Employee ID since the system automatically assign ID value for them. Finally, with the bottom DataGrid view, the user can view the data. If he/she want to make changes, by double clicking a cell the data can retrieve into text boxes easily. This makes to update records effectively.
Admin could add new equipment to the system	Only admin can log into Equipment Form and can add or make changes of equipment details. Because the equipment/s are an important asset of a company which should handle over by superior rather than sales representative. And also, Admin doesn't have to manually enter the Equipment ID since the system automatically assign ID value for them. Finally, with the bottom DataGrid view, the user can view the data. If he/she want to make changes, by double clicking a cell the data can retrieve into text boxes easily. This makes to update records effectively.
Adding new facilities to the system	Only admin can log into Facilities Form and can add or make changes of facility details. Because the facility details are an important asset of a company which should handle over by superior rather than sales representative.

	And also, in the Equipment Form there are 2 small DataGrid views to see employee types and equipment types of details. Which they are showing the data inserted through Employee Types Form and Equipment Form. When filling employee details and equipment details in Facilities Form, these 2 DataGrid views can be used effectively to retrieve employee details and equipment details into text boxes to save more time. I've included a combo box for employee type id and equipment type id in Facilities Form. My intention was to fill the combo boxes with ID details in the combo boxes from relevant forms of them. And also, Admin doesn't have to manually enter the Facility ID since the system automatically assign ID value for them. Finally, with the bottom DataGrid view, the user can view the data. If he/she want to make changes, by double clicking a cell the data can retrieve into text boxes easily. This makes to update records effectively
Customer could request installation according to the facilities Polly Pipe System has.	Sales Representative can log into Customer Form and can add or make changes of installation details.

Staff could assign employees to each installation. Staff could set period of each installation.	To make this system effective the customer requirements and handled by the sales representative. According having each facility, the sales representative can customize the employee types and equipment/s according to customer requirement. And also, despite each facility has period of days for installation, the sales rep can customize the days for the requirement of the customer which makes this system environment customer friendly.
User should have ability to move through forms easily.	Rather than having separate form for Menu For each Entity, I've created a selection menu panel at the left side of every form. User can move easily through the forms using that panel. This increases the effectiveness of the system. To improve the design, I've highlighted the title of button according to the relevant form. For example, in the Customers Form the customers button label is highlighted in red to indicate this is the customers Form. And also, I've designed cursor details also. When user moving around button the arrow cursor is change into hand sign to indicate this is a button. This helps a novice user to work with the system more effectively.

Table 1. 1 User requirement and Effectiveness of designed forms to fulfill the user requirement

System Requirements		
System Requirement	**Solution**	**Effectiveness of solution to fulfill the system requirement**
Create ERD.	Draw.io application to create ERD.	Since Draw.io is a free diagramming application it can be used to create flowcharts, network diagram software, ER diagrams, and can use to create database schema. By using this platform, I was able to create ERD with multiple shapes easily. And connecting and disconnecting each attribute, entity and relation with lines and arrows was easy. IF I want to move part of the design, only I had to do was selecting the portion of the design which I want to move. This helped me to save time rather than recreating the design from the start.
Create logical database.	LucidChart application to create logical database.	For this I've used "Lucidchart" webapplication. Lucidchart allows users to create and share professional flowchart diagrams for a variety of purposes, including brainstorming and project management.

		And with this application it was very easy to create logical database rather than using “draw.io” application. Since this is free and cloud-based application, I was able to save work anytime anywhere. IF I want to move part of the design, only I had to do was selecting the portion of the design which I want to move like I did in Draw.io application. This helped me to save time rather than recreating the design from the start. And this web application helped me to save time connecting and reconnecting each logical tables easily to show the relations. And displaying Primary Keys and Foreign Keys and interconnecting them was so efficient with this app.
Create desktop system application.	Visual Studio to create desktop system application.	To create a desktop application for the system I’ve used Visual Studio. We can create data-centric, modern line of business applications for Windows using.NET and Visual Studio. Since Windows Forms creation with Visual Studio is provide a UI(User Interface) framework I was able to create sophisticated form designs.

		And also, this platform provides easy drag-and-drop tools. Because of that it is easy to handle and create s desktop system application for even novice designers. And it was very easy to connect SQL with the platform. This platform has ain built SQL Server which allow us to create database in the visual studio with tables. It helped me a lot when I needed to check table designed to operational system before creating the database in Microsoft SQL Server management system.
Handle database.	Microsoft SQL Server Management Studio to handle database.	Then to create database for the system I selected Microsoft SQL Server Management Studio since it can be used to quickly create or modify SQL database. And also, it can use to add database objects including tables, views, and stored procedures. And it was very easy to connect Visual studio with the platform.

Table 1. 2 System requirement and Effectiveness of solution to fulfill the system requirement

Activity 2

2.1 SQL

2.1.1 What is SQL

SQL stands for Structured Query Language, and it is a computer language used to store, manipulate, and retrieve data from a relational database. The Relational Database System (RDBMS) standard language is SQL. SQL is the standard database language used by all Relational Database Management Systems (RDMS) such as MySQL, MS Access, Oracle, Sybase, Informix, Postgres, and SQL Server. SQL is commonly used because it has the following benefits.

- Allows users to use relational database management systems to retrieve data.
- Users can describe the data using this feature.
- Allows users to define and manipulate the data in a database.
- Allows SQL modules, libraries, and pre-compilers to be embedded within other languages.
- Users have the ability to build and delete databases and tables.

There are 4 types of SQL commands as following.

1. DDL (Data Definition Language)
2. DML (Data Manipulation Language)
3. DCL (Data Control Language)
4. TCL (Transaction Control Language)

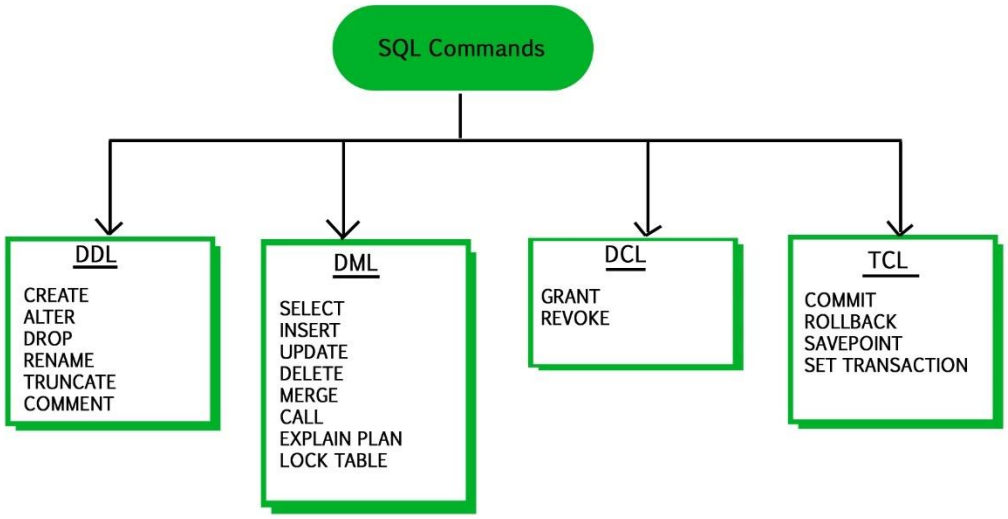


Figure 2. 1 4 types of SQL commands with key examples

2.2 DDL (Data Definition Language)

Data description language may sound like another programming language, but it's essentially a classification system for SQL commands. The set of SQL commands known as data definition language (DDL) can be used to design and alter database structures. DDL statements are used to create, alter, and delete objects such as indexes, triggers, tables, and views. The following are examples of DDL statements.

SQL Statement	Description
CREATE	This is used to create new table/database.
ALTER	This is used to modify the structure of table/database.
DROP	This is used to delete table/database.
TRUNCATE	This is used to remove all table records and to free the space contained the table.

Table 2. 1 Examples of DDL statements

2.2.1 CREATE command in DDL

The CREATE statement is used to add a new table to an already existing database. And also, it can be used to add a new database also. Other database objects, such as stored procedures and functions, are created using the CREATE statement.

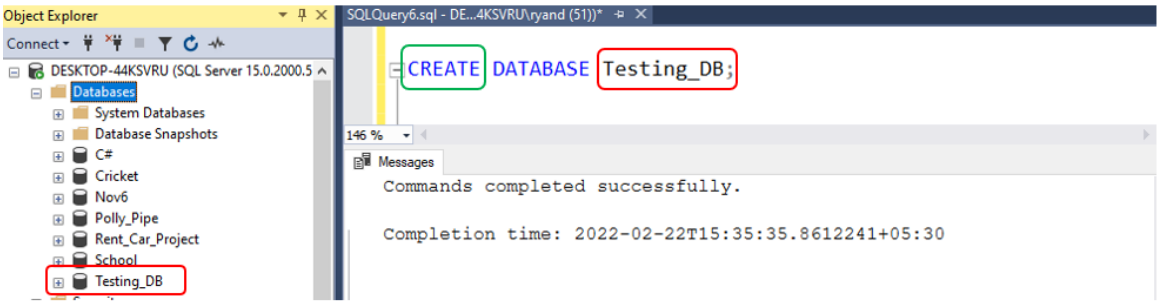


Figure 2. 2 Create Database in Microsoft SQL Server Management Studio using DDL commands

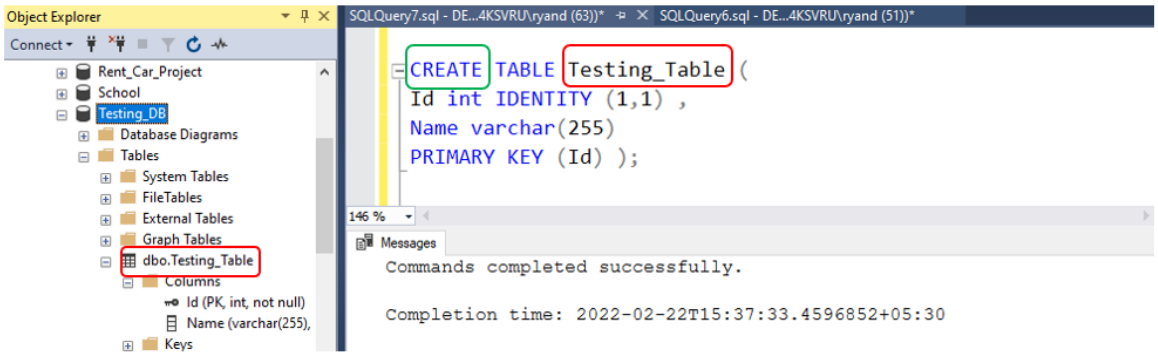


Figure 2. 3 Create Table in Microsoft SQL Server Management Studio using DDL commands

2.2.2 ALTER command in DDL

There are situations when it is necessary to modify an existing table in a SQL Server database, such as to add, remove, or change the size or type of data type of a table column. The ALTER TABLE statement allows you to make changes to an existing table, such as adding, modifying, or removing columns.

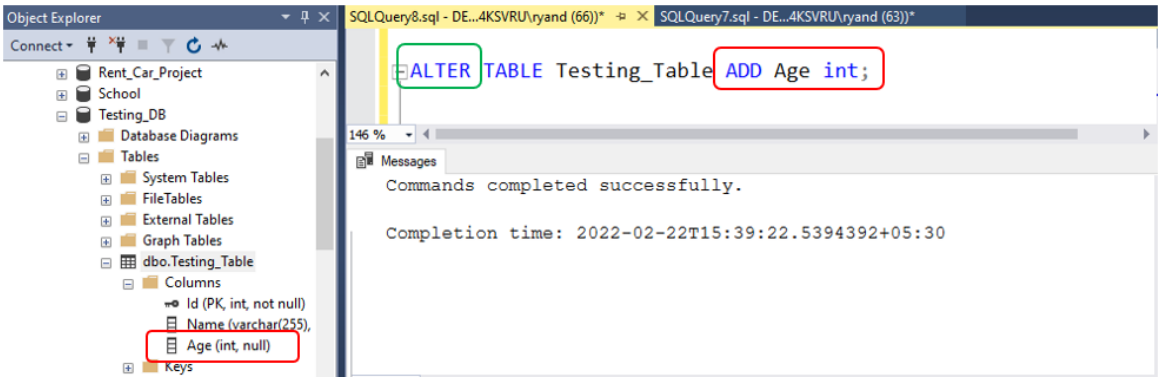


Figure 2. 4 Alter Table columns in Microsoft SQL Server Management Studio using DDL commands

2.2.3 TRUNCATE command in DDL

DELETE, TRUNCATE, and DROP In SQL Server, SQL queries are frequently used to delete data from a database. However, the TRUNCATE SQL command deletes all rows from a table without logging the individual row deletions. TRUNCATE is a faster query than DELETE.

TRUNCATE is performed using a table lock, which locks the entire table in order to erase all records. With TRUNCATE, we can't use the WHERE clause. And also, if the table contains an identity column, the column is reset to its initial value.

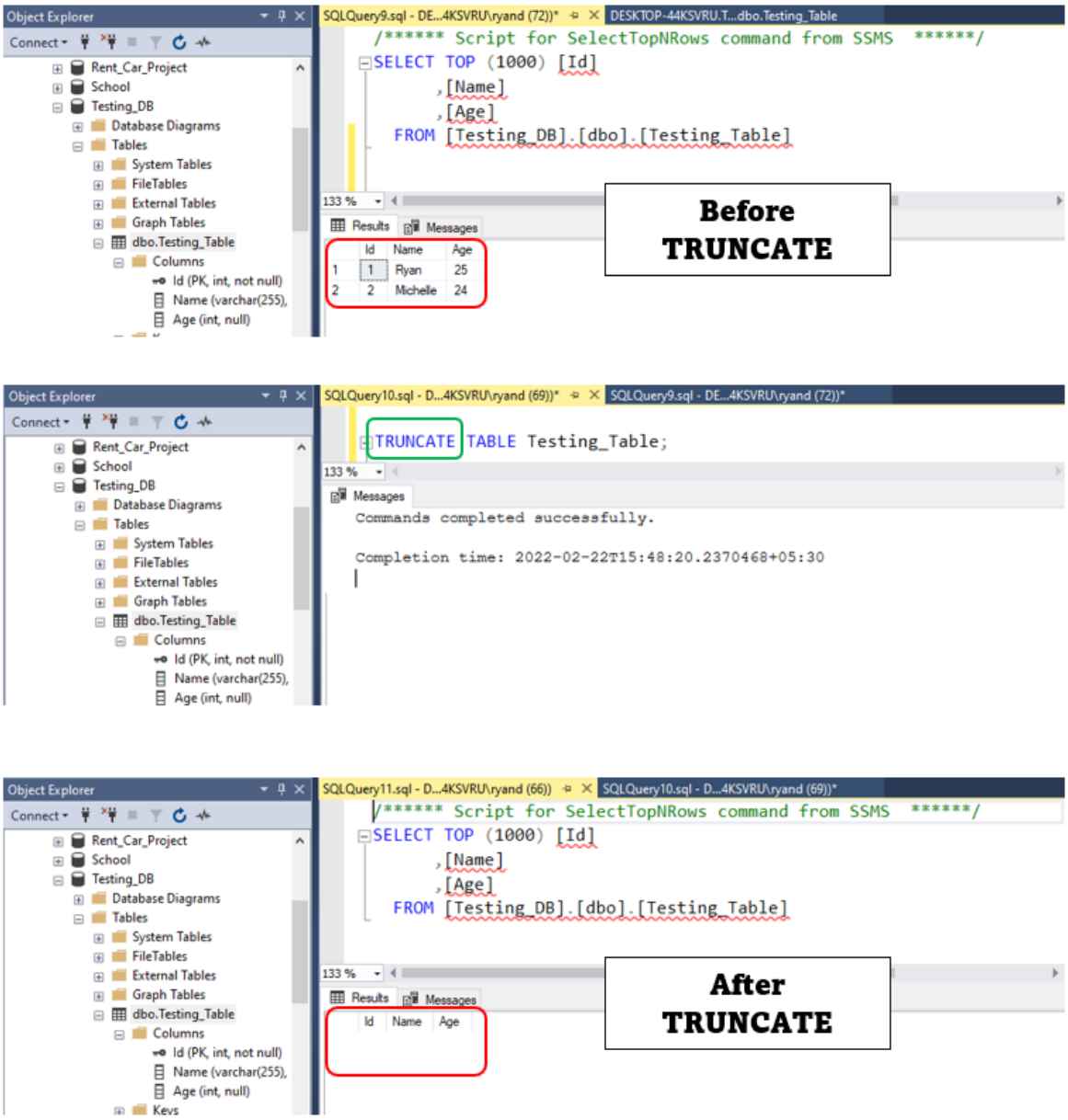


Figure 2. 5 Truncate Table in Microsoft SQL Server Management Studio using DDL commands

2.2.4 DROP command in DDL

The DROP table query deletes one or more table definitions as well as all associated data, indexes, triggers, constraints, and permissions. A table can be dropped from the database using the DROP command. All of the tables' rows, indexes, and privileges will be erased as well.

Since no DML triggers will be fired, the operation cannot be reversed. DDL instructions such as DROP and TRUNCATE are used, whereas DML commands such as DELETE are used. Rolling back (undoing) DELETE actions is possible, however DROP and TRUNCATE operations are not.

Figure 2. 6 Drop Table in Microsoft SQL Server Management Studio using DDL commands

2.3 DML (Data Manipulation Language)

The Data Manipulation Language (DML) is a group of computer language that provide commands for manipulating data in databases. DML is mostly used in SQL databases. Inserting data into database tables, fetching existing data, removing data from existing tables, and modifying existing data are all part of this manipulation.

SQL Statement	Description
SELECT	This is used to create new table/database.
INSERT	This is used to modify the structure of table/database.
UPDATE	This is used to delete table/database.
DELETE	This is used to remove all table records and to free the space contained the table.

Table 2. 2 DML Statements and description

2.3.1 INSERT command in DML

The INSERT command is used to enter data to a table. We can add one or many records to any particular table in a database using this command. It's also used to add records to a code that already exists.

For this, I've created a new table called team to demonstrate DML commands as below.

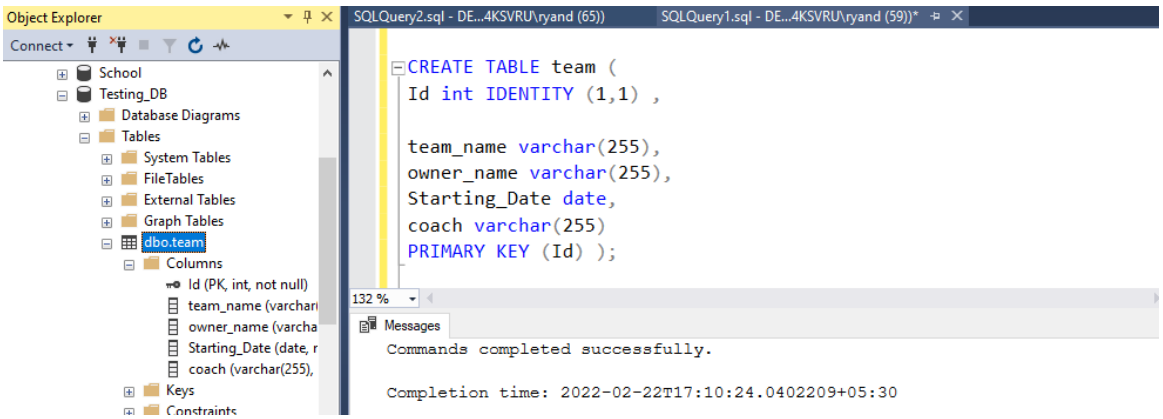


Figure 2. 7 New table "team" to demonstrate DML commands

We can insert records to table using these both ways as below figure. We can insert records 1 by 1 or we can insert all records at once.

```
INSERT INTO team (team_name,owner_name,Starting_Date,coach)
VALUES ('Sri Lanka', 'Presidet','1980-01-01','Sanath');

INSERT INTO team (team_name,owner_name,Starting_Date,coach)
VALUES ('England', 'Prime Minister','1980-01-01','George');

INSERT INTO team (team_name,owner_name,Starting_Date,coach)
VALUES ('India', 'Prime Minister','2000-01-01','Modi');
```

OR

```
INSERT INTO team (team_name,owner_name,Starting_Date,coach)
VALUES  ('Sri Lanka', 'Presidet','1980-01-01','Sanath'),
        ('England', 'Prime Minister','1980-01-01','George'),
        ('India', 'Prime Minister','2000-01-01','Modi');
```

Figure 2. 8 Records inserting 2 ways

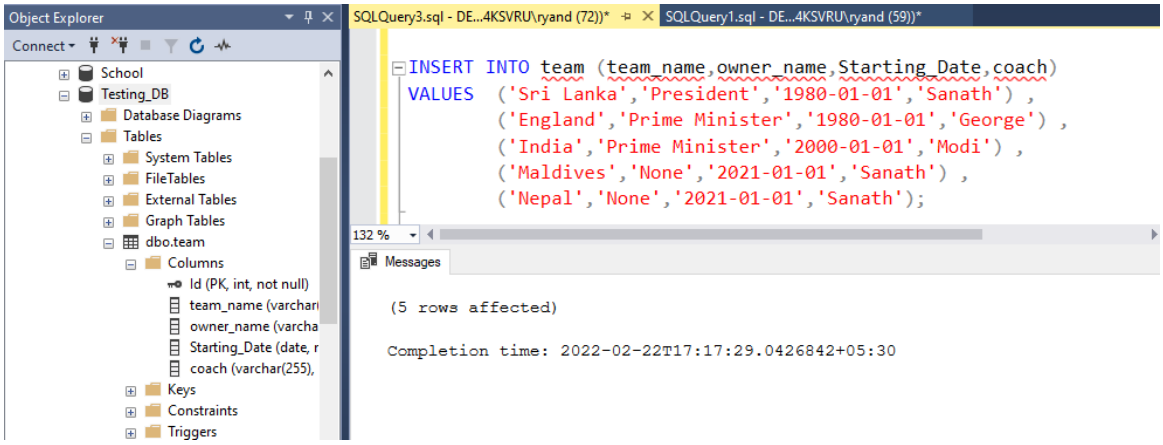


Figure 2. 9 Inserted Records to "team" table using DML commands

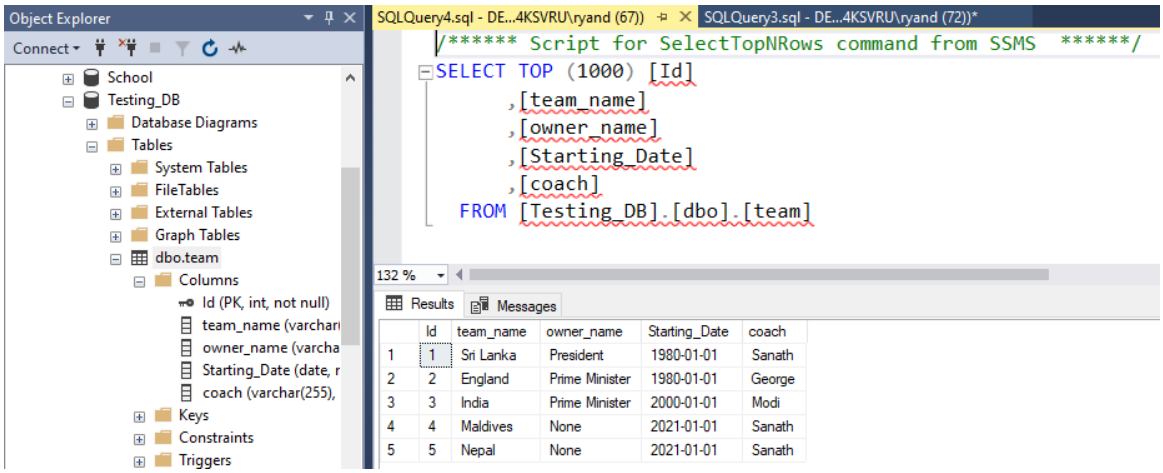


Figure 2. 10 Inserted Records to "team" table using DML commands

2.3.2 DELETE command in DML

The SQL DELETE command removes all records from a database table. Delete permissions on the target table are needed to execute a DELETE query. Select permissions are necessary if we need to use a WHERE clause in a DELETE.

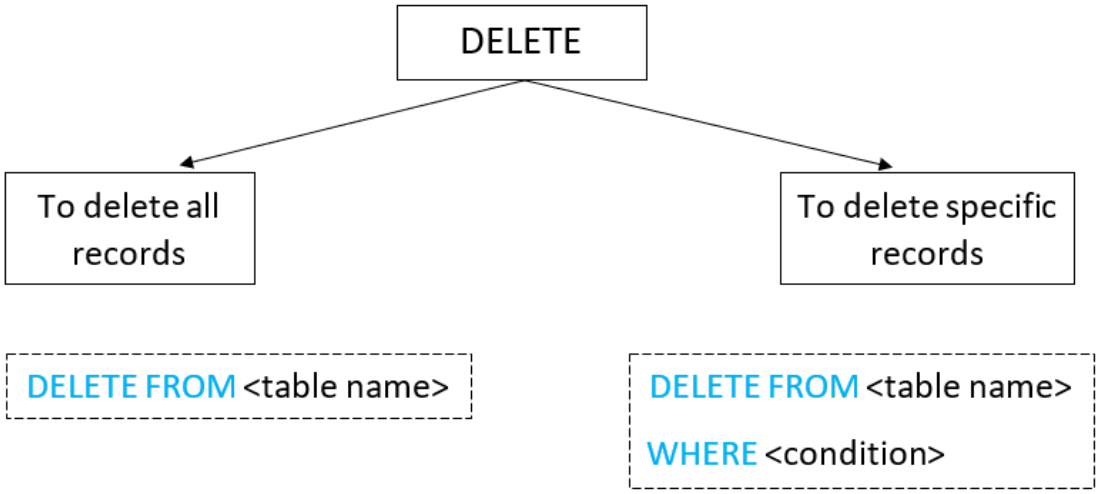


Figure 2. 11 DML delete statement classification with syntaxes

To delete all Records

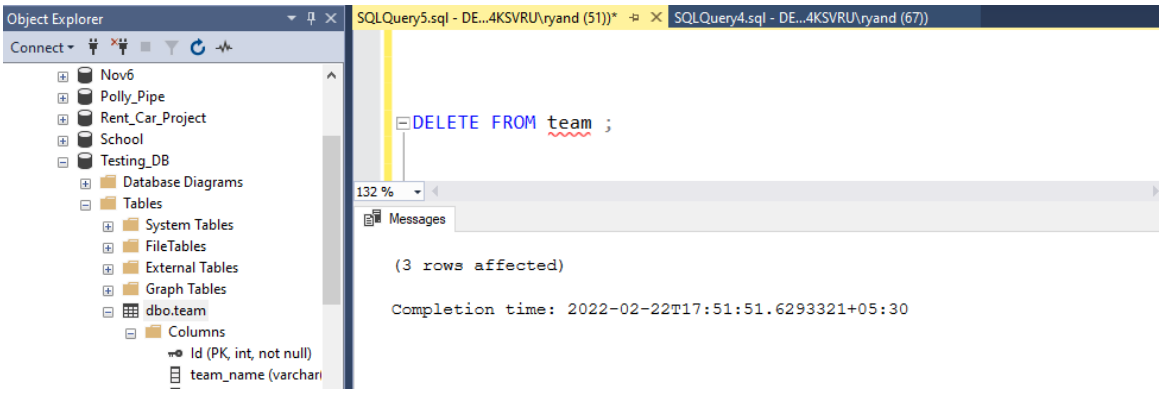


Figure 2. 12 Delete all records

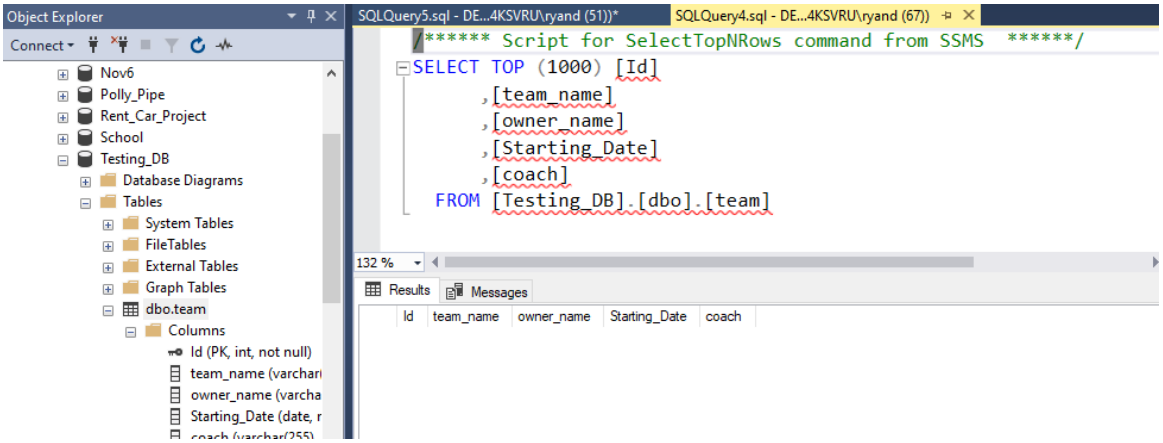


Figure 2. 13 Delete all records - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Presidet	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	6	Maldives	None	2021-01-01	Salmon
5	7	Nepal	None	2021-01-01	None

Figure 2. 14 Delete all records :– Which data was deleted

To delete specific Records – Delete 1 record

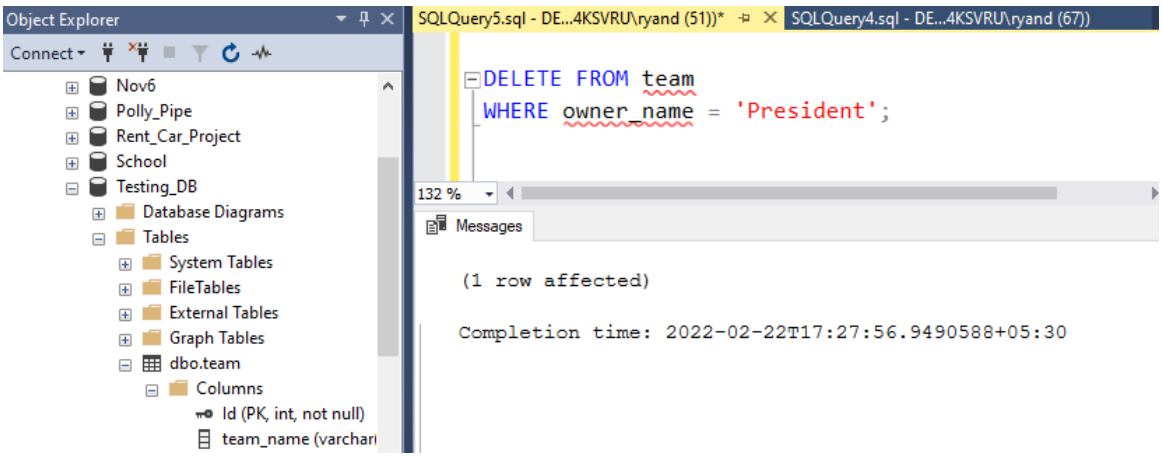


Figure 2. 15 Delete specific Records – Delete 1 record

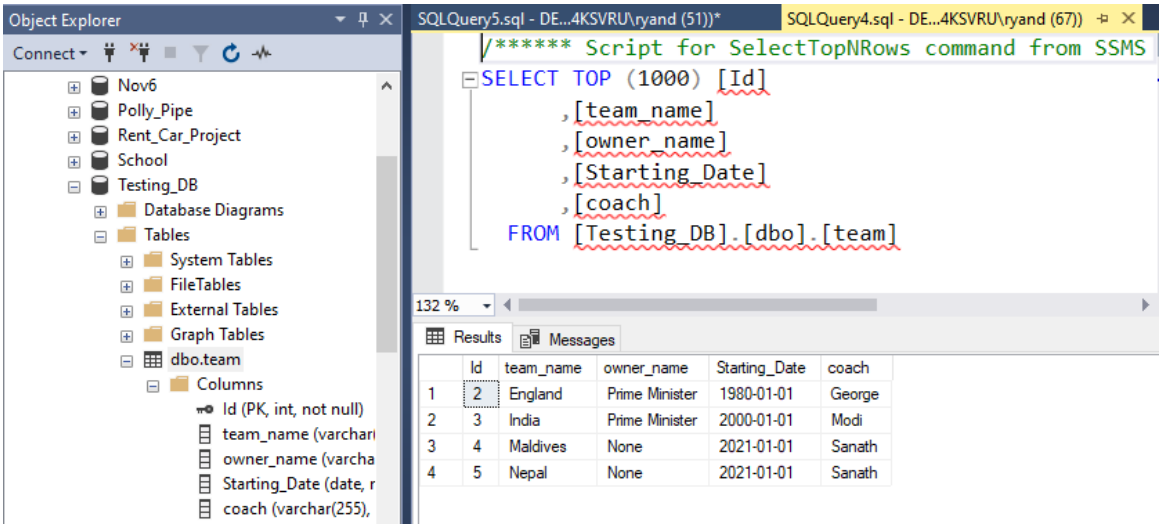


Figure 2. 16 Delete specific Records – Delete 1 record - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	~~Sri Lanka~~	~~President~~	~~1980-01-01~~	~~Sanath~~
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	6	Maldives	None	2021-01-01	Salmon
5	7	Nepal	None	2021-01-01	None

Figure 2. 17 Delete specific Records – Delete 1 record :– Which data was deleted

To delete specific Records – Delete multiple records with AND condition

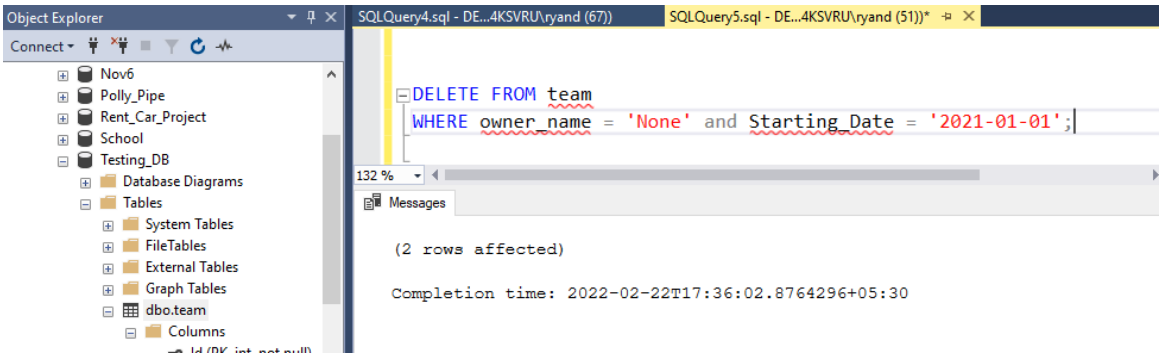


Figure 2. 18 Delete specific Records – Delete multiple records with AND condition

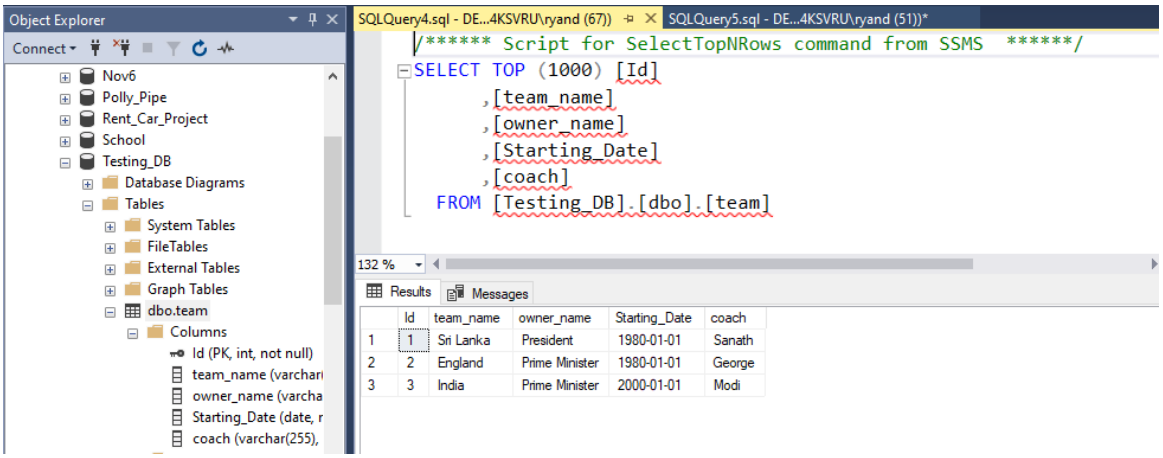


Figure 2. 19 Delete specific Records – Delete multiple records with AND condition - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Presidet	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	6	Maldives	None	2021-01-01	Salmon
5	7	Nepal	None	2021-01-01	None

Figure 2. 20 Delete specific Records – Delete multiple records with AND condition :– Which data was deleted

To delete specific Records – Delete multiple records with OR condition

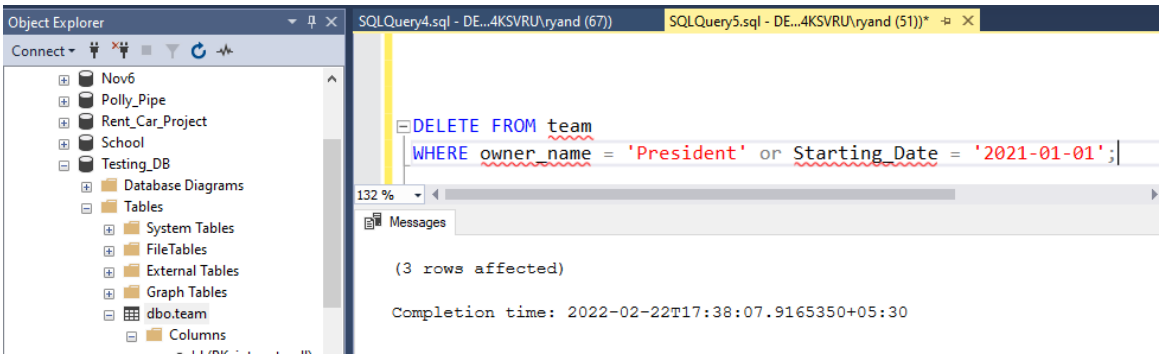


Figure 2. 21 Delete specific Records – Delete multiple records with OR condition

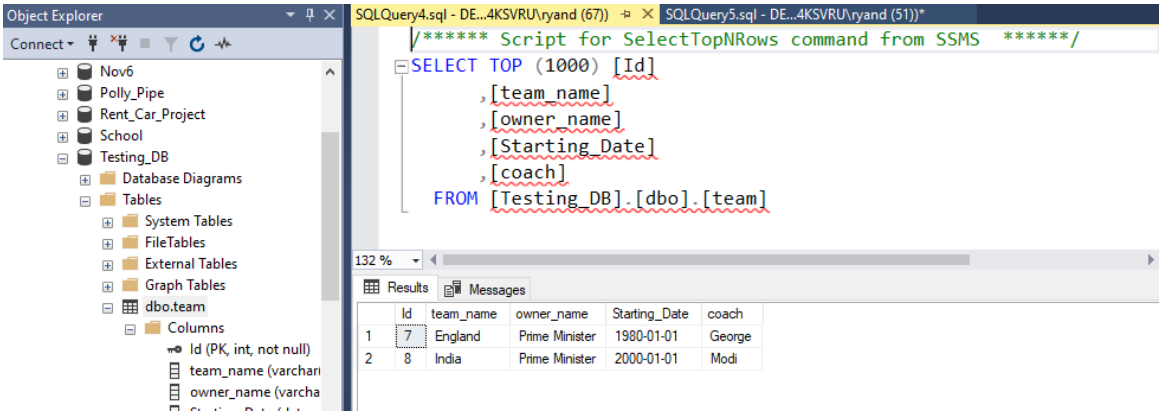


Figure 2. 22 Delete specific Records – Delete multiple records with OR condition - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Presidet	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	6	Maldives	None	2021-01-01	Salmon
5	7	Nepal	None	2021-01-01	None

Figure 2. 23 Delete specific Records – Delete multiple records with OR condition :– Which data was deleted

To delete specific Records - Delete all expect 1 record with NOT condition

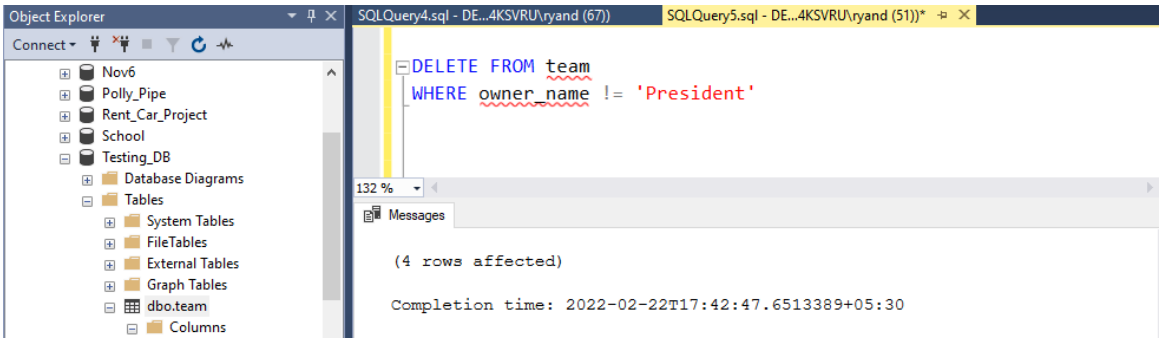


Figure 2. 24 Delete specific Records – Delete all expect 1 record with NOT condition

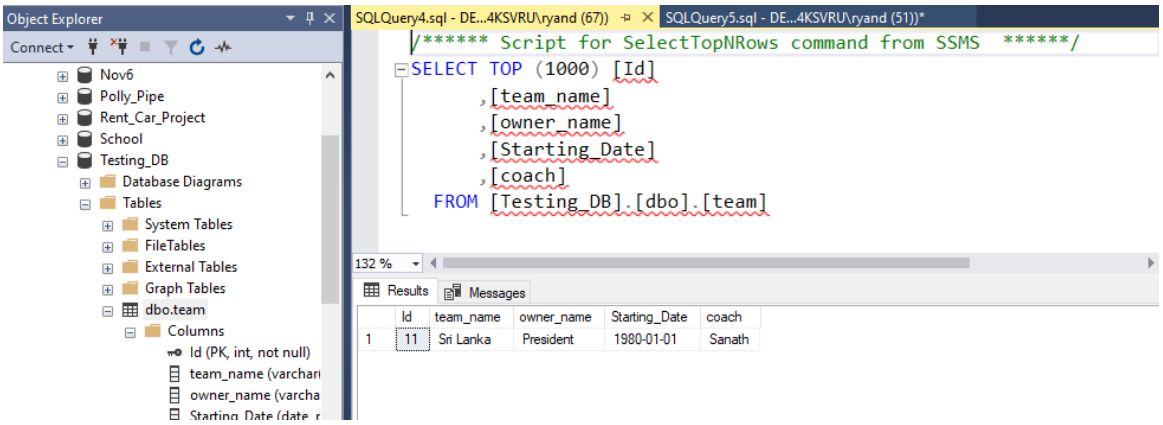


Figure 2. 25 Delete specific Records – Delete all expect 1 record with NOT condition - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Presidet	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	~~6~~	~~Maldives~~	~~None~~	~~2021-01-01~~	~~Salmon~~
5	~~7~~	~~Nepal~~	~~None~~	~~2021-01-01~~	~~None~~

Figure 2. 26 Delete specific Records – Delete all expect 1 record with NOT condition :– Which data was deleted

To delete specific Records - Delete records higher / lower than specific value

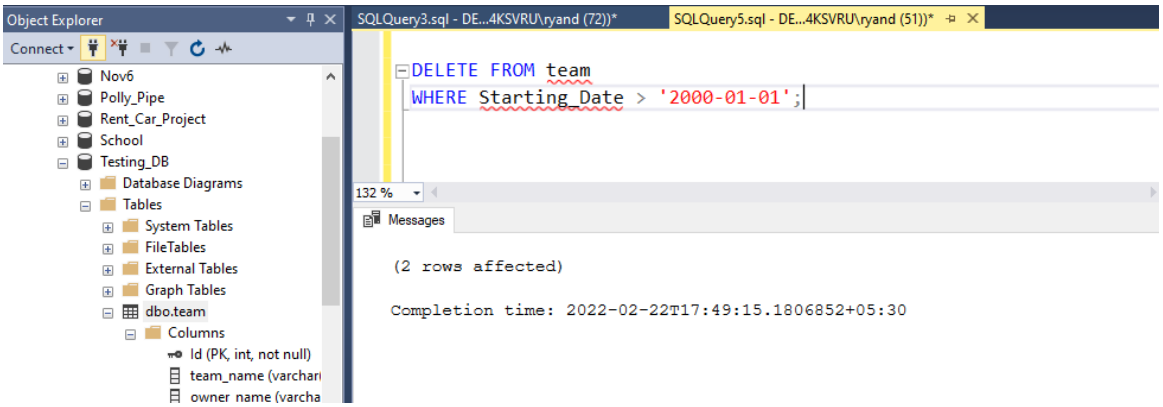


Figure 2. 27 Delete specific Records – Delete records higher / lower than specific value

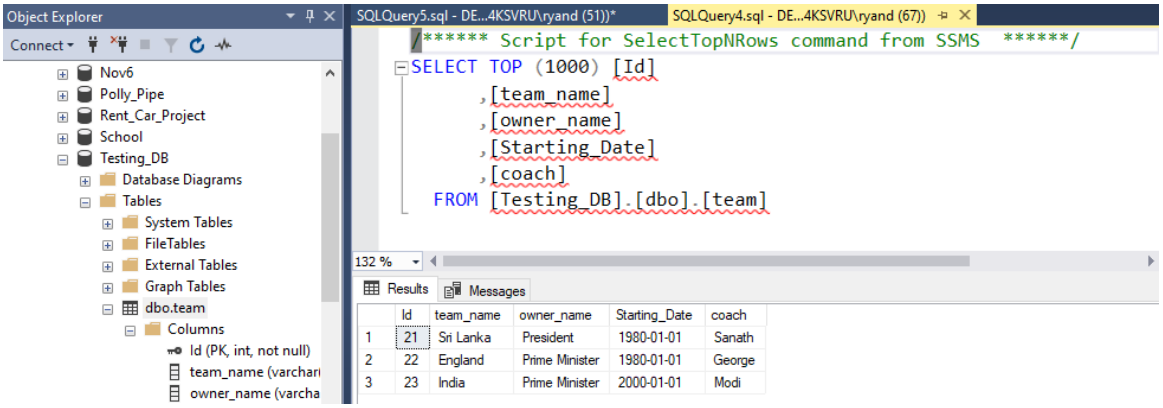


Figure 2. 28 Delete specific Records – Delete records higher / lower than specific value - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Presidet	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	~~6~~	~~Maldives~~	~~None~~	~~2021-01-01~~	~~Salmon~~
5	~~7~~	~~Nepal~~	~~None~~	~~2021-01-01~~	~~None~~

Figure 2. 29 Delete specific Records – Delete records higher / lower than specific value :– Which data was deleted

2.3.3 UPDATE command in DML

The SQL DELETE command removes all records from a database table. Delete permissions on the target table are needed to execute a DELETE query. Select permissions are necessary if we need to use a WHERE clause in a DELETE.

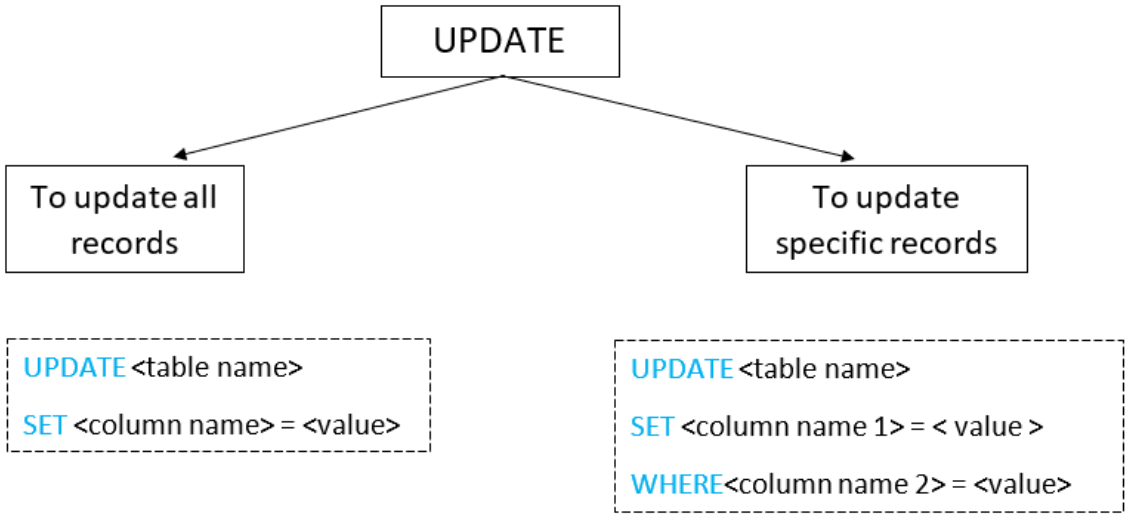


Figure 2. 30 DML delete statement classification with syntaxes

To update whole column Records

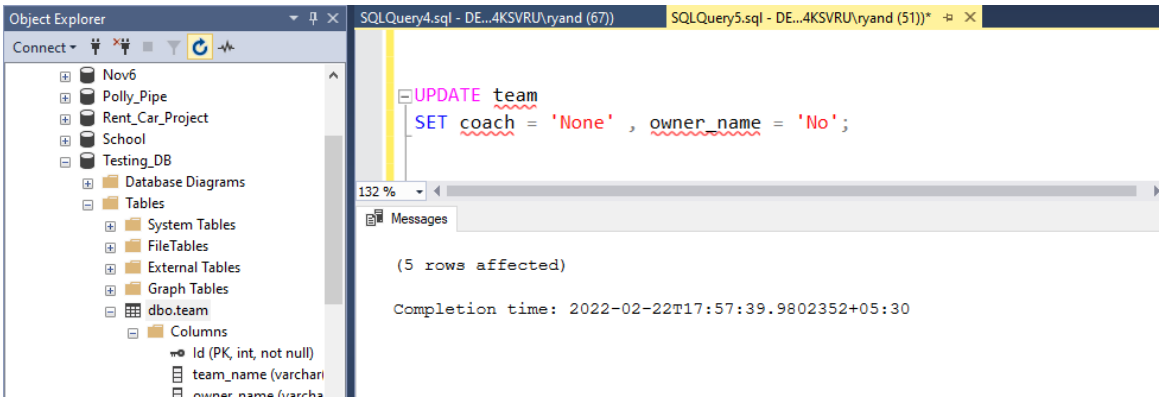


Figure 2. 31 DML delete statement classification with syntaxes

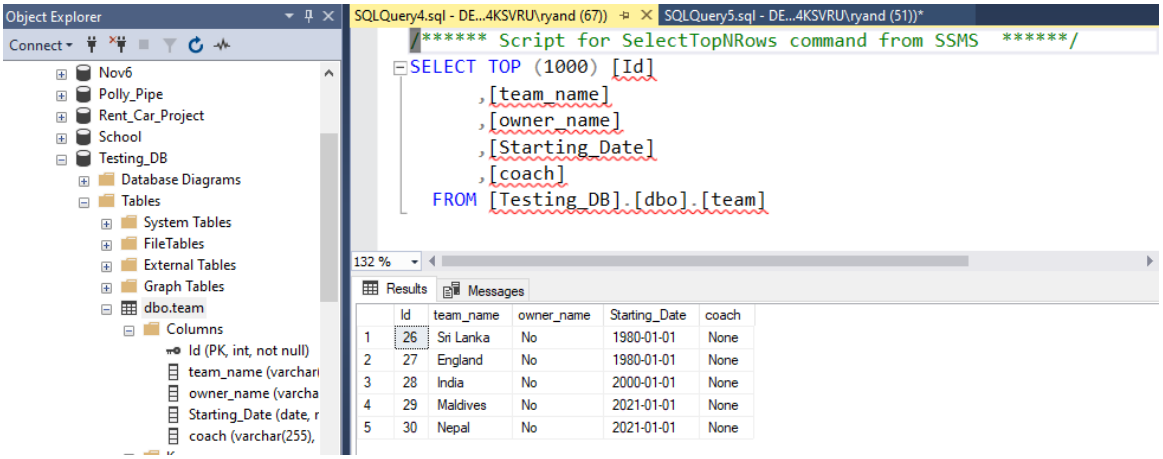


Figure 2. 32 Update whole column Records - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	NO	1980-01-01	None
2	4	England	NO	1980-01-01	None
3	5	India	NO	2000-01-01	None
4	6	Maldives	NO	2021-01-01	None
5	7	Nepal	NO	2021-01-01	None

Figure 2. 33 Update whole column Records :– Which data was updated

To update specific Records – Update 1 record

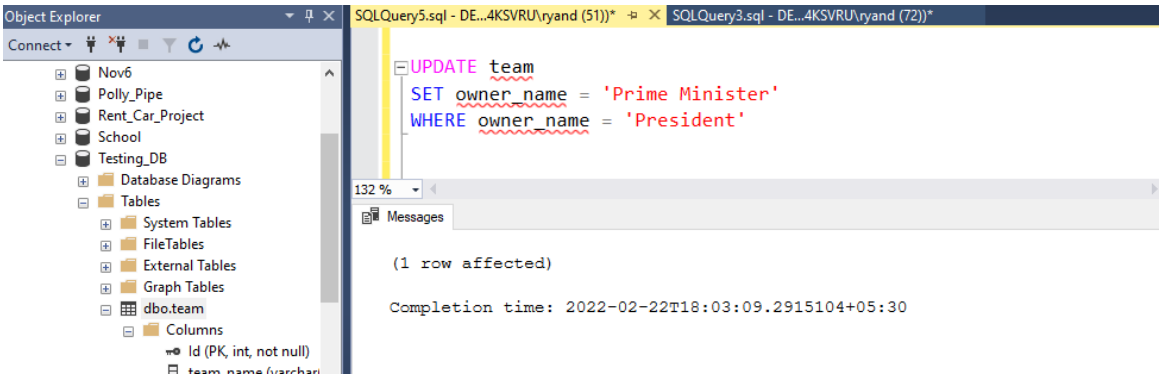


Figure 2. 34 Update specific Records – Update 1 record

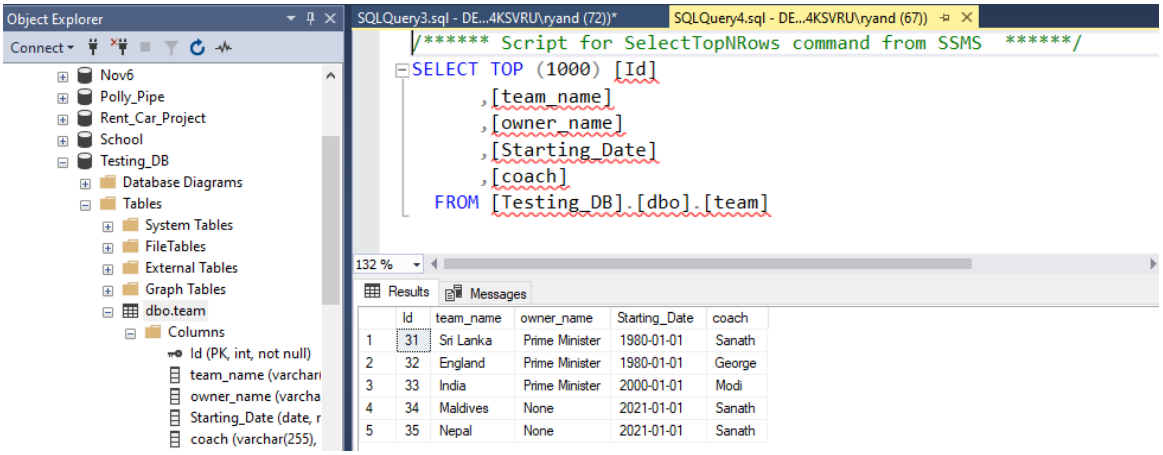


Figure 2. 35 Update specific Records – Update 1 record - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Prime Minister	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	George
3	5	India	Prime Minister	2000-01-01	Modi
4	6	Maldives	None	2021-01-01	Salmon
5	7	Nepal	None	2021-01-01	None

Figure 2. 36 Update specific Records – Update 1 record :– Which data was updated

To update specific Records – Update multiple records with OR condition

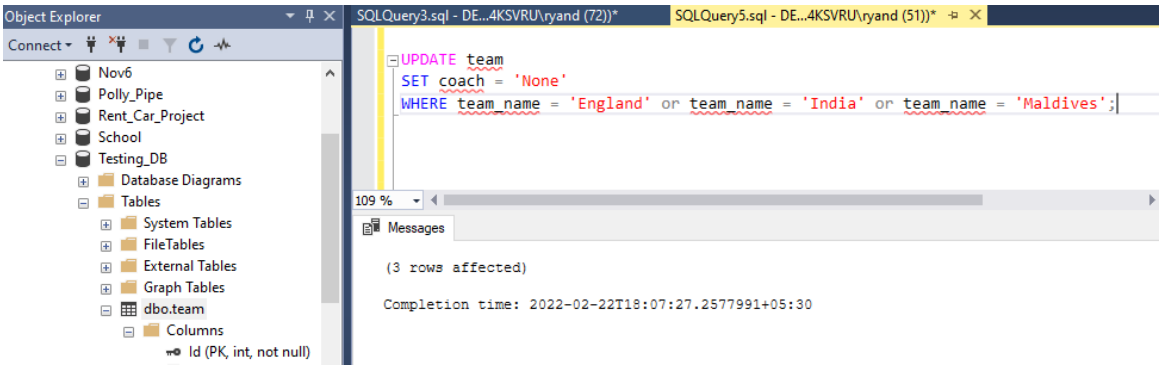


Figure 2. 37 Update specific Records – Update multiple records with OR condition

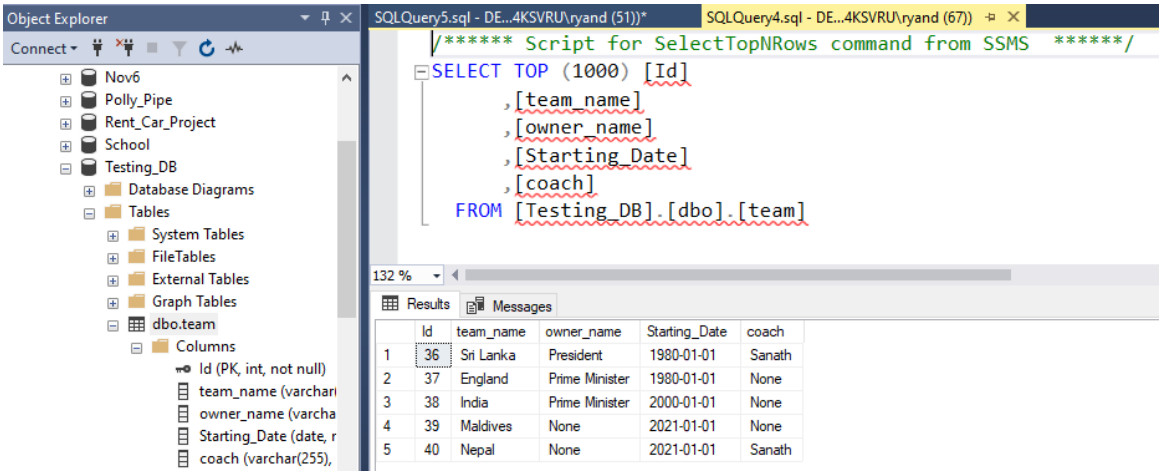


Figure 2. 38 Update specific Records – Update multiple records with OR condition - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Prime Minister	1980-01-01	Sanath
2	4	England	Prime Minister	1980-01-01	None
3	5	India	Prime Minister	2000-01-01	None
4	6	Maldives	None	2021-01-01	None
5	7	Nepal	None	2021-01-01	None

Figure 2. 39 Update specific Records – Update multiple records with OR condition :– Which data was updated

To update specific Records – Update multiple records with AND condition

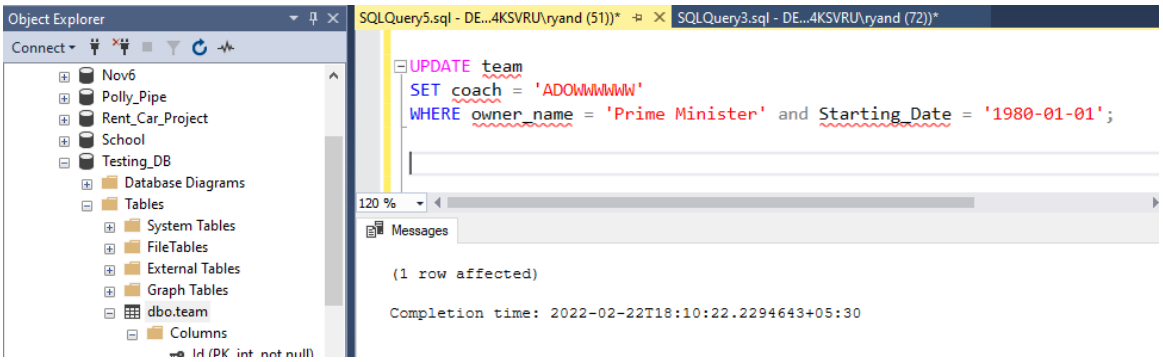


Figure 2. 40 Update specific Records – Update multiple records with AND condition

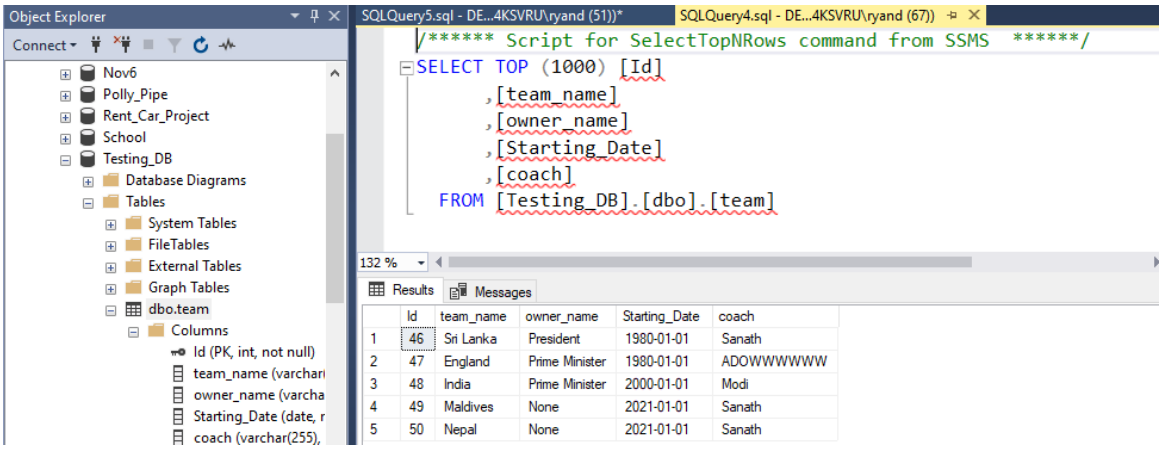


Figure 2. 41 Update specific Records – Update multiple records with AND condition - Output

	Id	team_name	owner_name	Starting_Date	coach
1	3	Sri Lanka	Prime Minister	1980-01-01	ADOWWWWWW
2	4	England	Prime Minister	1980-01-01	ADOWWWWWW
3	5	India	Prime Minister	2000-01-01	None
4	6	Maldives	None	2021-01-01	None
5	7	Nepal	None	2021-01-01	None

Figure 2. 42 Update specific Records – Update multiple records with AND condition :– Which data was updated

2.3.4 SELECT command in DML

The SQL DELETE command removes all records from a database table. Delete permissions on the target table are needed to execute a DELETE query. Select permissions are necessary if we need to use a WHERE clause in a DELETE.

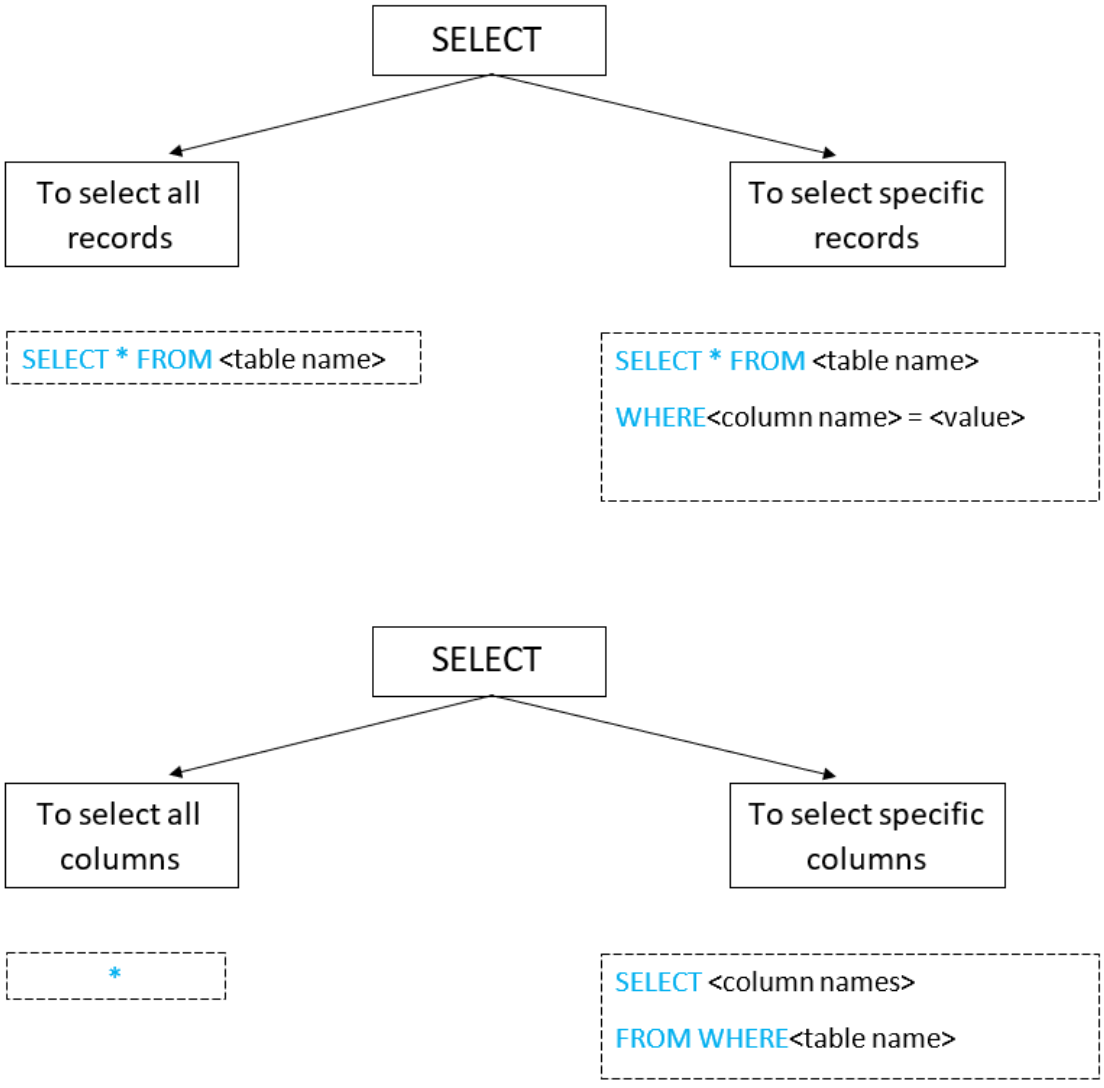


Figure 2. 43 DML select statement classification with syntaxes

To select all Records

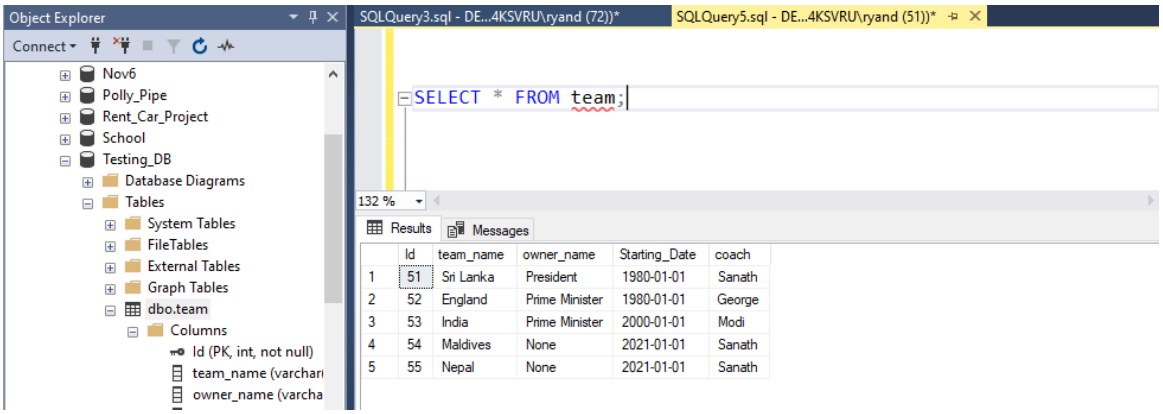


Figure 2. 44 Select all Records with the output

To select specific Records – Select 1 record

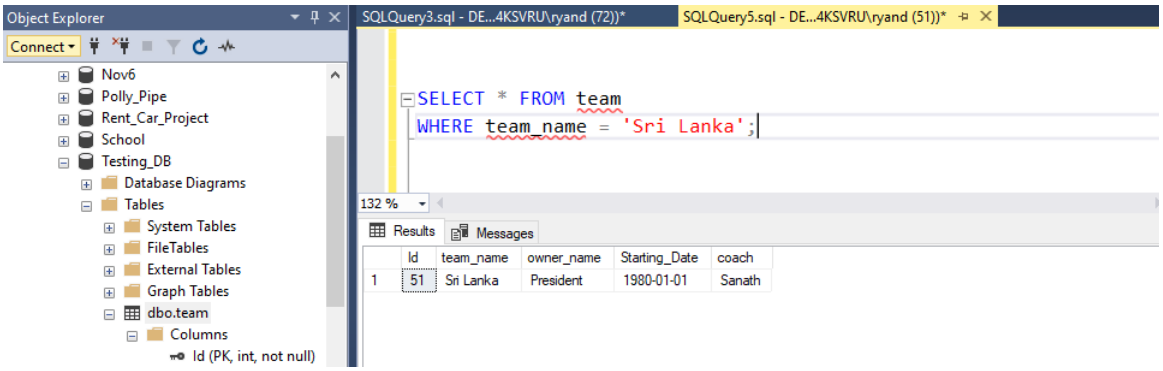


Figure 2. 45 Select specific Records – Select 1 record with the output

To select specific Records – Select multiple records with OR condition

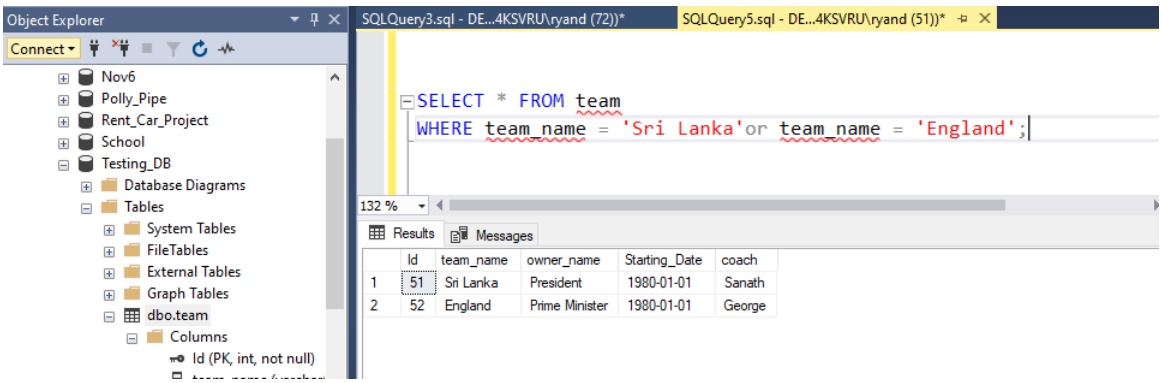


Figure 2. 46 Select specific Records – Select multiple records with OR condition with the output

To select specific Records – Select multiple records with AND condition

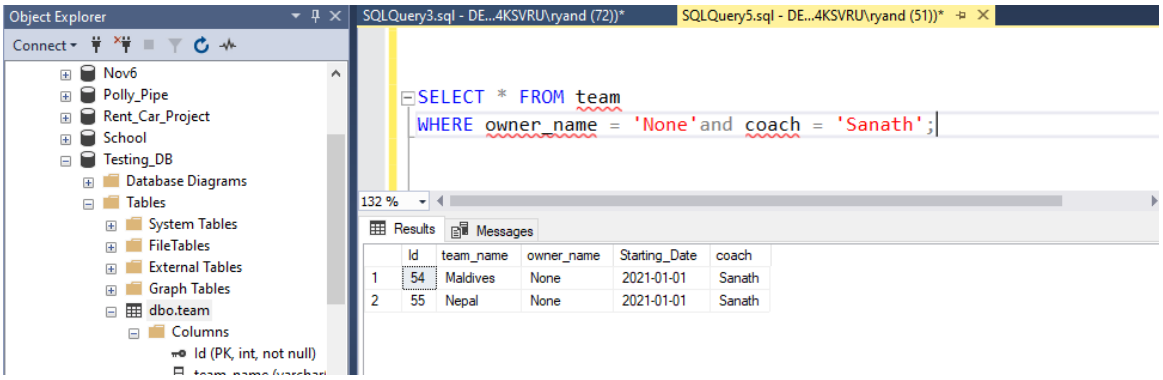


Figure 2. 47 Select specific Records – Select multiple records with AND condition with the output

To select specific Records – Select all expect 1 record with NOT condition

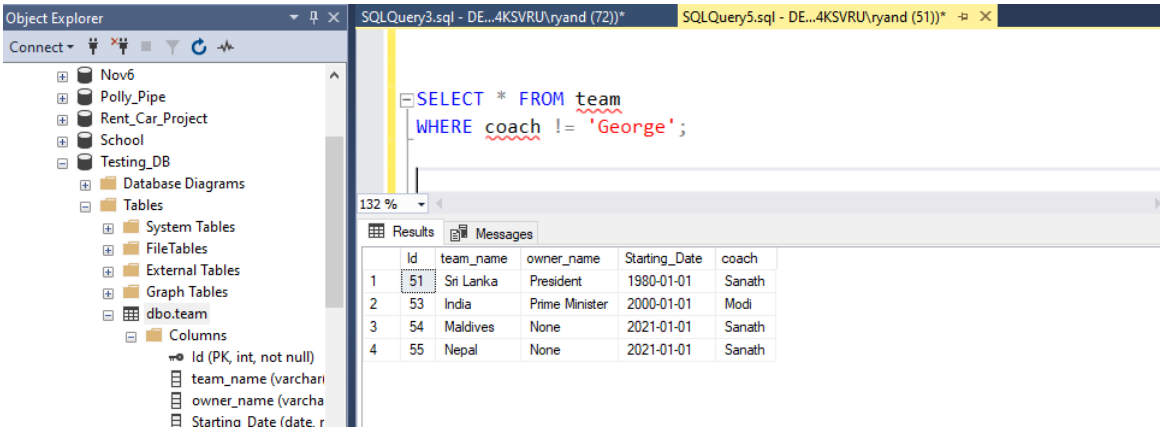


Figure 2. 48 Select specific Records – Select all expect 1 record with NOT condition and the output

To select specific Records – Select records higher / lower than specific value

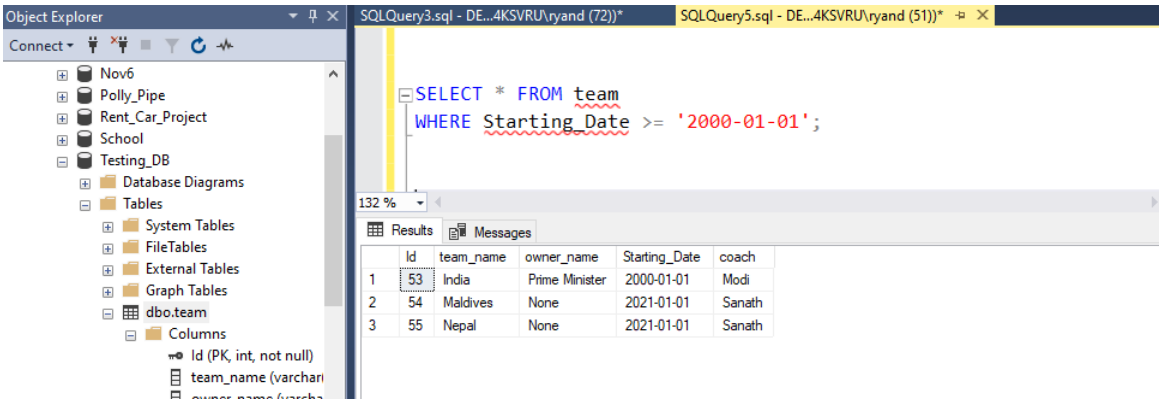


Figure 2. 49 Select specific Records – Select records higher / lower than specific value and the output

To select specific Columns

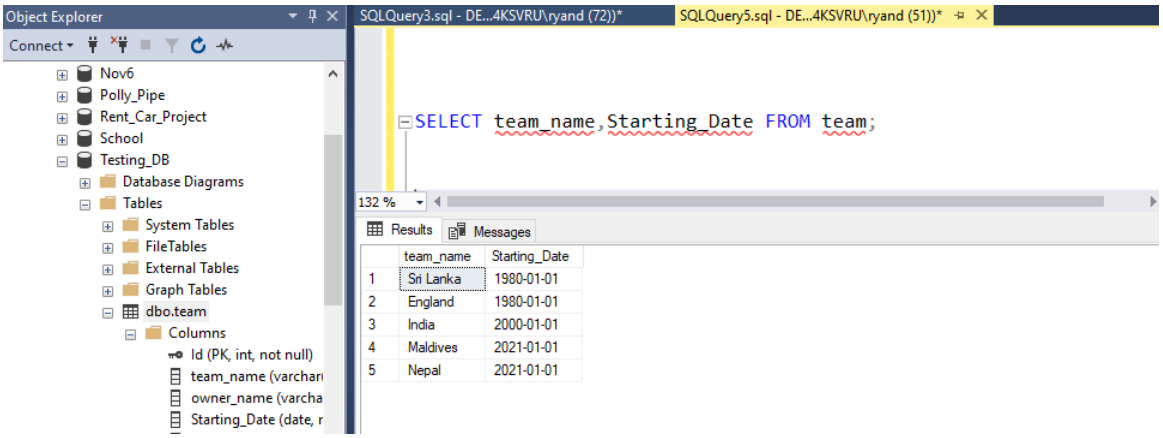


Figure 2. 50 Select specific Columns with the output

2.3.5 Different criteria and clauses used in DML

WHERE criteria with Select statement

We may want to concentrate on a certain part of the table, such as the Prime Ministers in the Owner Name column. We can apply the SELECT statement with the WHERE criteria in this case.

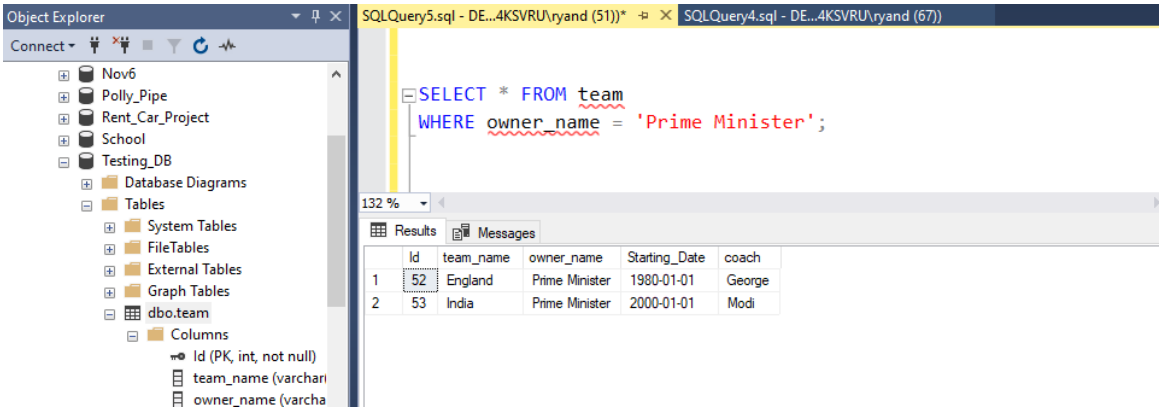


Figure 2. 51 WHERE criteria with Select statement

Like clause with Select statement

The LIKE keyword selects rows with columns that match specified character string segments. LIKE can be used with data types such as char, varchar, text and datetime. The user can use a wildcard to match fields that include specific letters. The wildcard team name = 'Sri % ', for example, returns all team names that begin with the letters 'Sri.'

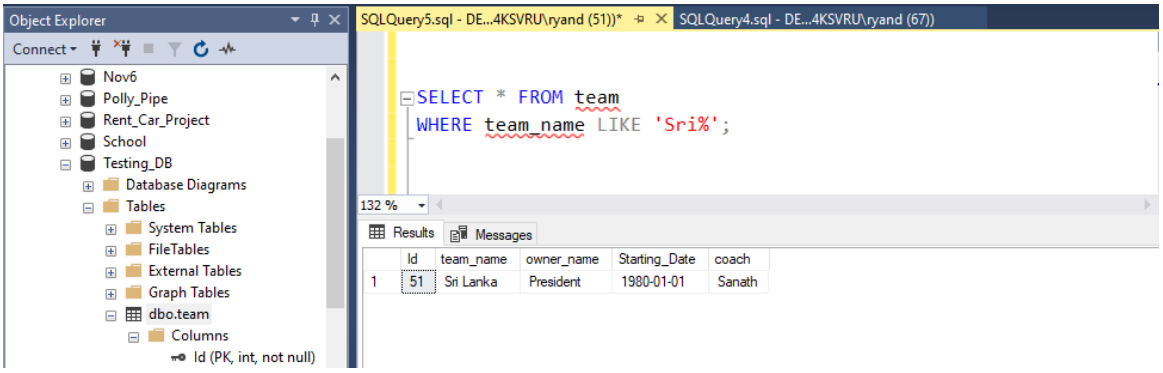


Figure 2. 52 Like clause with Select statement

ORDER BY clause with Select statement

To sort the records in the resulting list, we use the ORDER BY clause. ASC can be used to sort the results in ascending order, while DESC can be used to sort them in descending order.

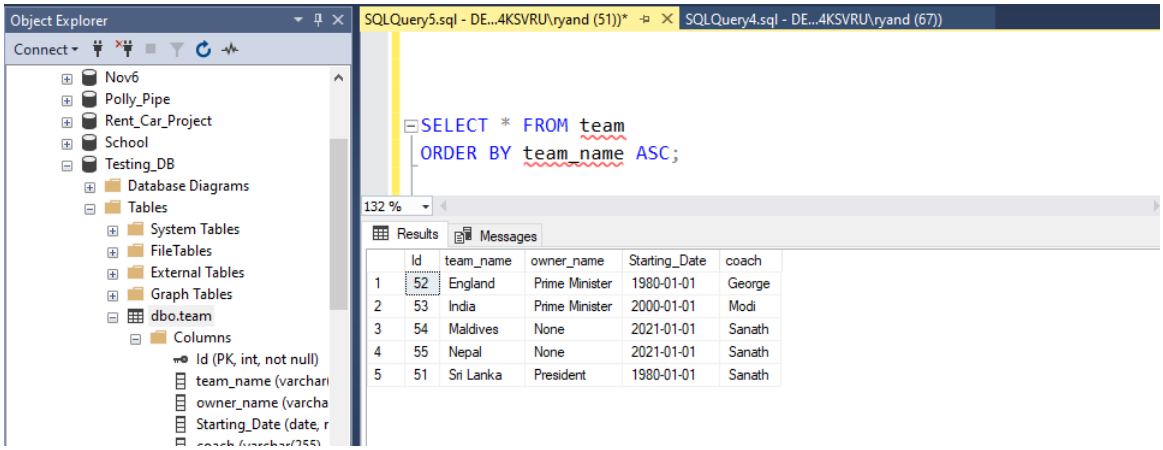


Figure 2. 53 ORDER BY clause with Select statement in ascending order

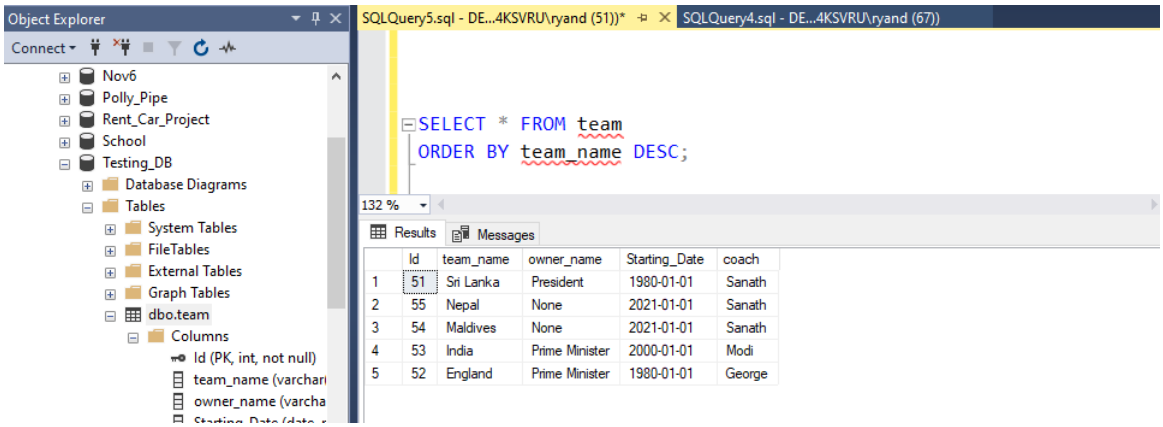


Figure 2. 54 ORDER BY clause with Select statement descending order

GROUP BY clause with Select statement

The group by clause groups the results of a SELECT query according to one or more columns. It can also be used with SQL functions to group data from multiple tables. To explain this this, I've created following table.

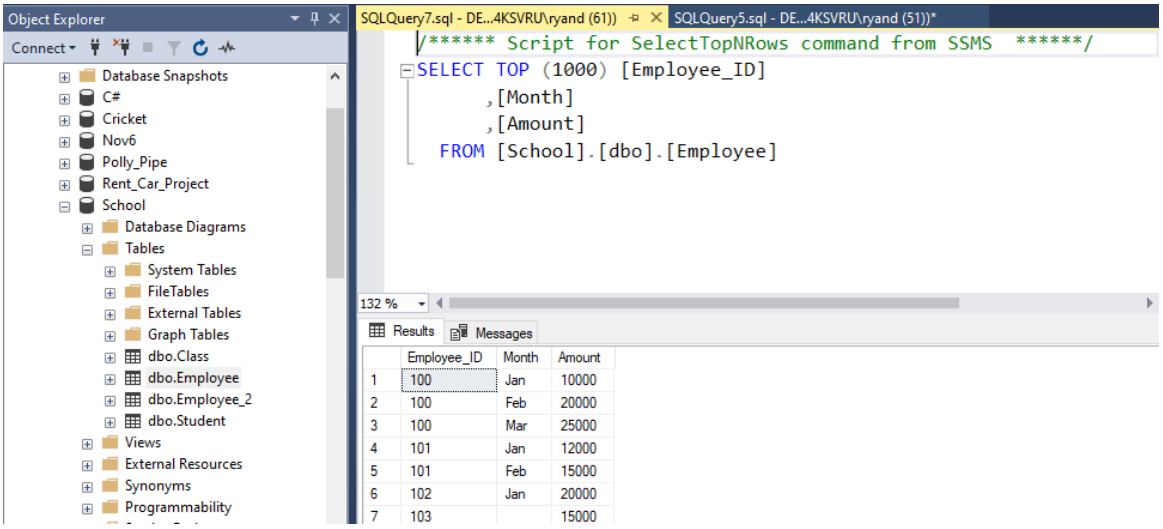


Figure 2. 55 Create new table with data for Group By clause

The GROUP BY statement is frequently used in combination with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to arrange the result-set by one or more columns.

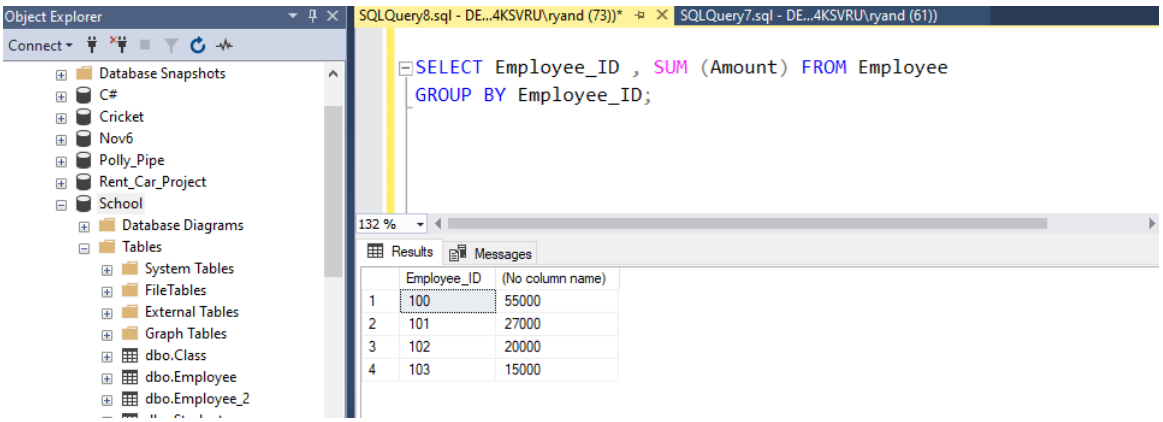


Figure 2. 56 GROUP BY clause with aggregate function SUM

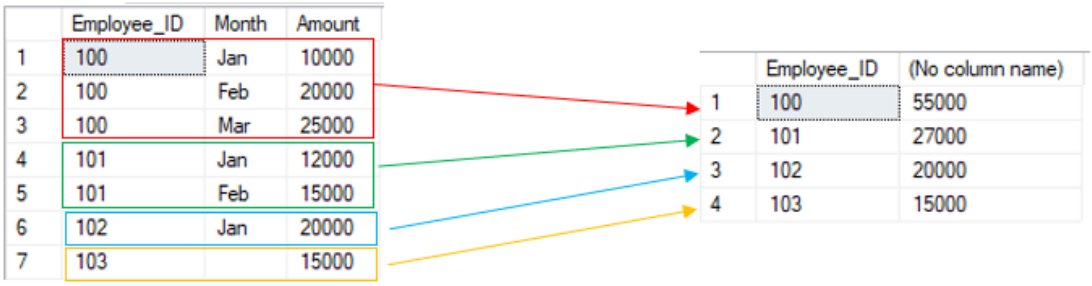


Figure 2. 57 GROUP BY clause with aggregate function SUM :- How this is happened

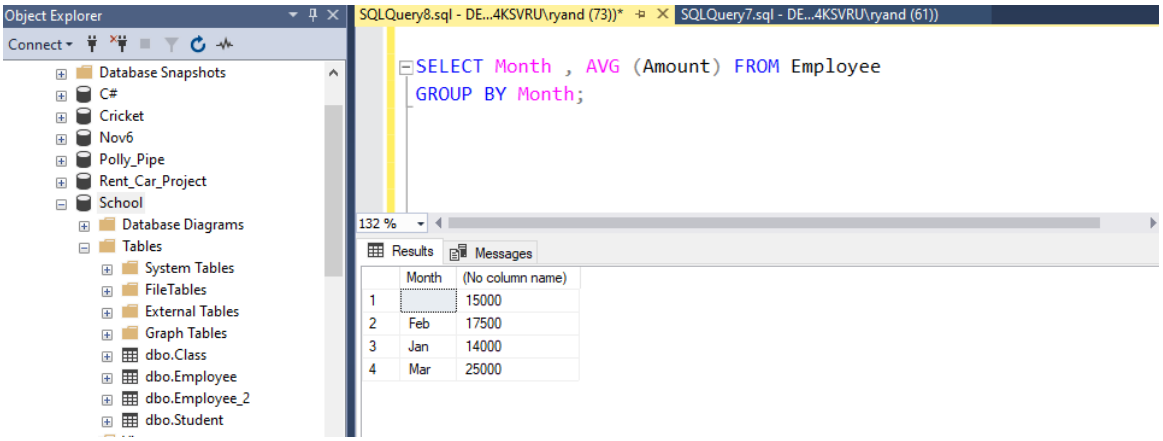


Figure 2. 58 GROUP BY clause with aggregate function AVG

HAVING clause with Select statement

The HAVING clause can be used to limit the number of rows in a table. The difference between HAVING and the WHERE condition is that HAVING can include the aggregate function, while the WHERE cannot.

The HAVING clause works similarly to the WHERE clause, however it only applies to groups. In this case, the HAVING clause is used to filter out to get data whose Employee ID above than 100.

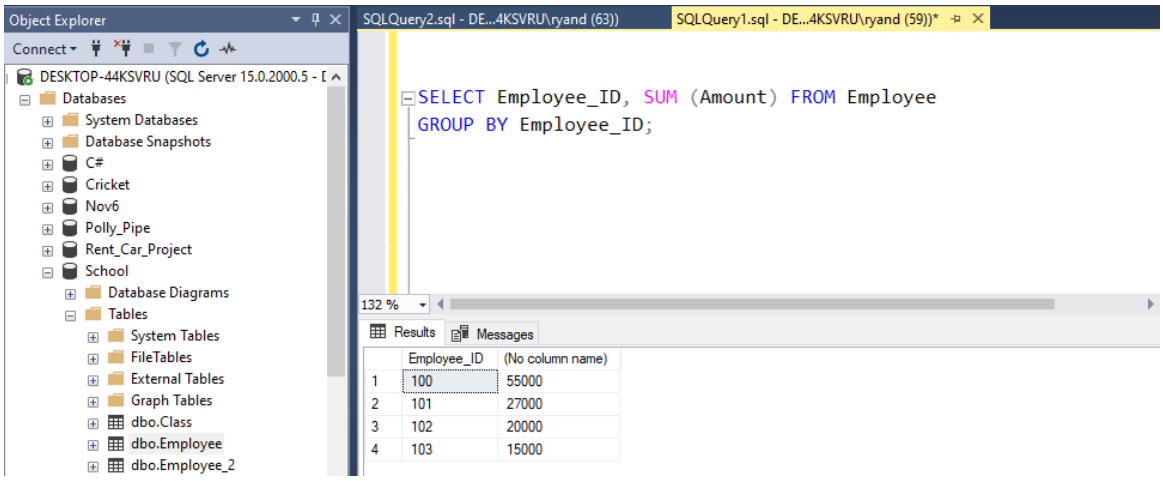


Figure 2. 59 Without Having clause with Select statement

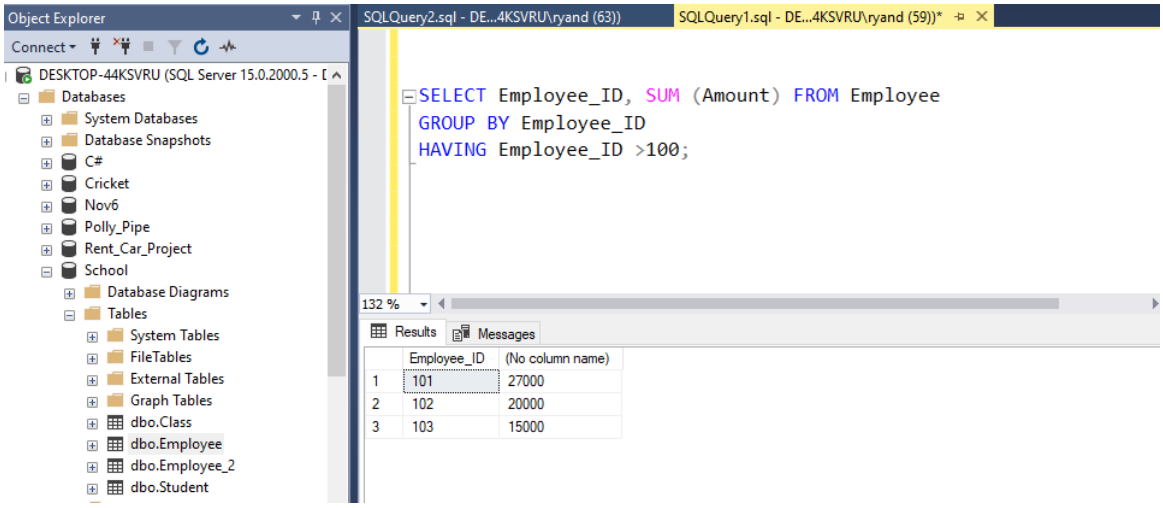


Figure 2. 60 Having clause with Select statement

SQL IN Operator

The SQL IN condition (also known as the IN operator) makes it simple to check whether an expression matches any value in a list of values. It's applied in SELECT, INSERT, UPDATE, and DELETE statements to help eliminate the need for numerous OR conditions.

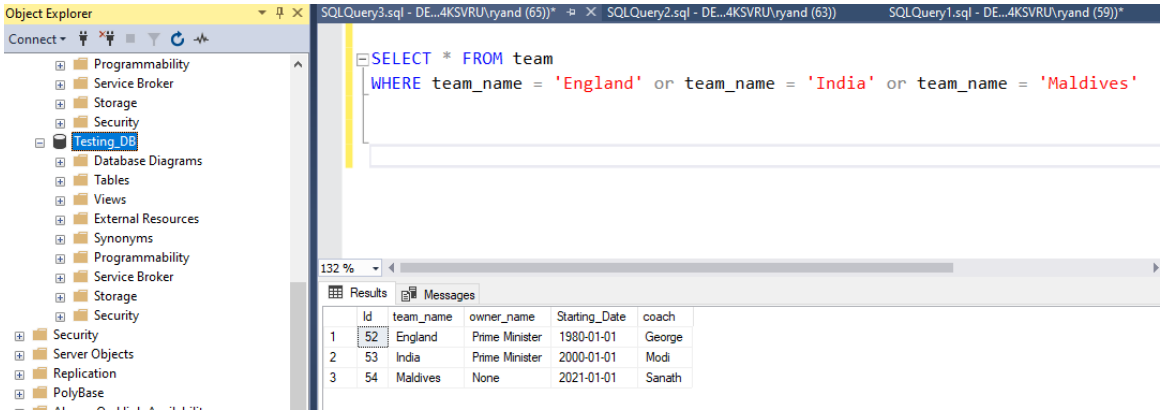


Figure 2. 61 Without In operator with Select statement

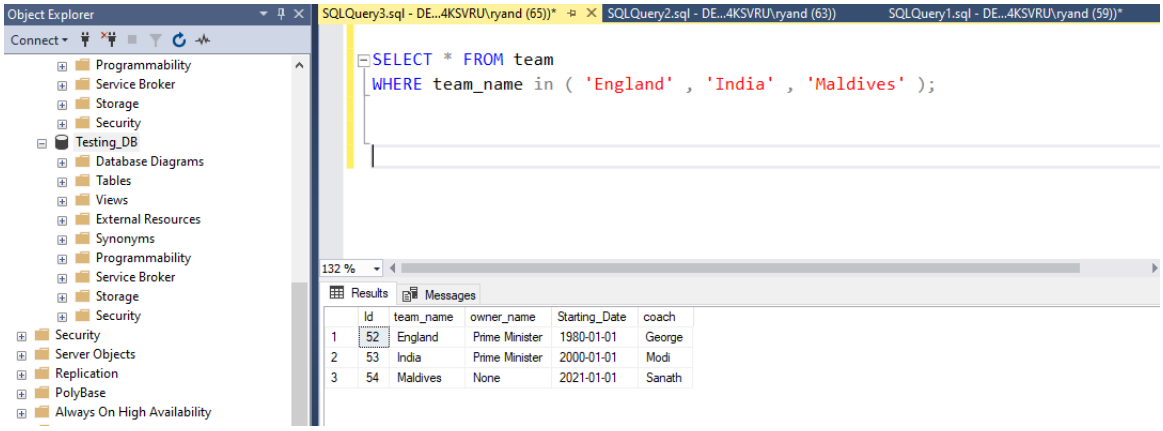


Figure 2. 62 In operator with Select statement

SQL BETWEEN Operator

The BETWEEN operator chooses values from a specified range. Numbers, text, and dates can all be used as values. The BETWEEN operator takes both the start and finish points into consideration.

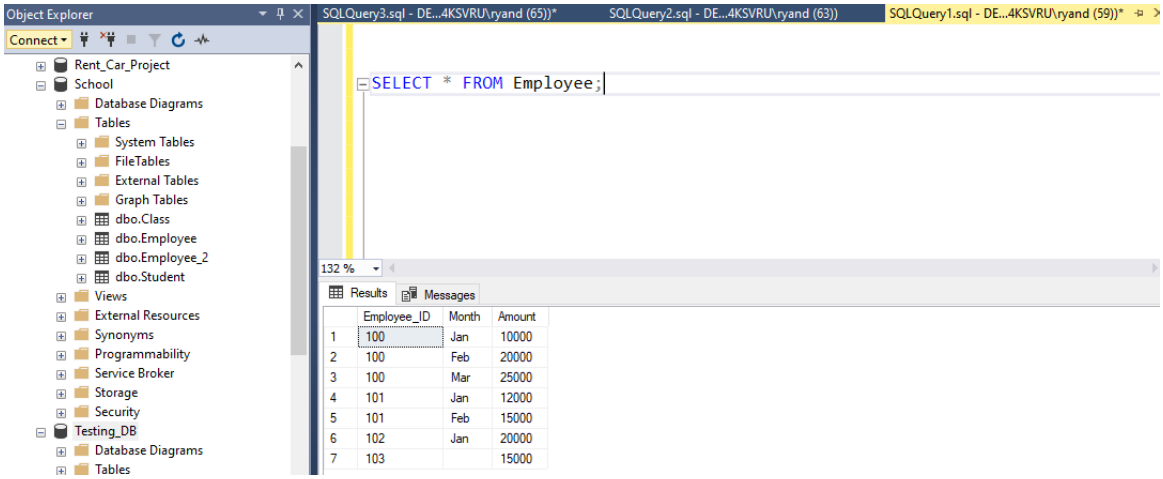


Figure 2. 63 Without Between operator with Select statement

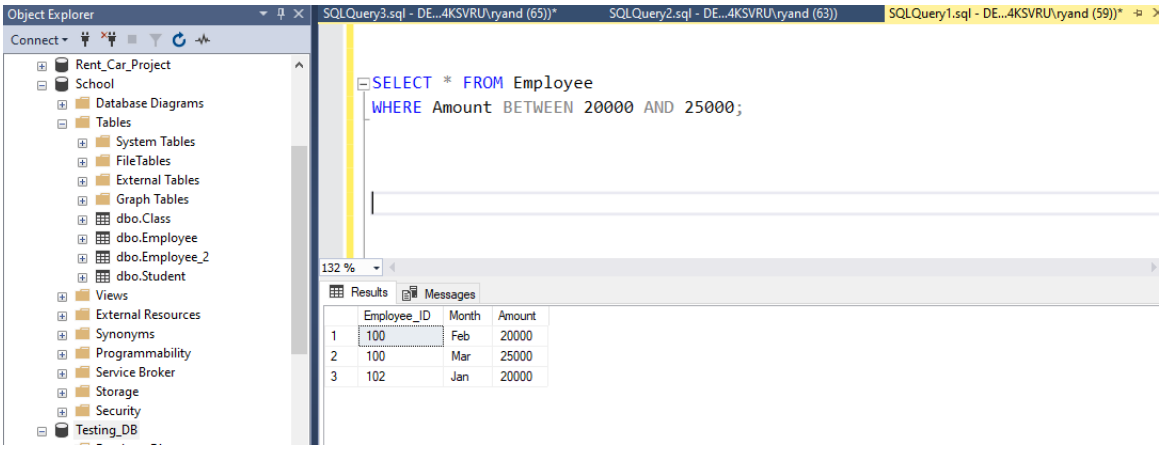


Figure 2. 64 Between operator with Select statement

2.4 Create database for Polly Pipe using SQL DDL statements according to designed ERD and Logical Database

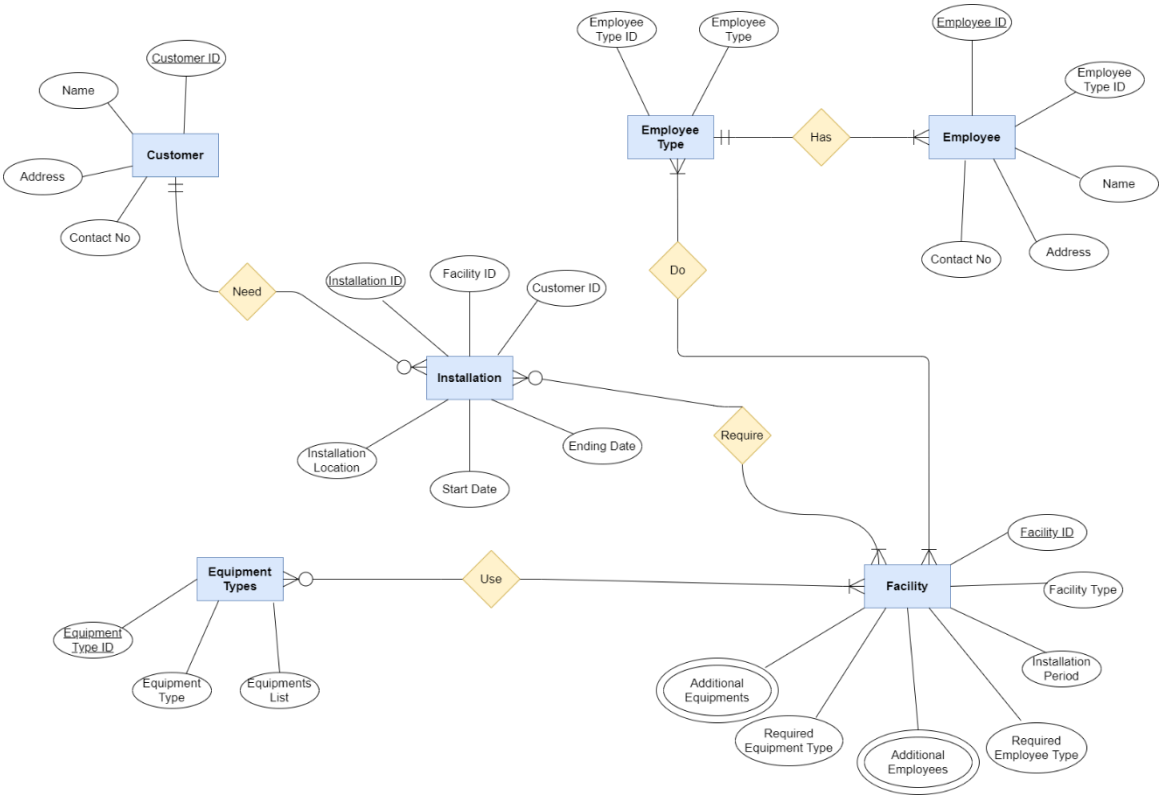


Figure 2. 65 Designed ERD for Polly Pipe

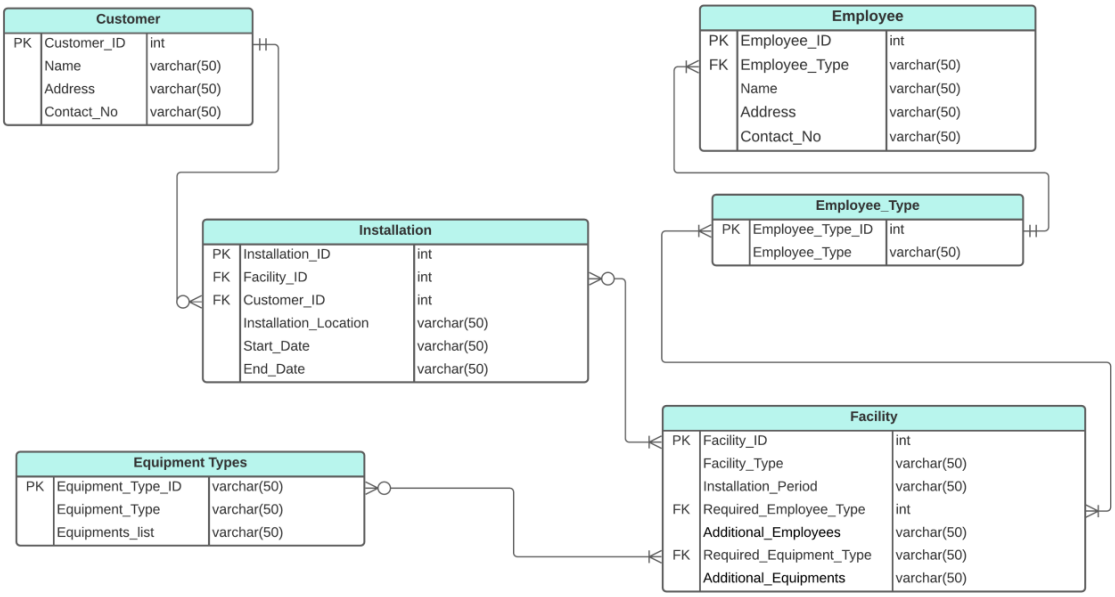


Figure 2. 66 Designed Logical Database for Polly Pipe

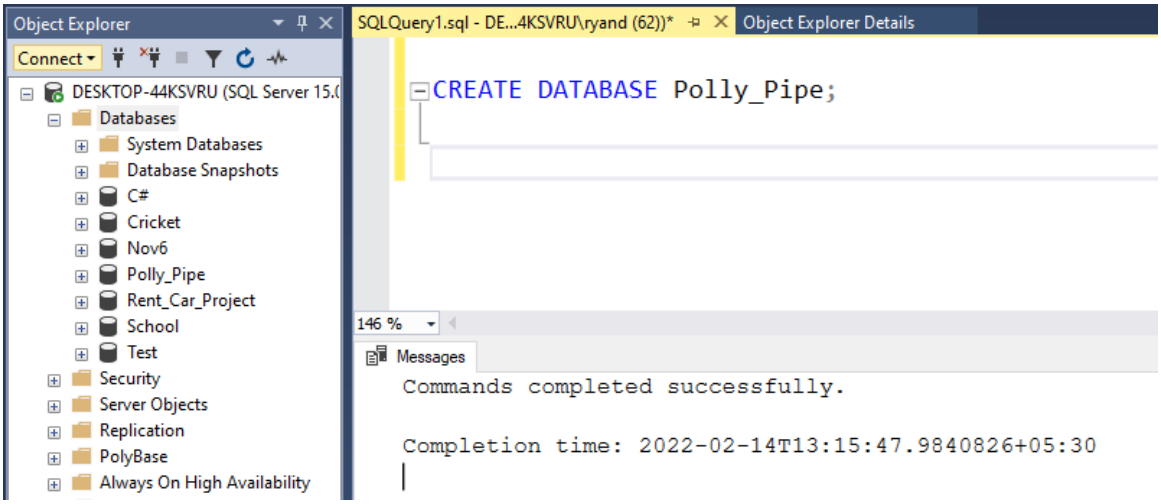


Figure 2. 67 Create Database for Polly Pipe using SQL DDL statements

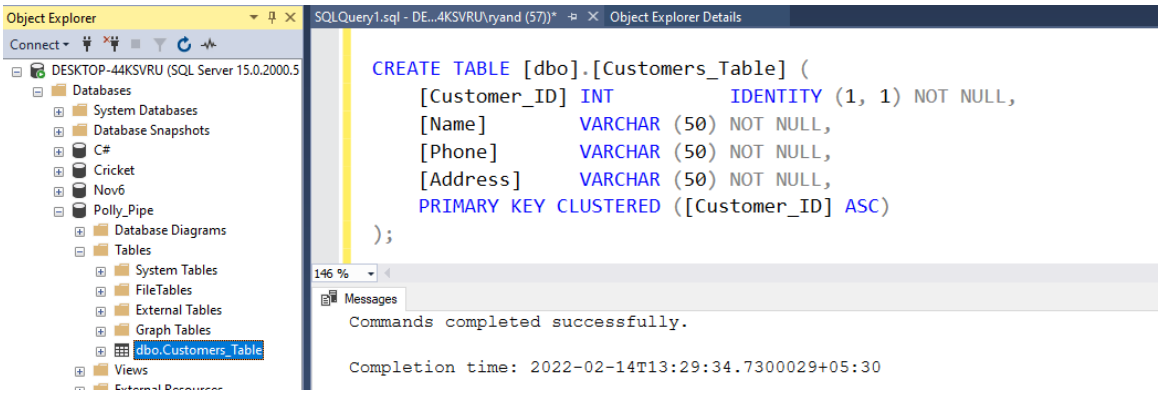


Figure 2. 68 Create Customers Table for Polly Pipe using SQL DDL statements

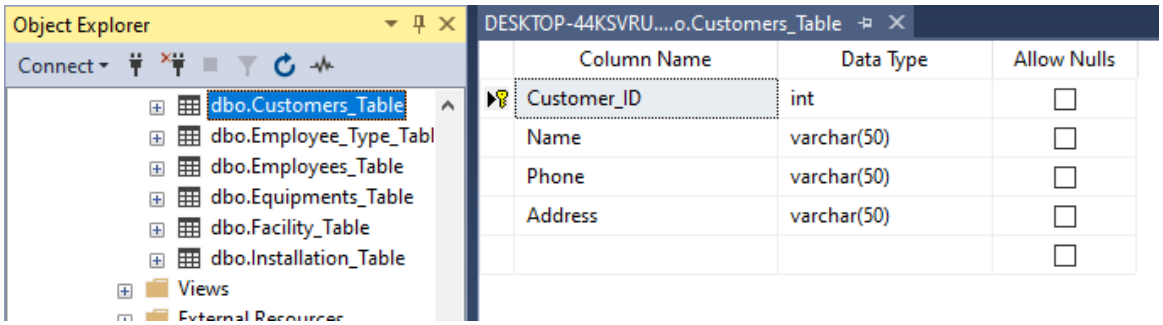


Figure 2. 69 Customers Table output design

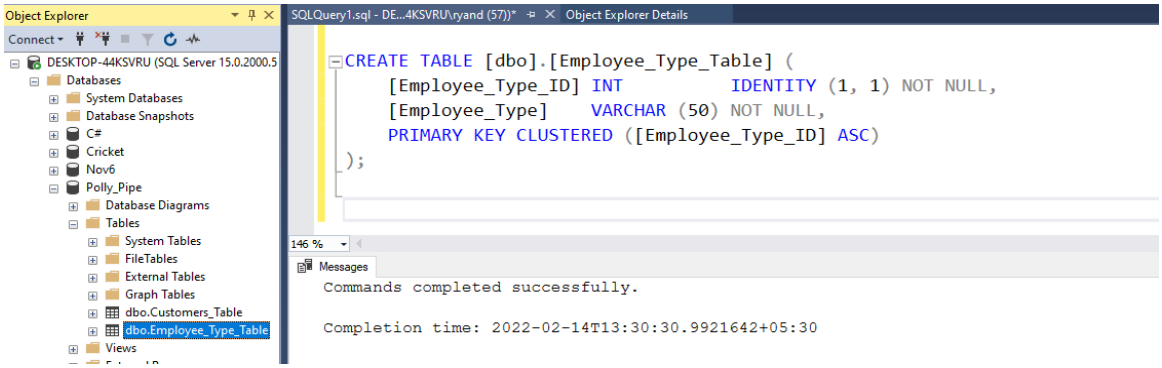


Figure 2. 70 Create Employee Type Table for Polly Pipe using SQL DDL statements

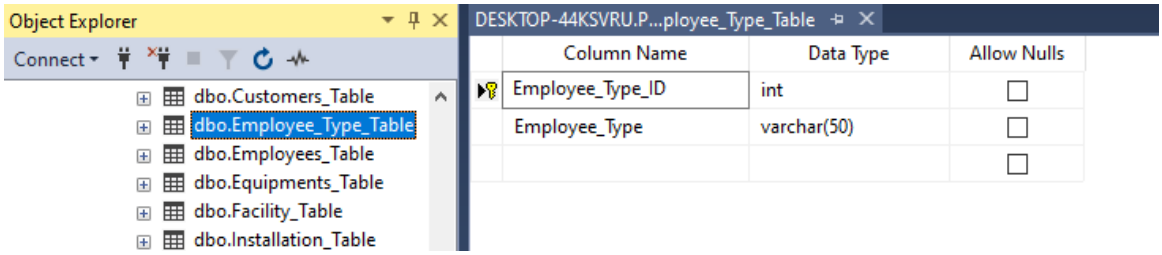


Figure 2. 71 Employee Type Table output design

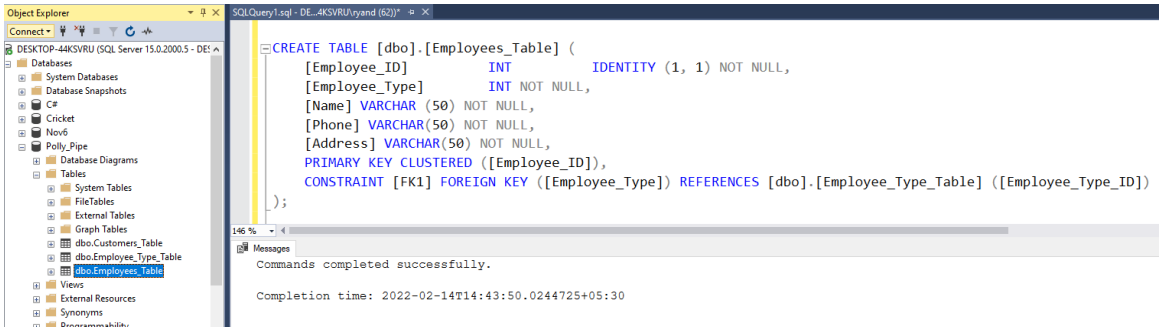


Figure 2. 72 Create Employee Table for Polly Pipe using SQL DDL statements

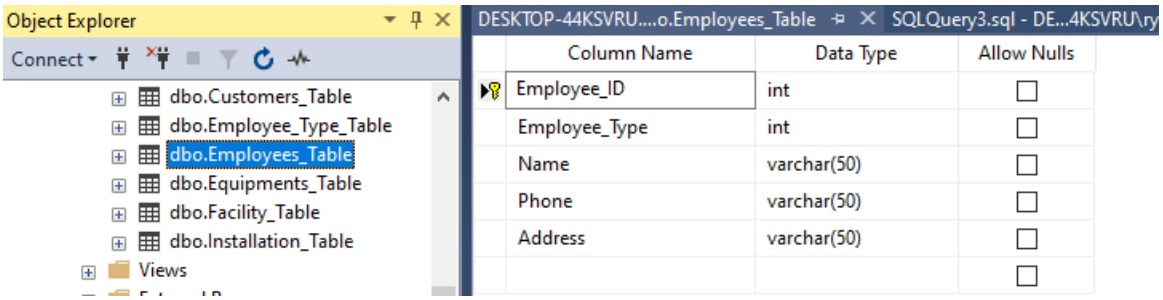


Figure 2. 73 Employee Table output design

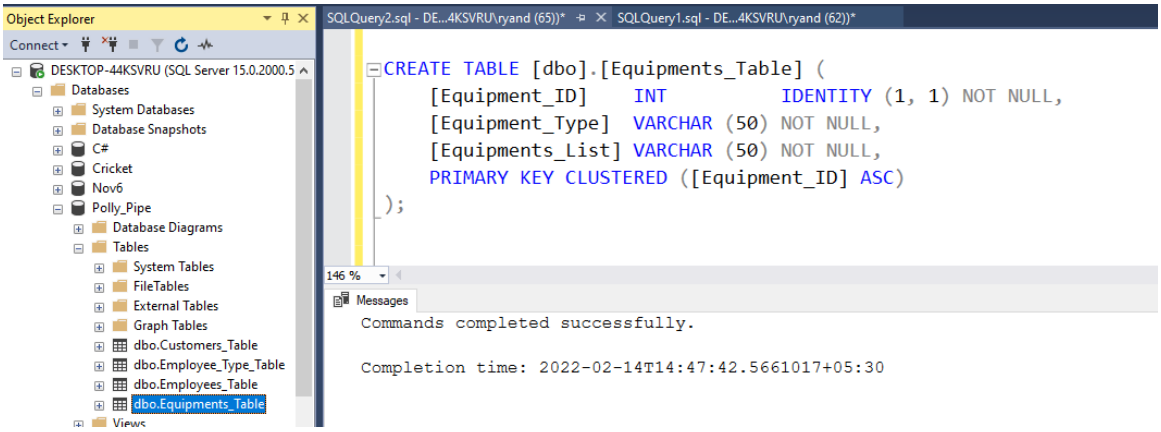


Figure 2. 74 Create Equipment Table for Polly Pipe using SQL DDL statements

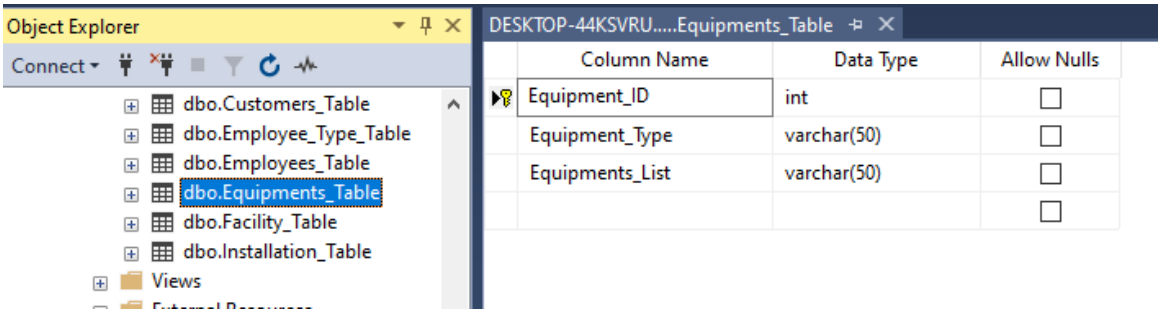


Figure 2. 75 Equipment Table output design

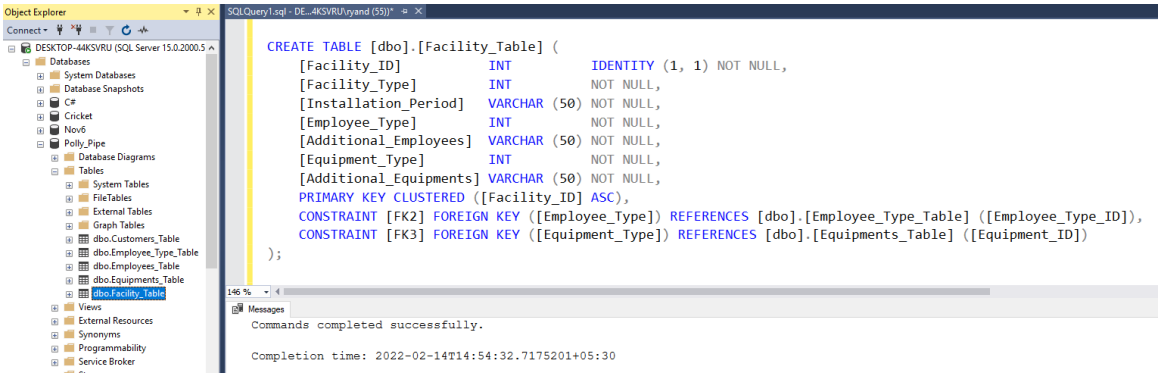


Figure 2. 76 Create Facility Table for Polly Pipe using SQL DDL statements

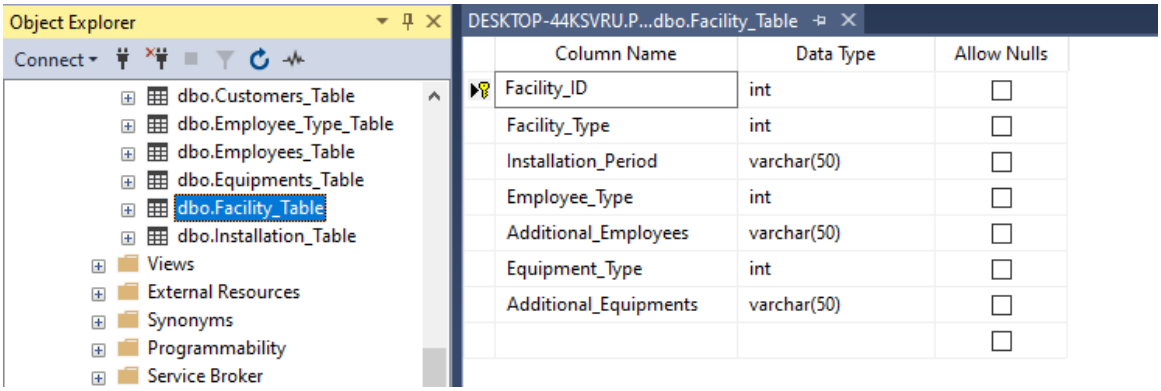


Figure 2. 77 Facility Table output design

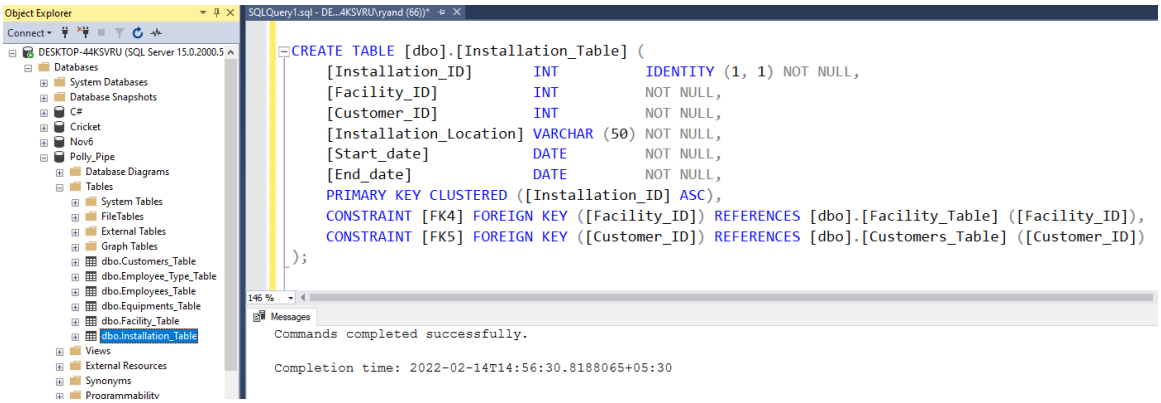


Figure 2. 78 Create Installation Table for Polly Pipe using SQL DDL statements

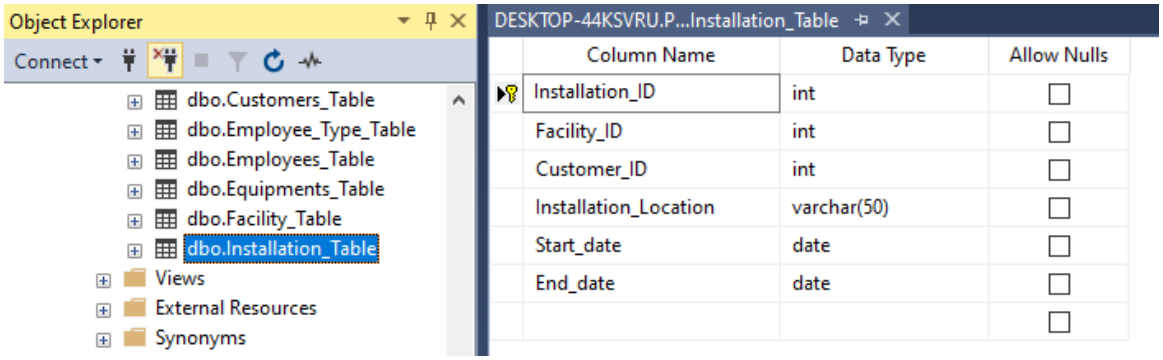


Figure 2. 79 Installation Table output design

2.5 Connecting SQL Polly Pipe Database with C# System

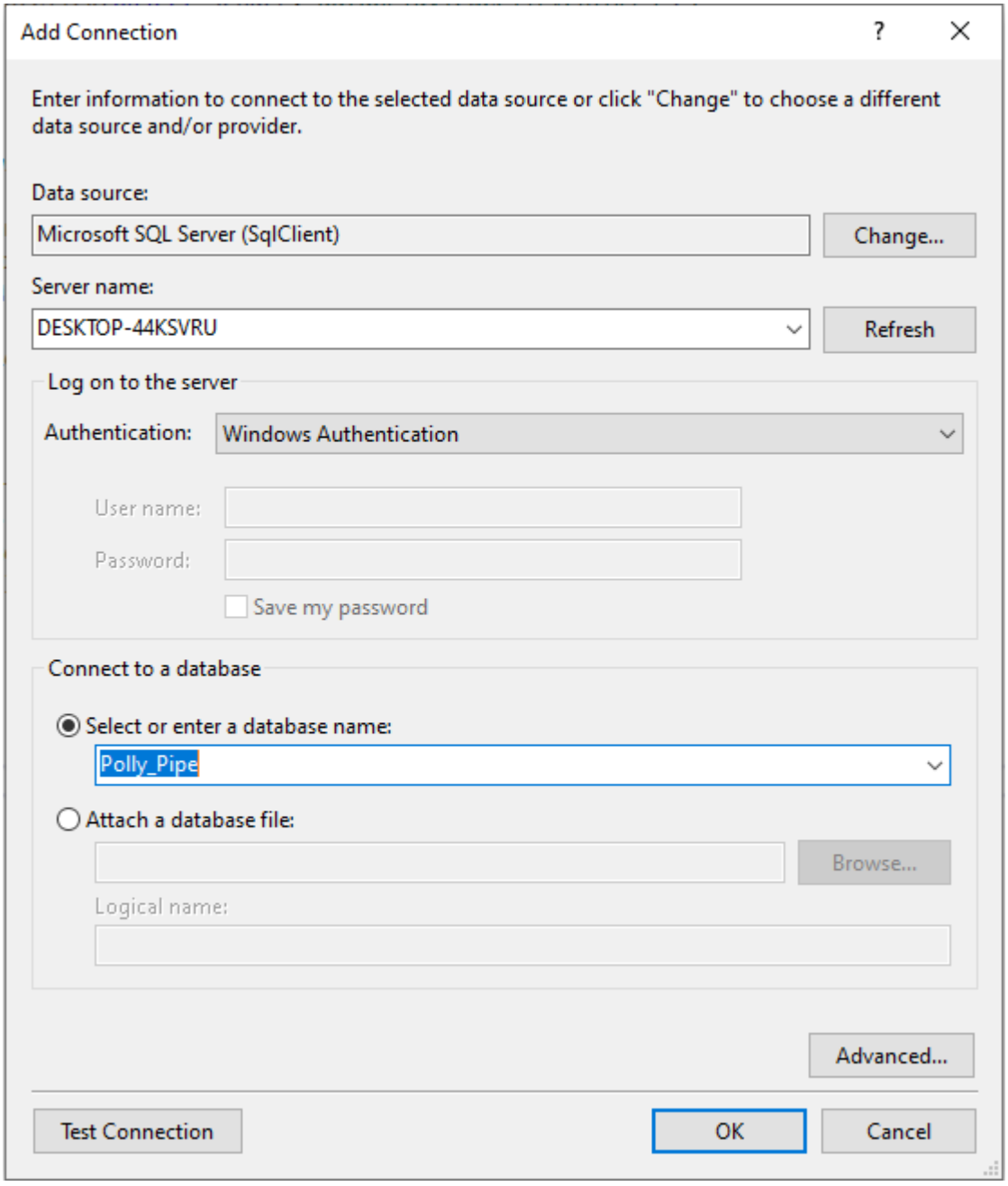


Figure 2. 80 Adding SQL Server connection to Visual Studio C# Server Explorer

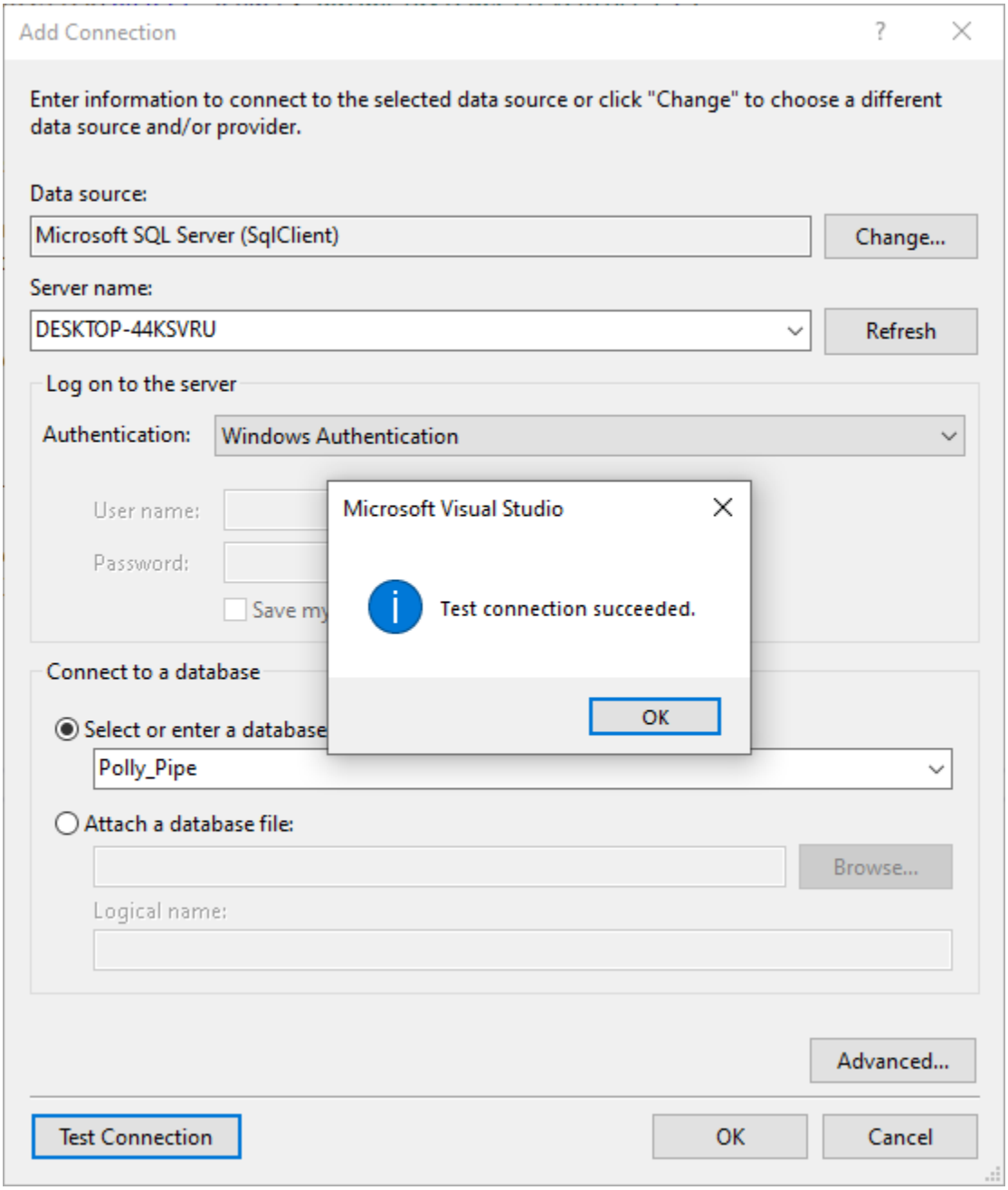


Figure 2. 81 Testing SQL Server connection which added to Visual Studio C# Server Explorer

To connect each page to the SQL, I've added connection string to each form. Below figure illustrate with a sample form what's each attribute of connection string is meant by.

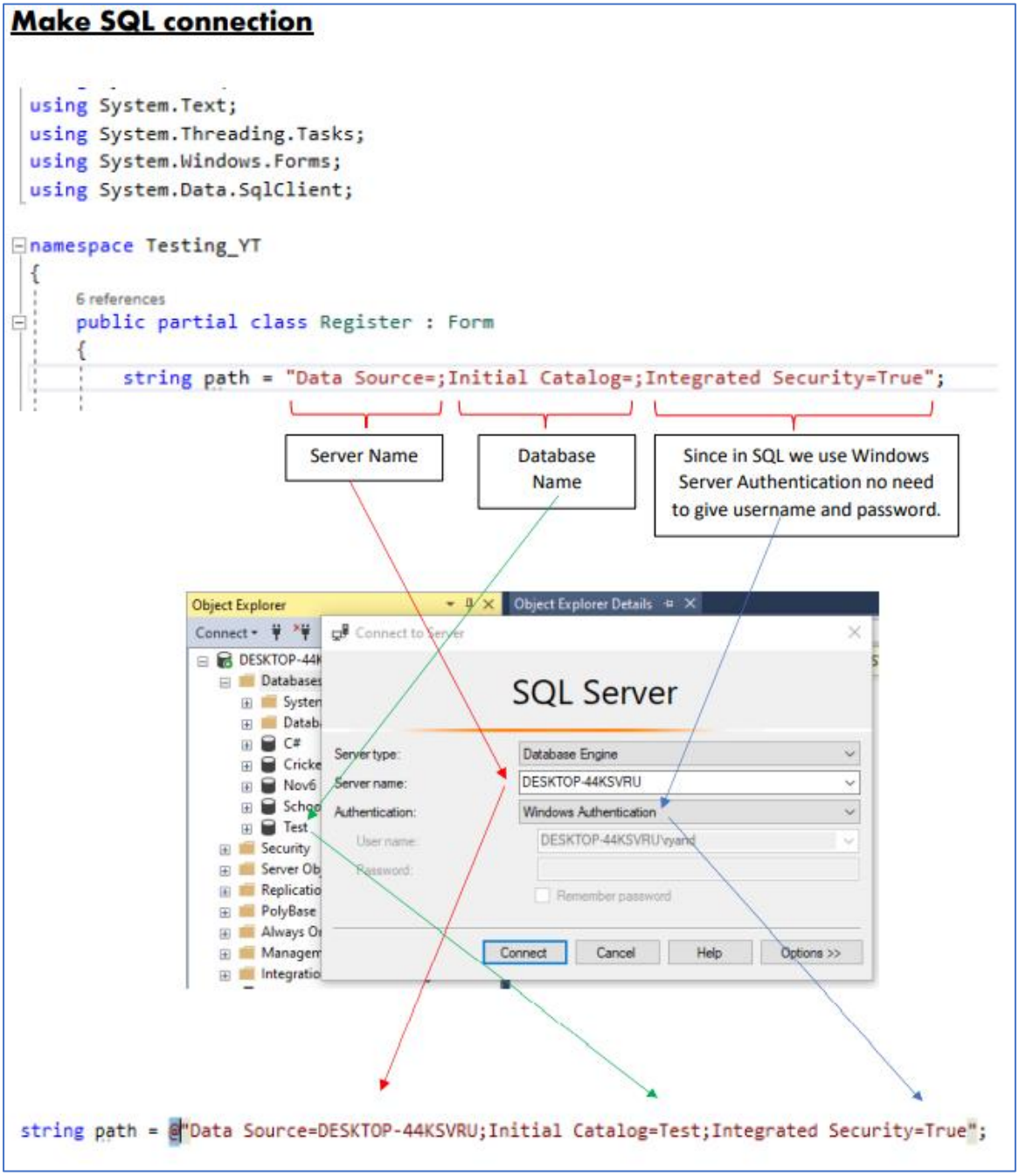


Figure 2. 82 Properties of SQL string path

The below figure here illustrates how I've added connection string to each form with a sample form (Customers Form).

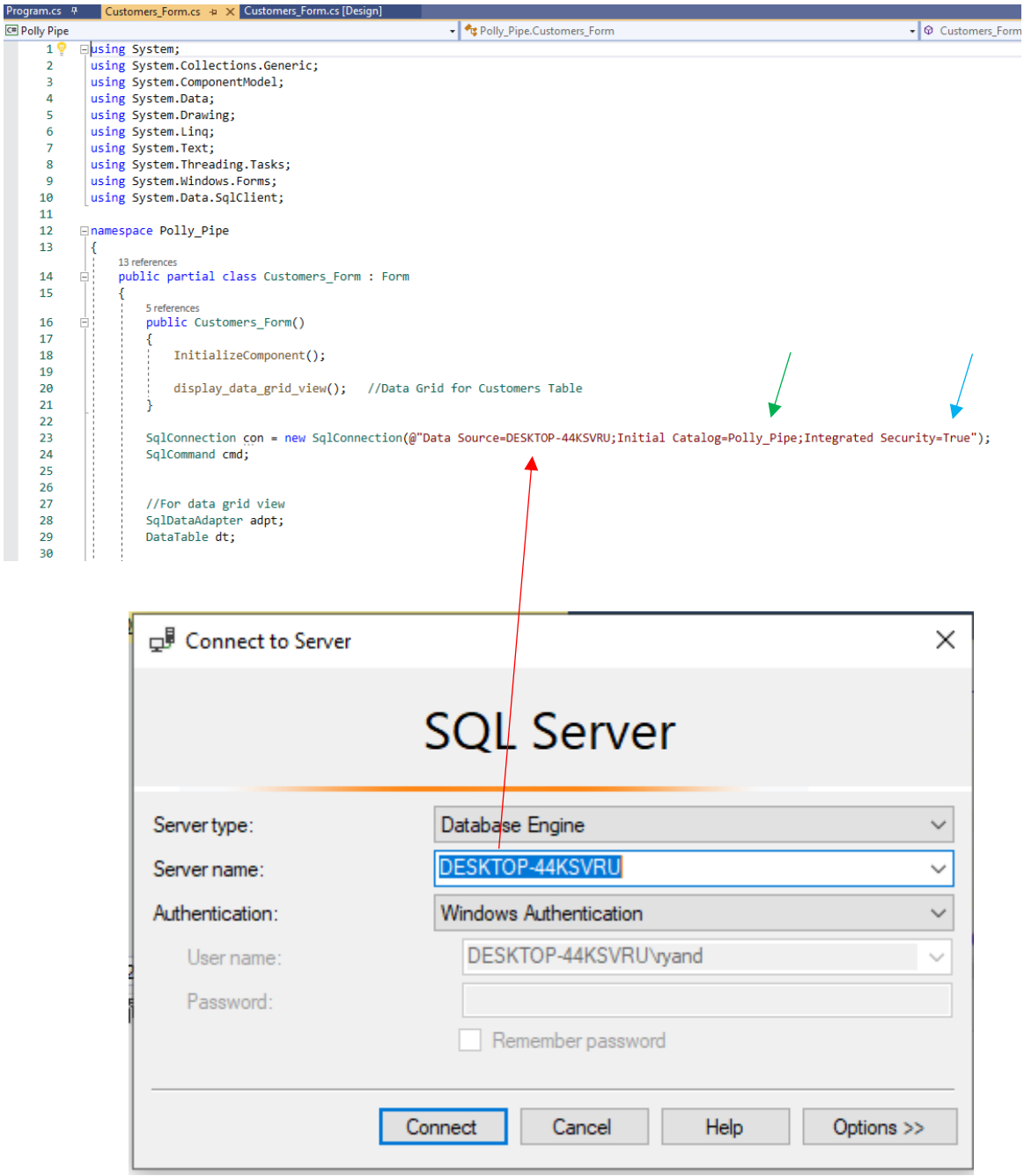


Figure 2. 83 How I added SQL connection string for each form

2.6 Creating a system application for Polly Pipe

2.6.1 Creating C# application with a Login system

Splash Screen Form →

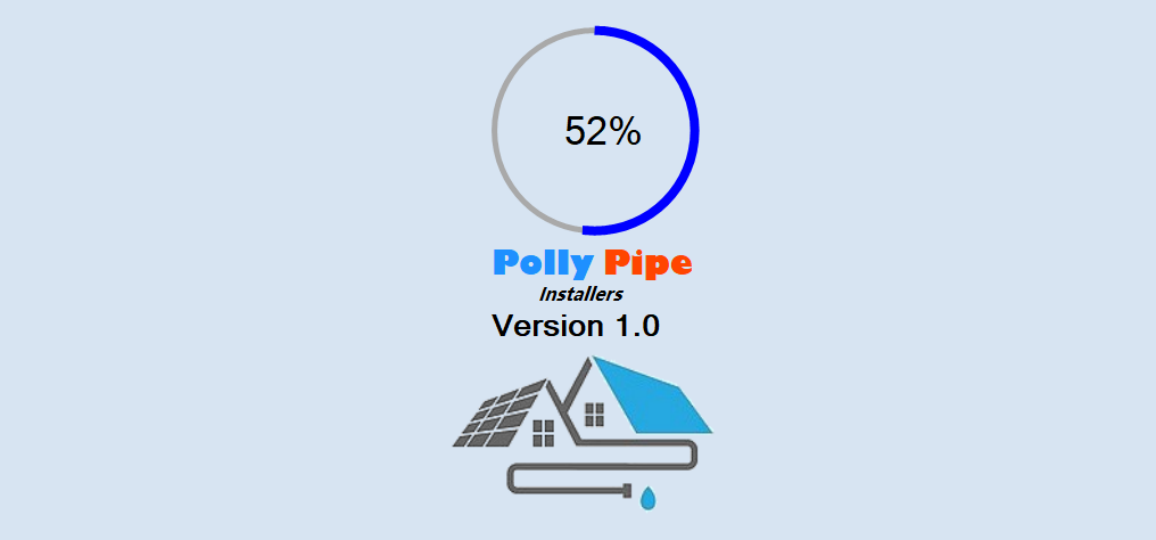


Figure 2. 84 Splash Screen interface

Program.cs | Splash_Screen_Form.cs | Splash_Screen_Form.cs [Design]

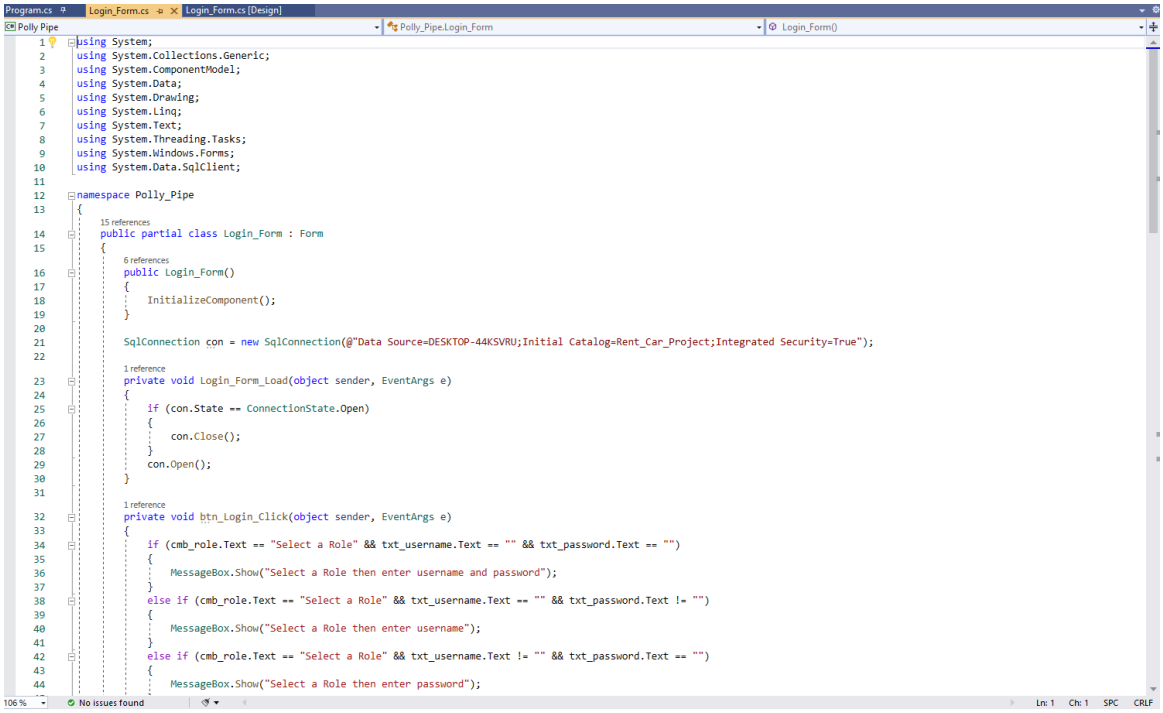
Polly Pipe | Polly_Pipe.Splash_Screen_Form | startpoint

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Polly_Pipe
{
    4 references
    public partial class Splash_Screen_Form : Form
    {
        int startpoint = 0;

        1 reference
        public Splash_Screen_Form()
        {
            InitializeComponent();
        }

        1 reference
        private void timer1_Tick(object sender, EventArgs e)
        {
            bunifuCircleProgressbar1.Value = startpoint;
            startpoint += 10;

            if (bunifuCircleProgressbar1.Value == 100)
            {
                timer1.Stop();

                Login_Form obj = new Login_Form();
                this.Hide();
                obj.Show();
            }
        }

        1 reference
        private void Splash_Screen_Form_Load(object sender, EventArgs e)
        {
            timer1.Start();
        }
    }
}
```

Figure 2. 85 Splash Screen Code - part 1

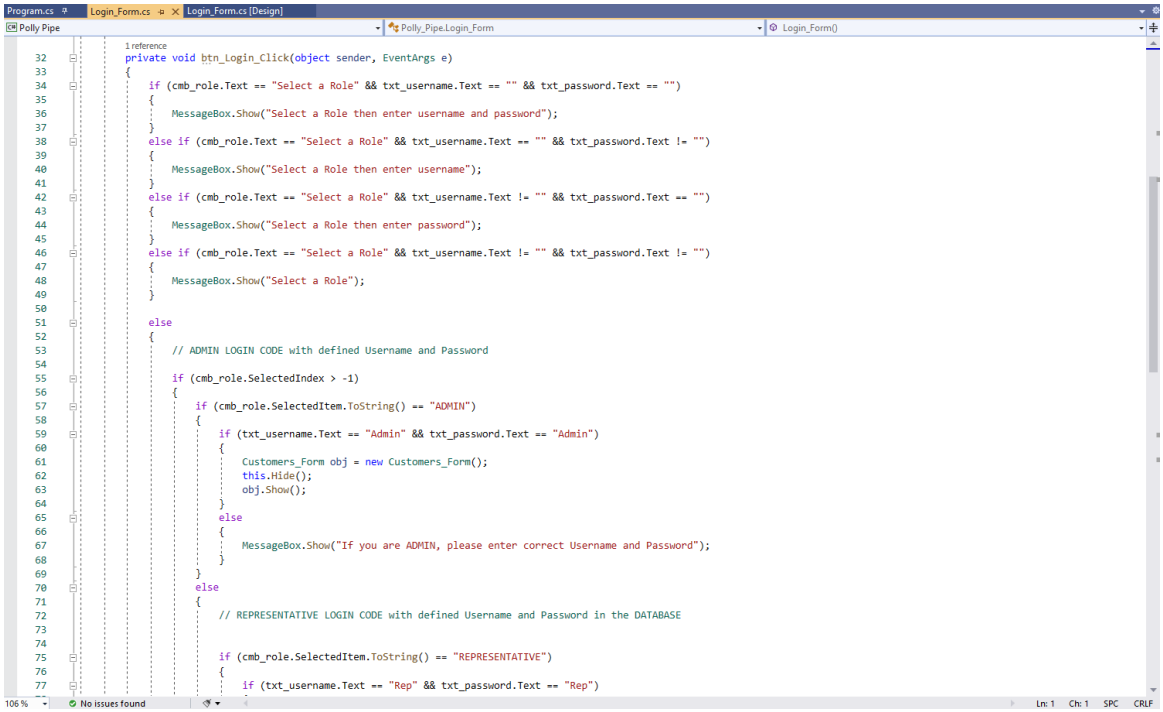
Login Form →

Figure 2. 86 Login form interface



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    15 references
    public partial class Login_Form : Form
    {
        6 references
        public Login_Form()
        {
            InitializeComponent();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");

        1 reference
        private void Login_Form_Load(object sender, EventArgs e)
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
            con.Open();
        }

        1 reference
        private void btn_Login_Click(object sender, EventArgs e)
        {
            if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter username and password");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text != "")
            {
                MessageBox.Show("Select a Role then enter username");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text != "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter password");
```

Figure 2. 87 Login form code - part 1



```
        1 reference
        private void btn_Login_Click(object sender, EventArgs e)
        {
            if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter username and password");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text != "")
            {
                MessageBox.Show("Select a Role then enter username");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text != "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter password");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text != "" && txt_password.Text != "")
            {
                MessageBox.Show("Select a Role");
            }

            else
            {
                // ADMIN LOGIN CODE with defined Username and Password

                if (cmb_role.SelectedIndex > -1)
                {
                    if (cmb_role.SelectedItem.ToString() == "ADMIN")
                    {
                        if (txt_username.Text == "Admin" && txt_password.Text == "Admin")
                        {
                            Customers_Form obj = new Customers_Form();
                            this.Hide();
                            obj.Show();
                        }
                        else
                        {
                            MessageBox.Show("If you are ADMIN, please enter correct Username and Password");
                        }
                    }
                    else
                    {
                        // REPRESENTATIVE LOGIN CODE with defined Username and Password in the DATABASE


                        if (cmb_role.SelectedItem.ToString() == "REPRESENTATIVE")
                        {
                            if (txt_username.Text == "Rep" && txt_password.Text == "Rep")
```

Figure 2. 88 Login form code - part 2

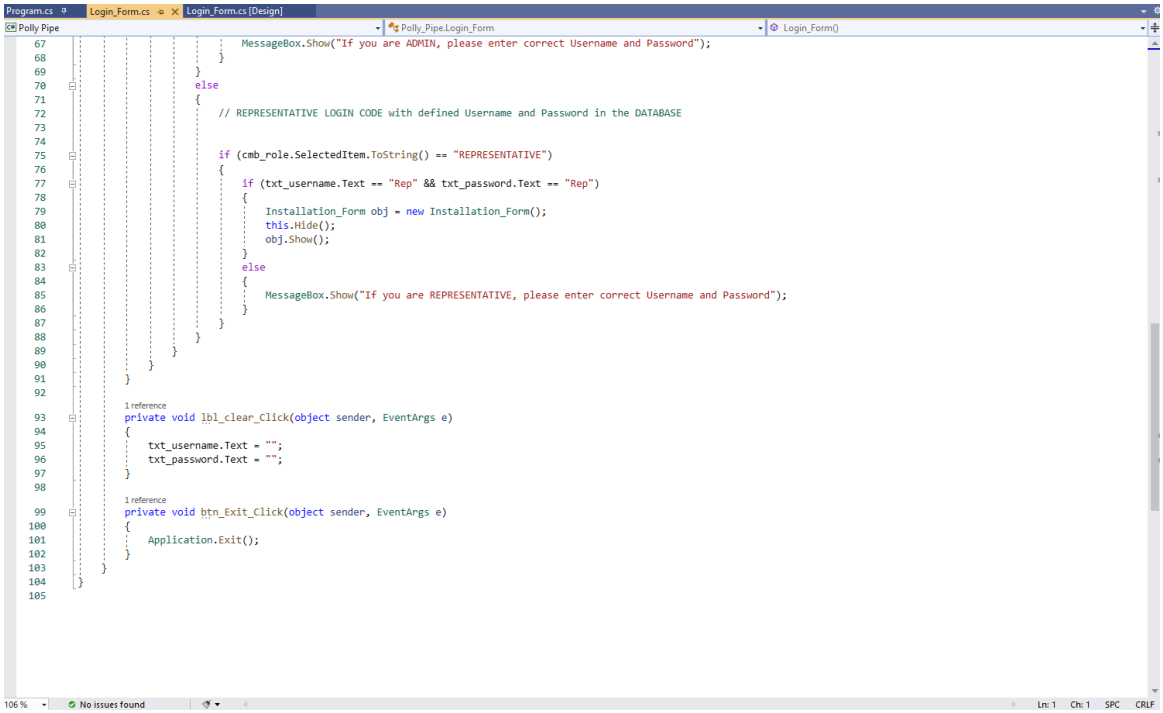


Figure 2. 89 Login form code - part 3

Customer Form (For Admin) →

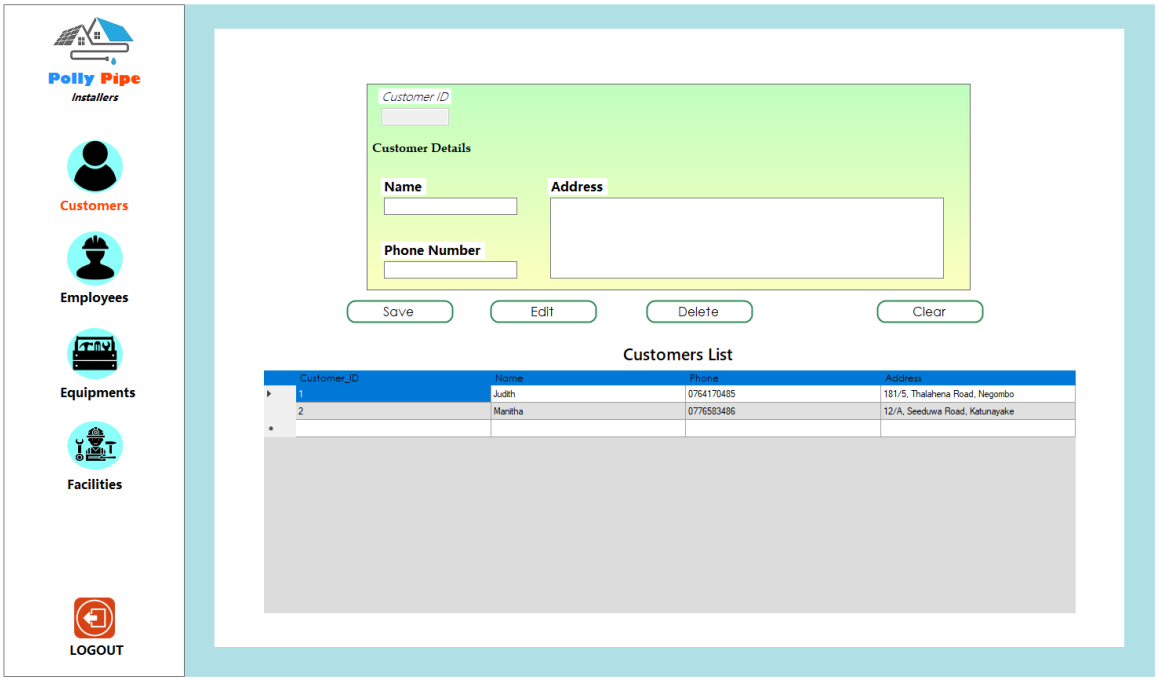
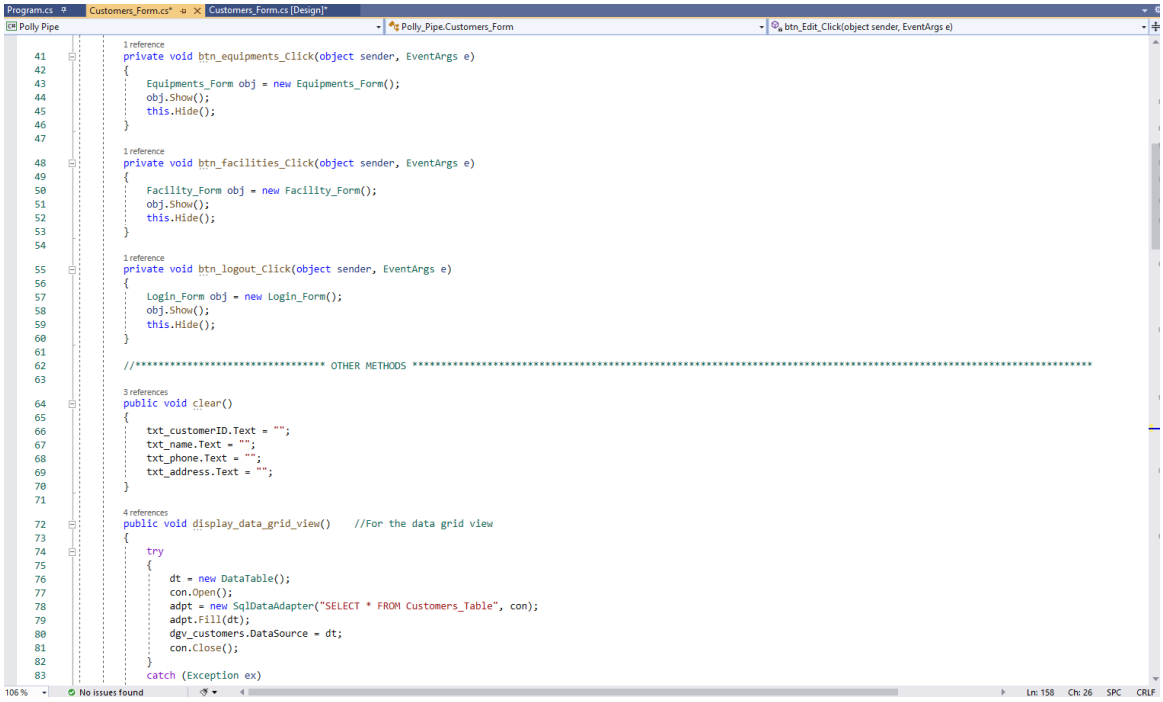


Figure 2. 90 Customer Form (For Admin) interface

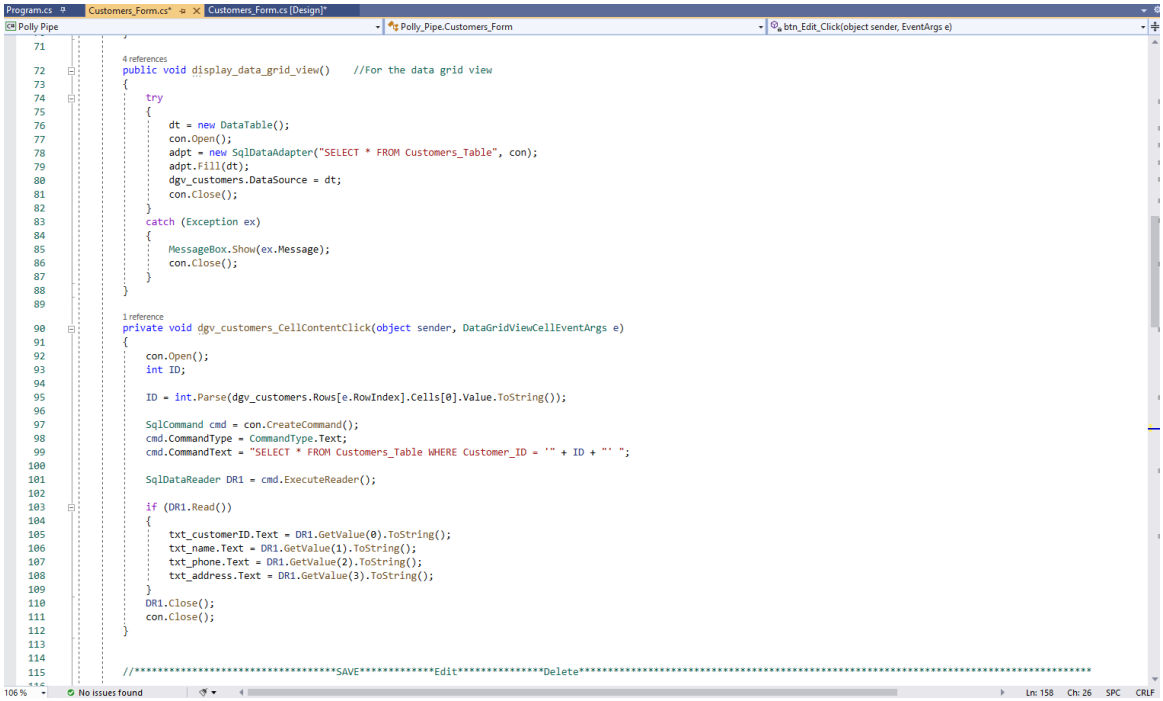
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    13 references
    public partial class Customers_Form : Form
    {
        5 references
        public Customers_Form()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Customers Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***********************************************************************************************************

        1 reference
        private void btn_employees_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_equipments_Click(object sender, EventArgs e)
        {
            Equipments_Form obj = new Equipments_Form();
            obj.Show();
```

Figure 2. 91 Customer Form (For Admin) code - part 1



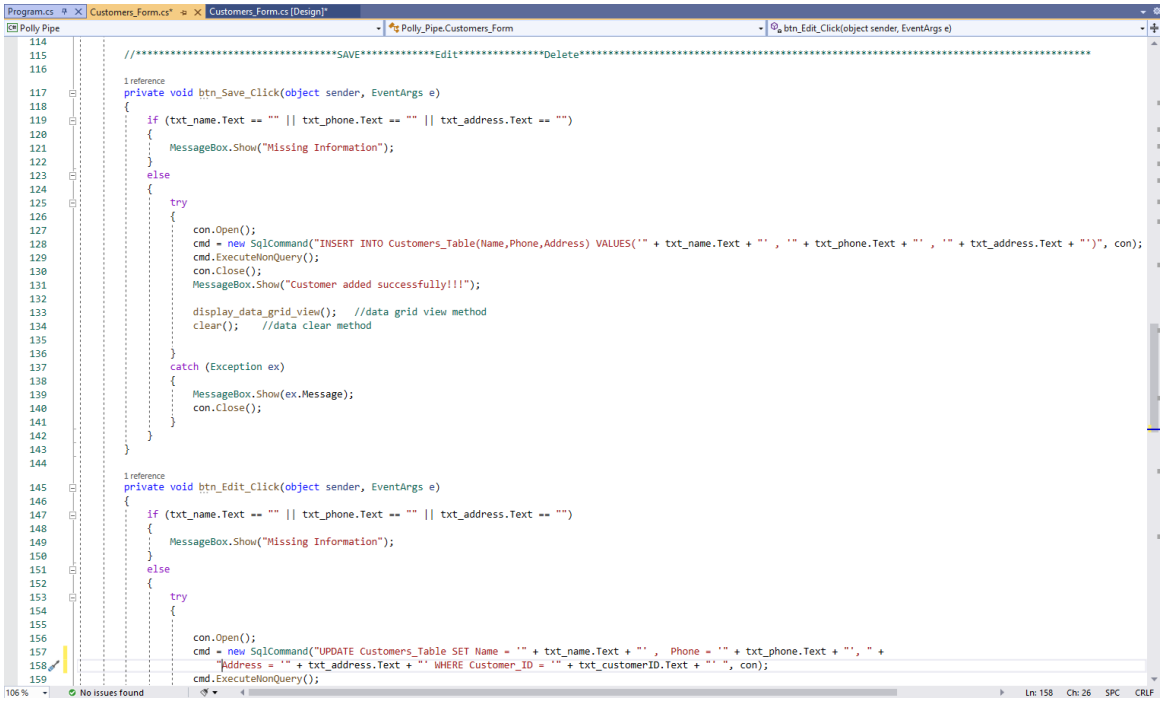
```
1 reference
private void btn_equipments_Click(object sender, EventArgs e)
{
    Equipments_Form obj = new Equipments_Form();
    obj.Show();
    this.Hide();
}

1 reference
private void btn_facilities_Click(object sender, EventArgs e)
{
    Facility_Form obj = new Facility_Form();
    obj.Show();
    this.Hide();
}

1 reference
private void btn_logout_Click(object sender, EventArgs e)
{
    Login_Form obj = new Login_Form();
    obj.Show();
    this.Hide();
}

//********************************* OTHER METHODS ***********************************************************************************************************

3 references
public void clear()
{
    txt_customerID.Text = "";
    txt_name.Text = "";
    txt_phone.Text = "";
    txt_address.Text = "";
}

4 references
public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
        adpt.Fill(dt);
        dgv_customers.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
```

Figure 2. 92 Customer Form (For Admin) code - part 2



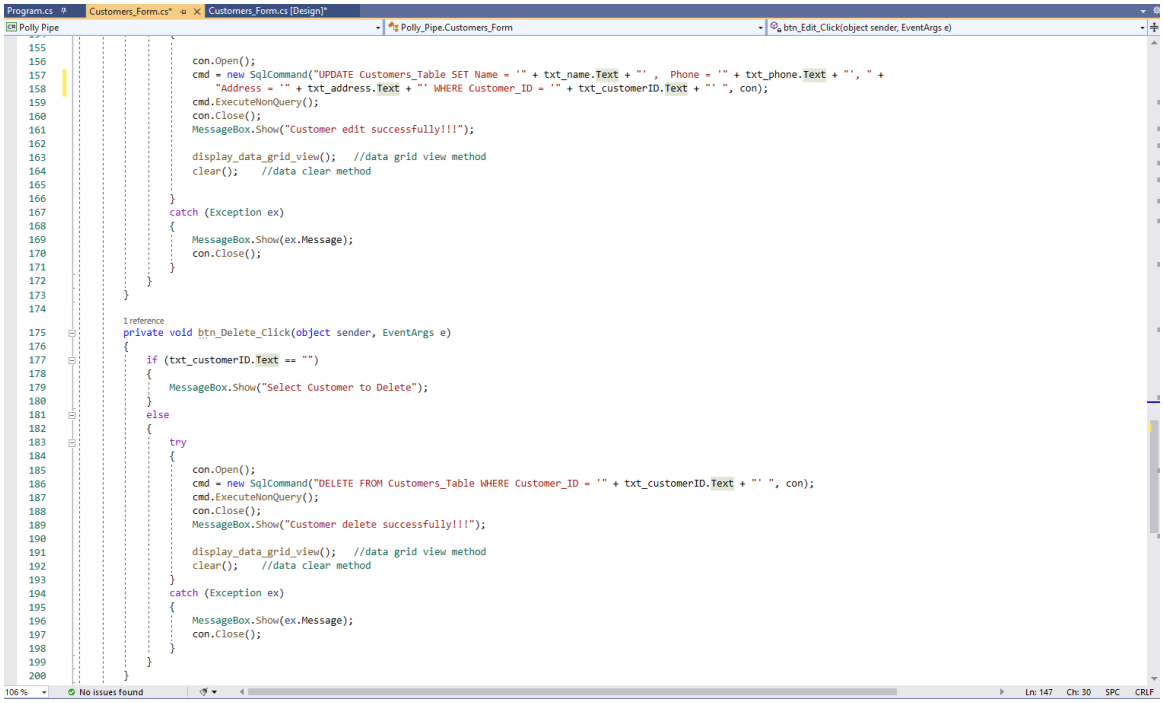
```
4 references
public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
        adpt.Fill(dt);
        dgv_customers.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Customers_Table WHERE Customer_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_customerID.Text = DR1.GetValue(0).ToString();
        txt_name.Text = DR1.GetValue(1).ToString();
        txt_phone.Text = DR1.GetValue(2).ToString();
        txt_address.Text = DR1.GetValue(3).ToString();
    }
    DR1.Close();
    con.Close();
}


//*************************************SAVE************Edit**************Delete*****************************************************************************
```

Figure 2. 93 Customer Form (For Admin) code - part 3



```
//**********SAVE**********Edit**********Delete**********

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Customers_Table(Name,Phone,Address) VALUES('" + txt_name.Text + "' , '" + txt_phone.Text + "' , '" + txt_address.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer added successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
            cmd = new SqlCommand("UPDATE Customers_Table SET Name = '" + txt_name.Text + "' ,  Phone = '" + txt_phone.Text + "', " +
                "Address = '" + txt_address.Text + "' WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
            cmd.ExecuteNonQuery();
```

Figure 2. 94 Customer Form (For Admin) code - part 4



```
            con.Open();
            cmd = new SqlCommand("UPDATE Customers_Table SET Name = '" + txt_name.Text + "' ,  Phone = '" + txt_phone.Text + "', " +
                "Address = '" + txt_address.Text + "' WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer edit successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_customerID.Text == "")
    {
        MessageBox.Show("Select Customer to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Customers_Table WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer delete successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}
```

Figure 2. 95 Customer Form (For Admin) code - part 5

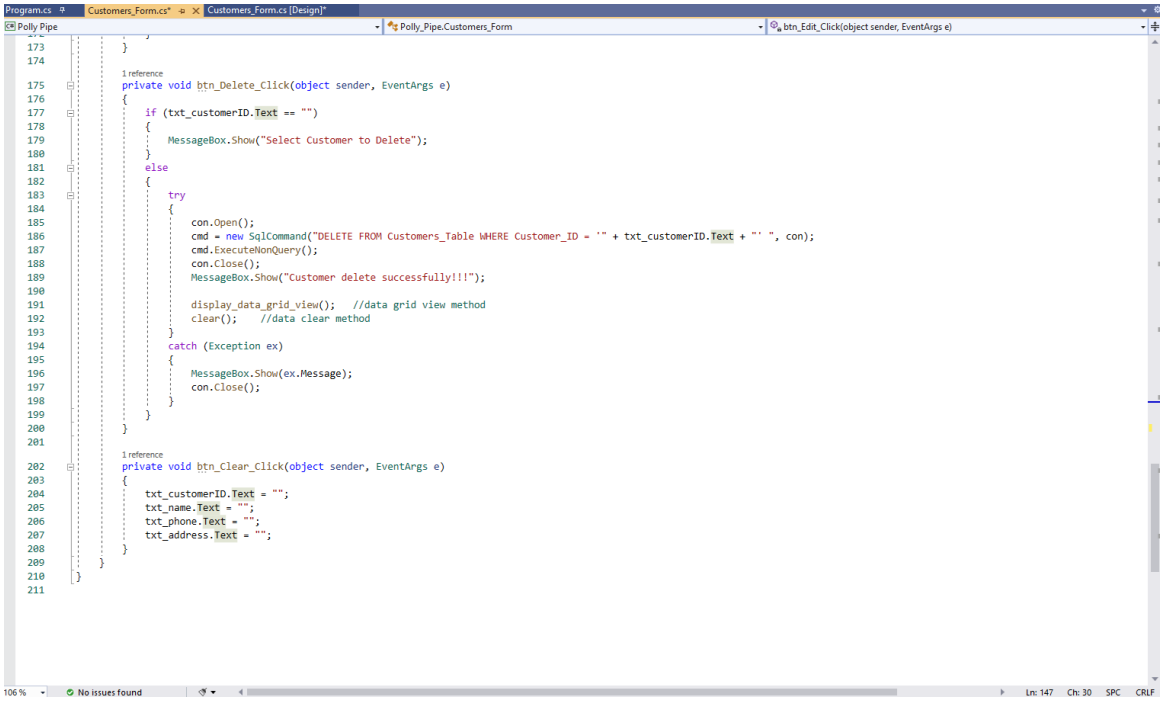


Figure 2. 96 Customer Form (For Admin) code - part 6

Employee Form (For Admin) →

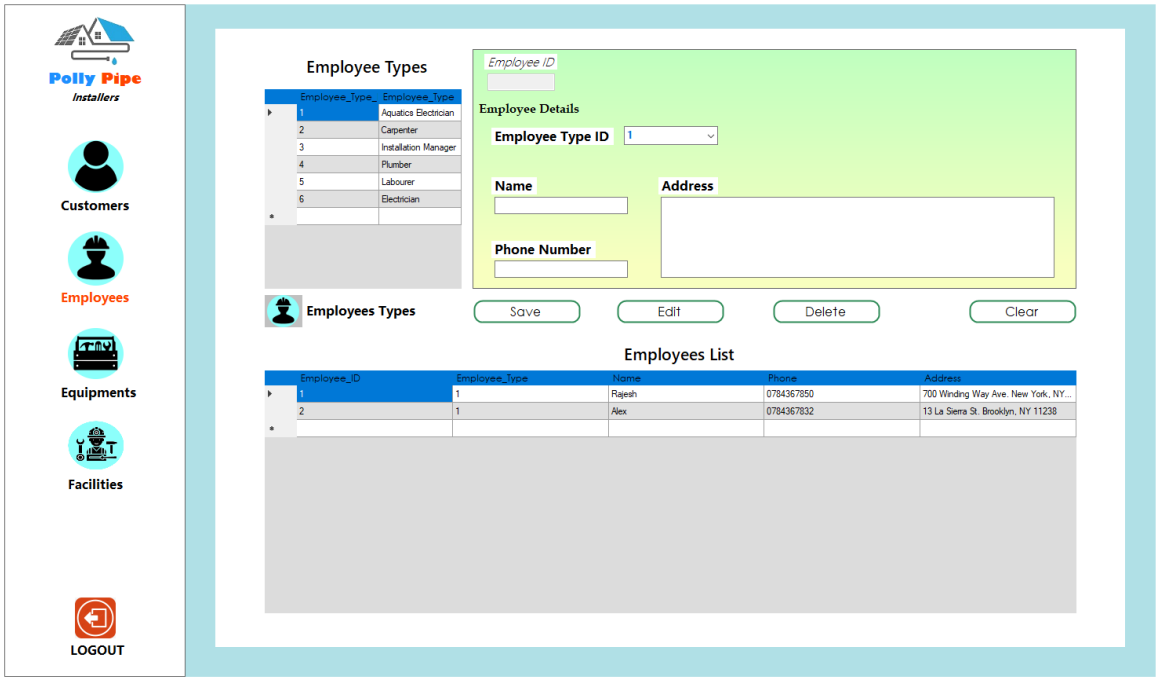
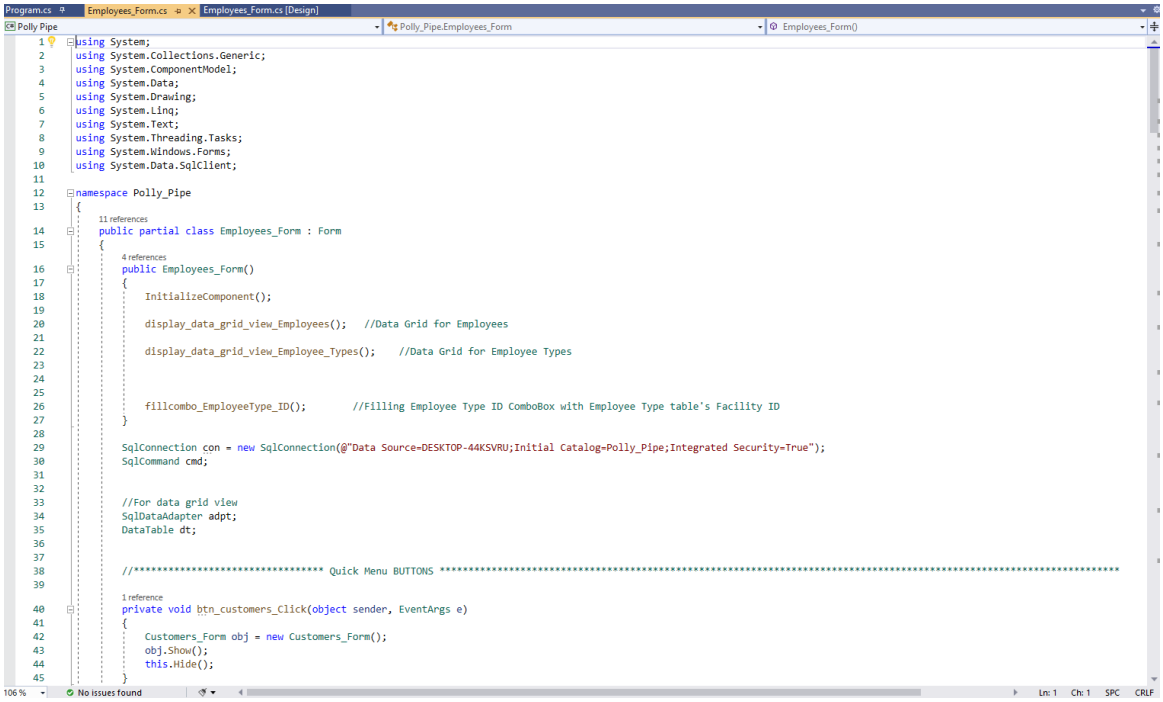


Figure 2. 97 Employee Form (For Admin) interface



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    11 references
    public partial class Employees_Form : Form
    {
        4 references
        public Employees_Form()
        {
            InitializeComponent();

            display_data_grid_view_Employees();   //Data Grid for Employees

            display_data_grid_view_Employee_Types();    //Data Grid for Employee Types


            fillcombo_EmployeeType_ID();        //Filling Employee Type ID ComboBox with Employee Type table's Facility ID
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************

        1 reference
        private void btn_customers_Click(object sender, EventArgs e)
        {
            Customers_Form obj = new Customers_Form();
            obj.Show();
            this.Hide();
        }
```

Figure 2. 98 Employee Form (For Admin) code - part 1

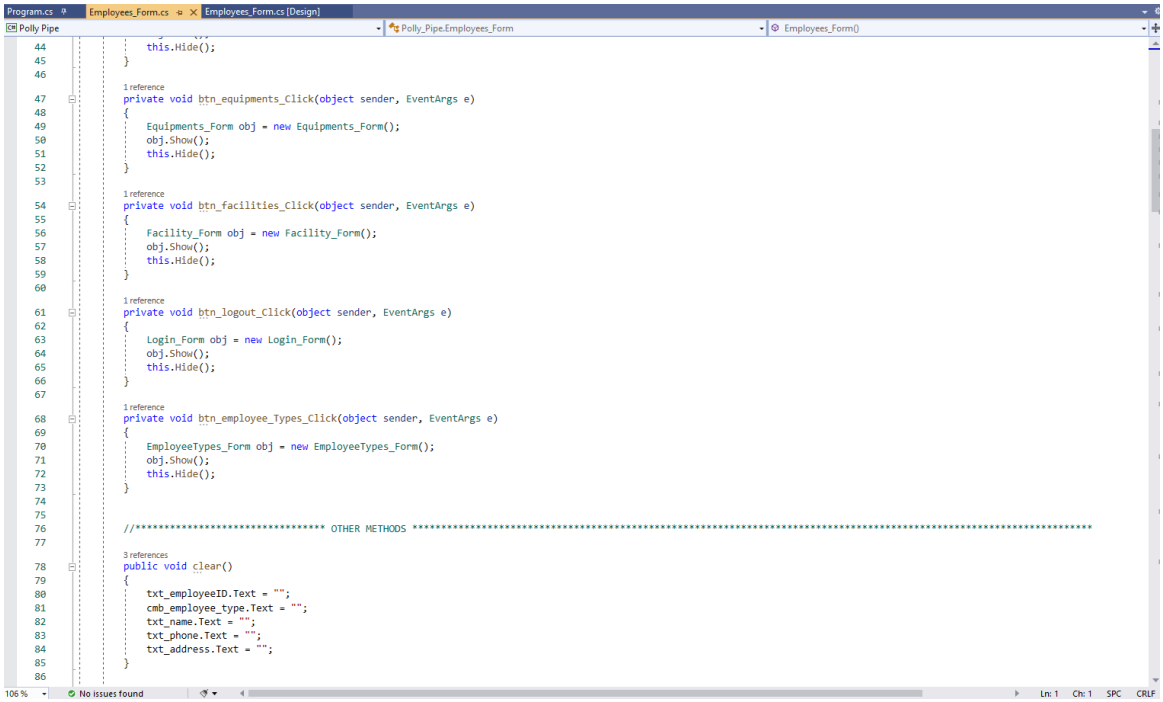


Figure 2. 99 Employee Form (For Admin) code - part 2

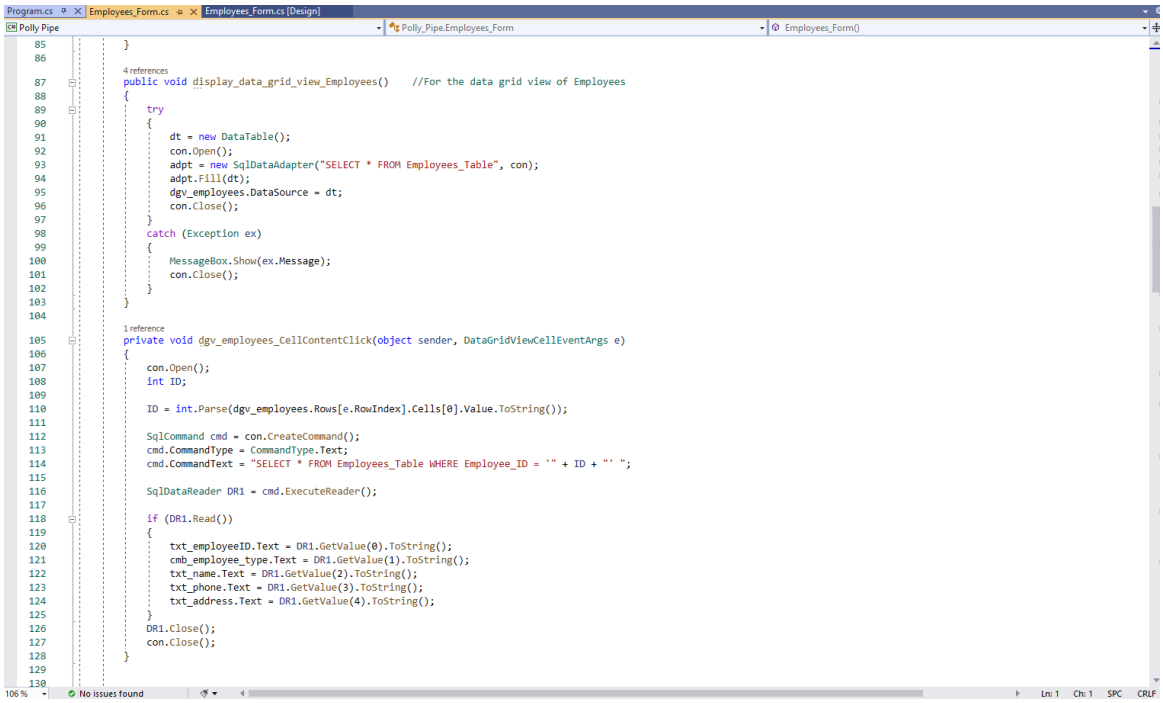
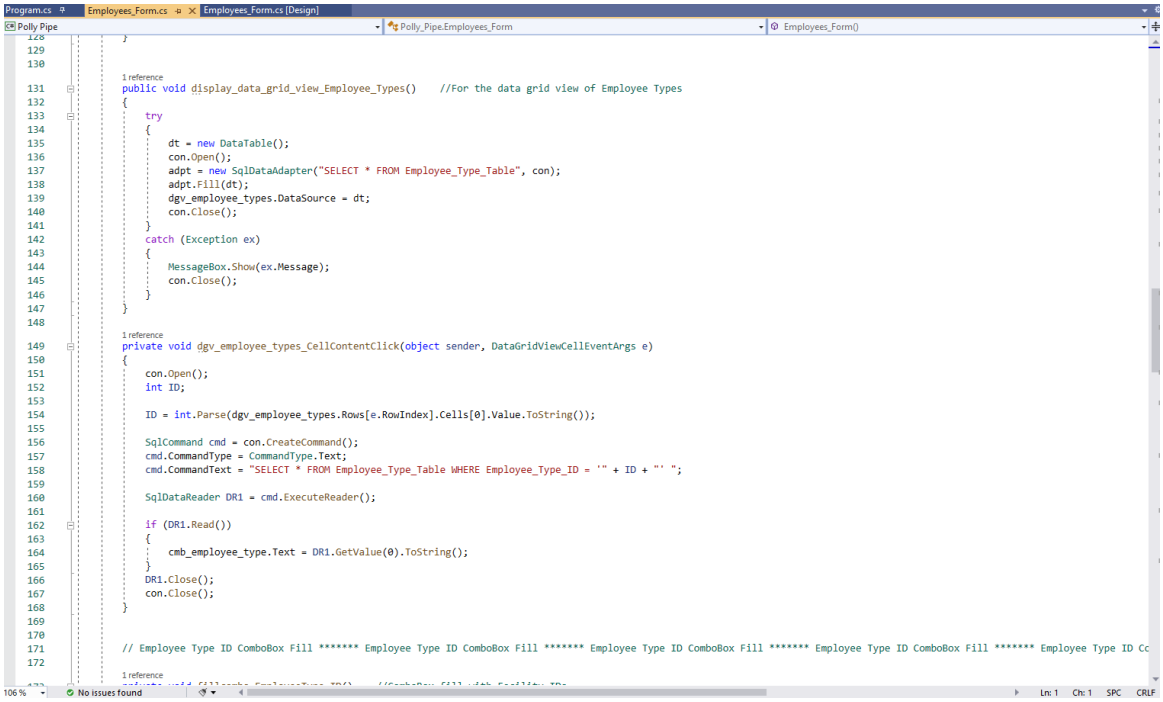
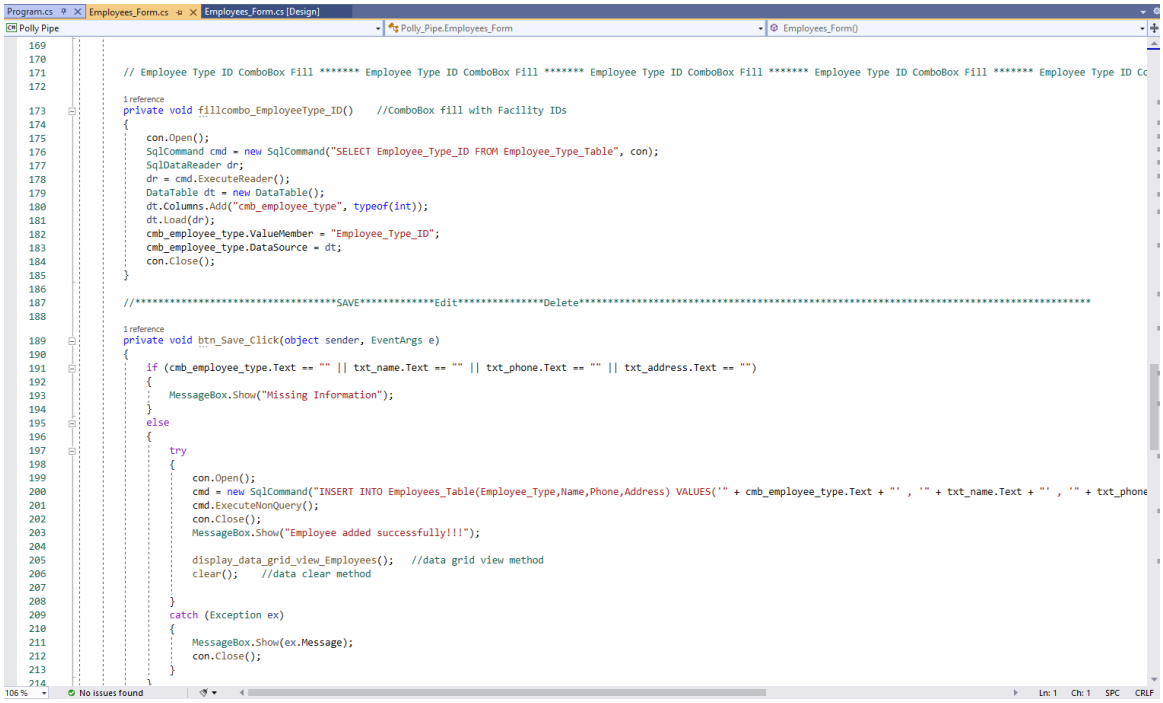


Figure 2. 100 Employee Form (For Admin) code - part 3



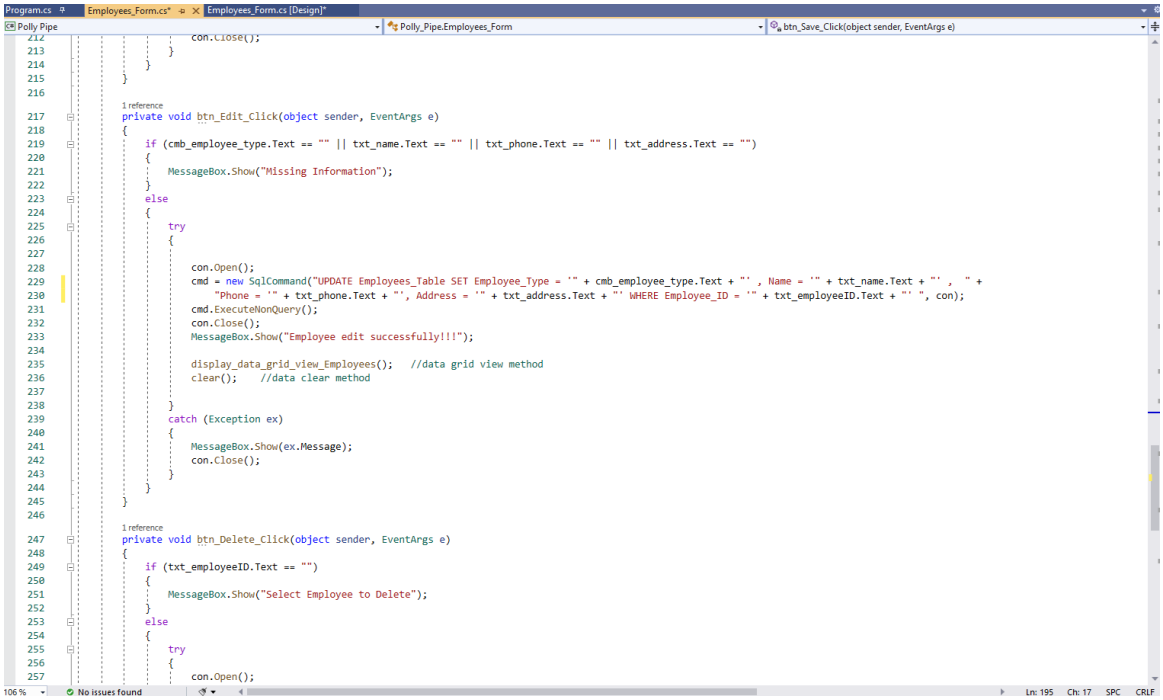
```
        }


        1 reference
        public void display_data_grid_view_Employee_Types()    //For the data grid view of Employee Types
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Employee_Type_Table", con);
                adpt.Fill(dt);
                dgv_employee_types.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_employee_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_employee_types.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Employee_Type_Table WHERE Employee_Type_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                cmb_employee_type.Text = DR1.GetValue(0).ToString();
            }
            DR1.Close();
            con.Close();
        }


        // Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Co

        1 reference
```

Figure 2. 101 Employee Form (For Admin) code - part 4



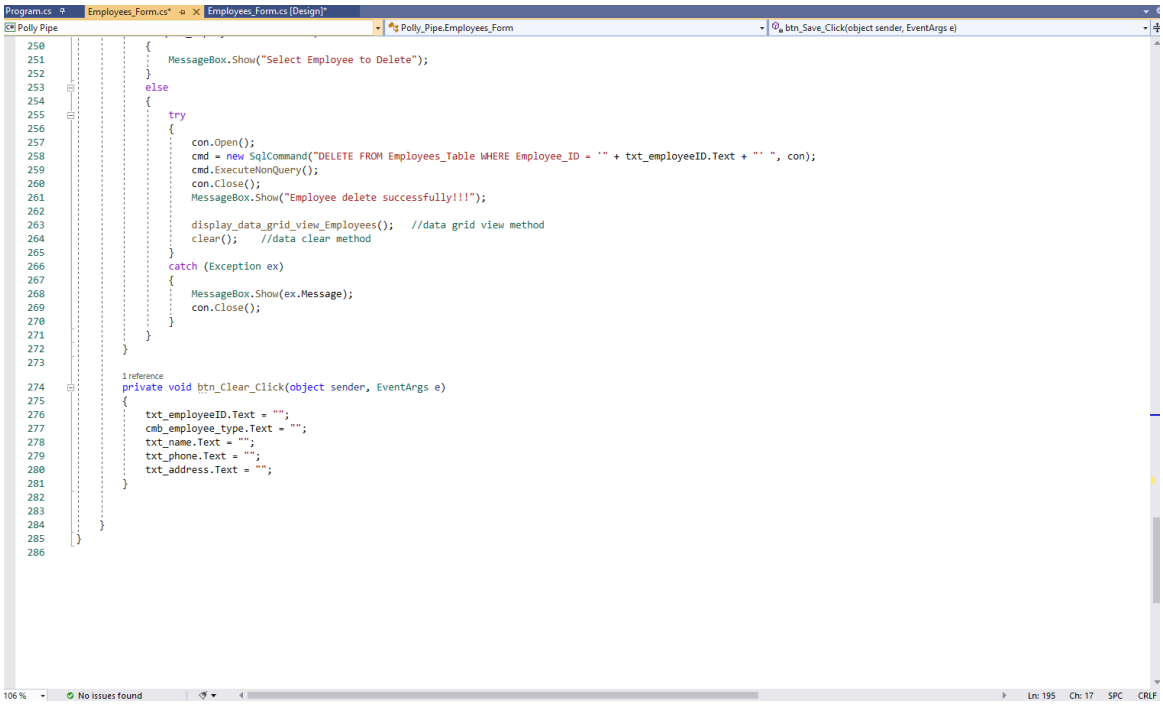
```
        // Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Co

        1 reference
        private void fillcombo_EmployeeType_ID()    //ComboBox fill with Facility IDs
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT Employee_Type_ID FROM Employee_Type_Table", con);
            SqlDataReader dr;
            dr = cmd.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Columns.Add("cmb_employee_type", typeof(int));
            dt.Load(dr);
            cmb_employee_type.ValueMember = "Employee_Type_ID";
            cmb_employee_type.DataSource = dt;
            con.Close();
        }

        //**********************************SAVE***********Edit**********Delete*****************************************************************************************

        1 reference
        private void btn_Save_Click(object sender, EventArgs e)
        {
            if (cmb_employee_type.Text == "" || txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("INSERT INTO Employees_Table(Employee_Type,Name,Phone,Address) VALUES('" + cmb_employee_type.Text + "' , '" + txt_name.Text + "' , '" + txt_phone
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Employee added successfully!!!");

                    display_data_grid_view_Employees();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
```

Figure 2. 102 Employee Form (For Admin) code - part 6



```
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Edit_Click(object sender, EventArgs e)
        {
            if (cmb_employee_type.Text == "" || txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {

                    con.Open();
                    cmd = new SqlCommand("UPDATE Employees_Table SET Employee_Type = '" + cmb_employee_type.Text + "' , Name = '" + txt_name.Text + "' ,  " +
                        "Phone = '" + txt_phone.Text + "', Address = '" + txt_address.Text + "' WHERE Employee_ID = '" + txt_employeeID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Employee edit successfully!!!");

                    display_data_grid_view_Employees();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (txt_employeeID.Text == "")
            {
                MessageBox.Show("Select Employee to Delete");
            }
            else
            {
                try
                {
                    con.Open();
```

Figure 2. 103 Employee Form (For Admin) code - part 7

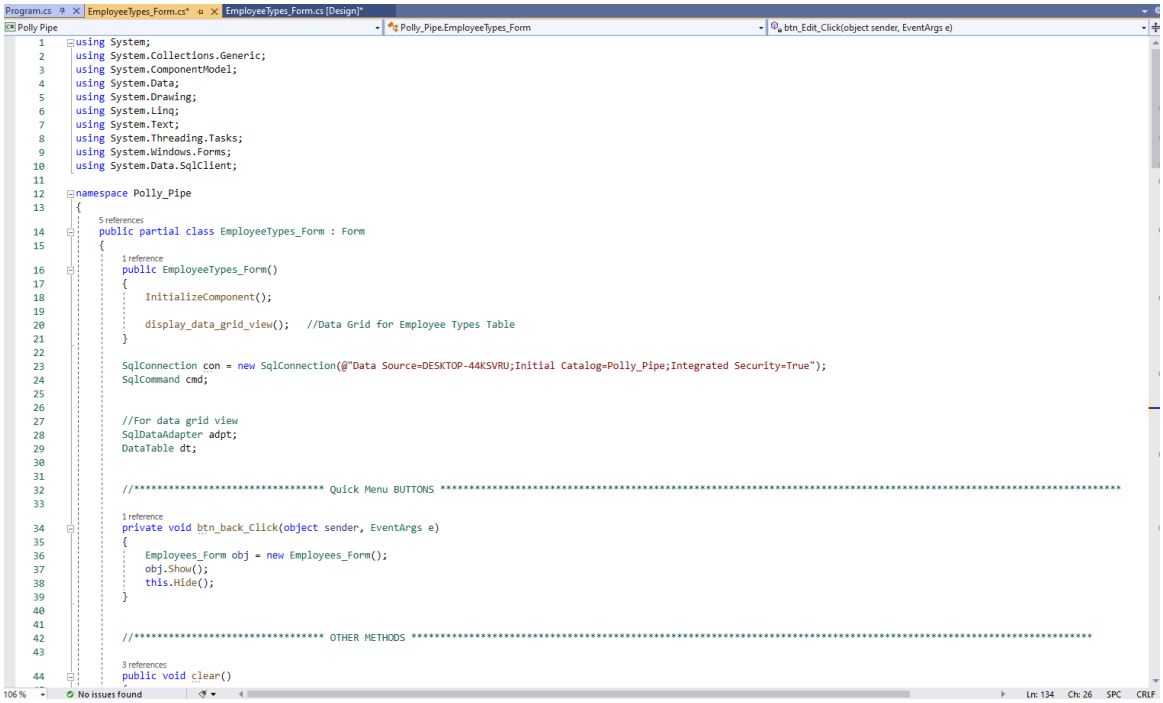


```
            {
                MessageBox.Show("Select Employee to Delete");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("DELETE FROM Employees_Table WHERE Employee_ID = '" + txt_employeeID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Employee delete successfully!!!");

                    display_data_grid_view_Employees();   //data grid view method
                    clear();    //data clear method
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Clear_Click(object sender, EventArgs e)
        {
            txt_employeeID.Text = "";
            cmb_employee_type.Text = "";
            txt_name.Text = "";
            txt_phone.Text = "";
            txt_address.Text = "";
        }


    }
}
```

Figure 2. 104 Employee Form (For Admin) code - part 8

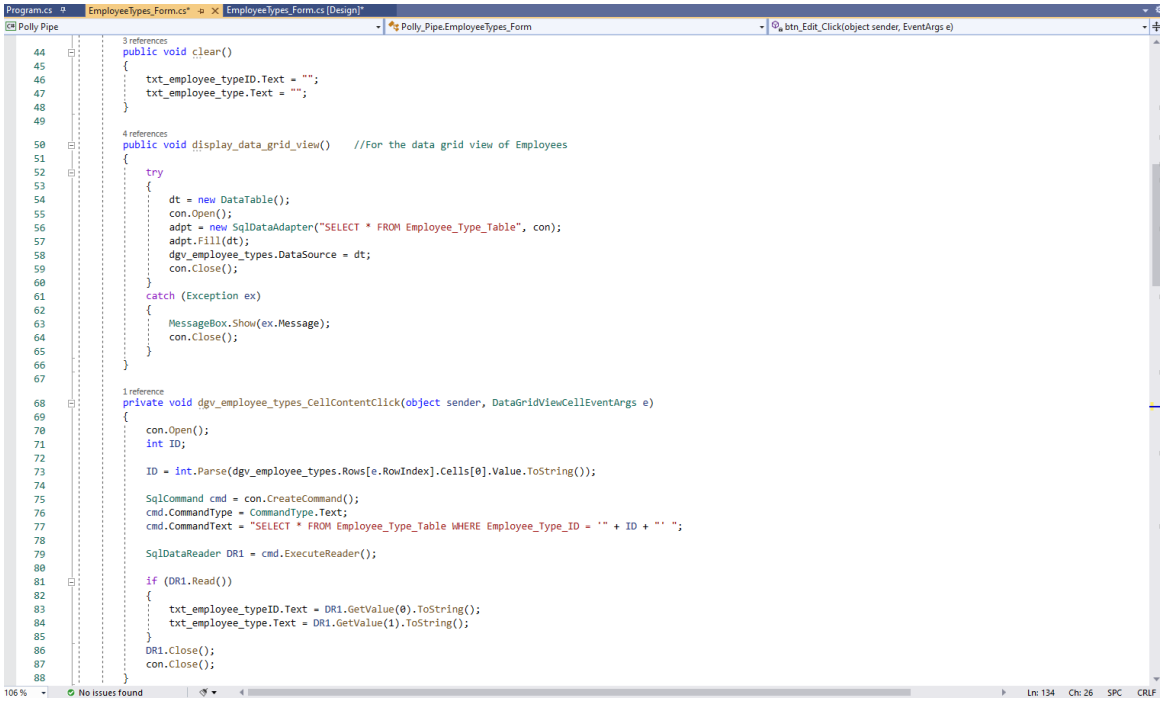
Employee Type Form (For Admin) →

Figure 2. 105Employee Type Form (For Admin) interface



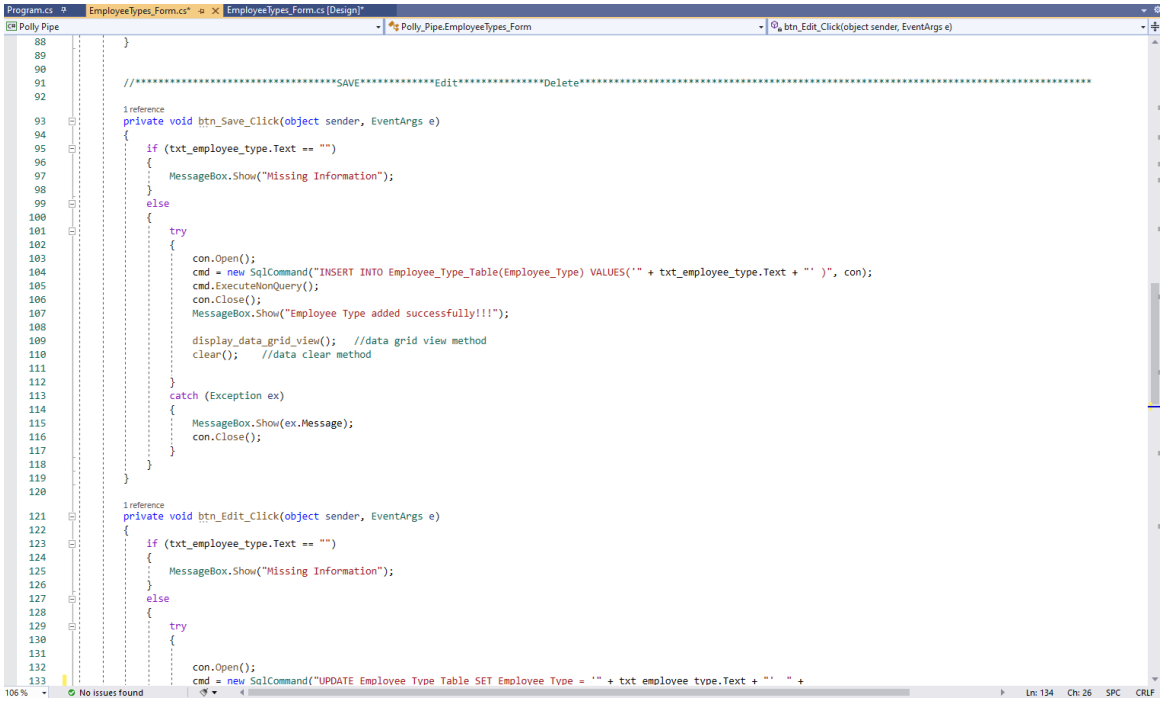
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    5 references
    public partial class EmployeeTypes_Form : Form
    {
        1 reference
        public EmployeeTypes_Form()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Employee Types Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***********************************************************************************************************

        1 reference
        private void btn_back_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
            this.Hide();
        }


        //********************************* OTHER METHODS ****************************************************************************************************************

        3 references
        public void clear()
```

Figure 2. 106 Employee Type Form (For Admin) code - part 1



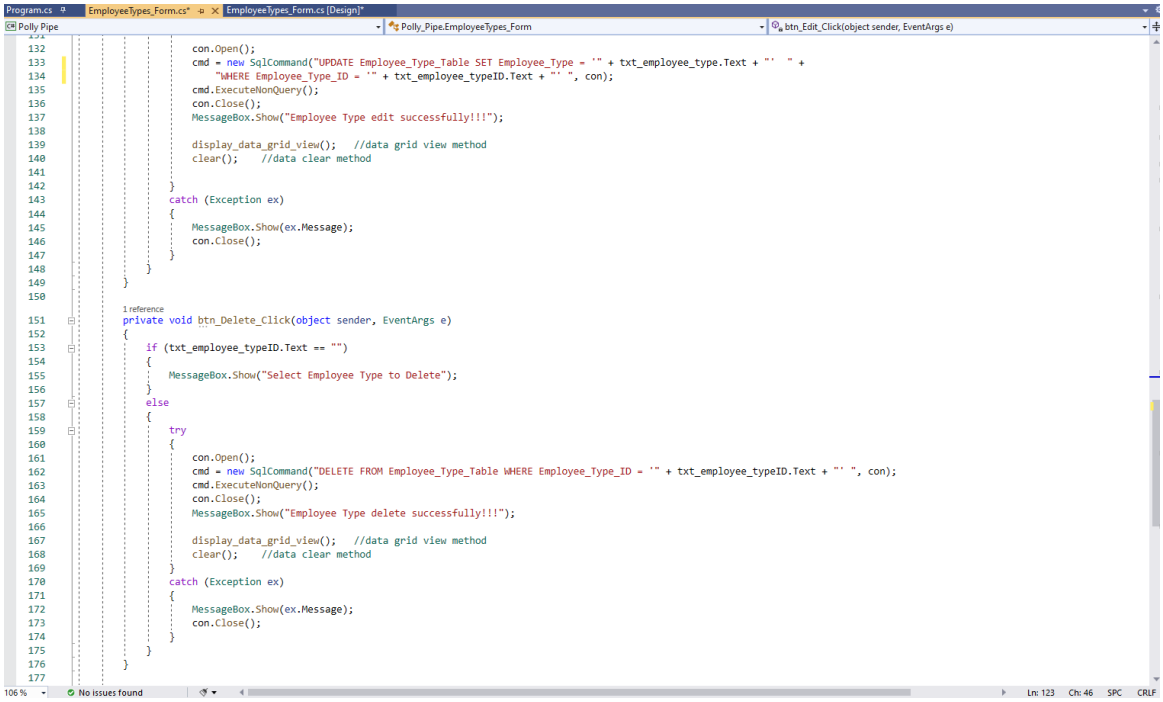
```
3 references
public void clear()
{
    txt_employee_typeID.Text = "";
    txt_employee_type.Text = "";
}

4 references
public void display_data_grid_view()    //For the data grid view of Employees
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Employee_Type_Table", con);
        adpt.Fill(dt);
        dgv_employee_types.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void dgv_employee_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_employee_types.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Employee_Type_Table WHERE Employee_Type_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_employee_typeID.Text = DR1.GetValue(0).ToString();
        txt_employee_type.Text = DR1.GetValue(1).ToString();
    }
    DR1.Close();
    con.Close();
}
```

Figure 2. 107 Employee Type Form (For Admin) code - part 2



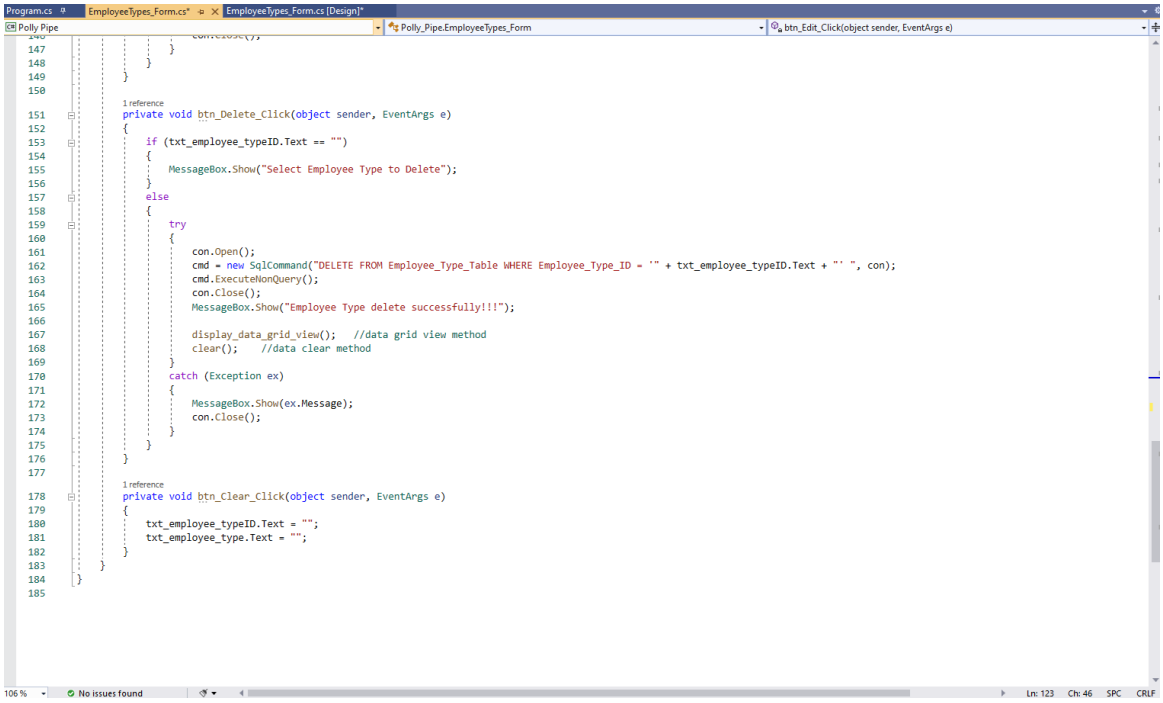
```
}


//*********************************SAVE***********Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_employee_type.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Employee_Type_Table(Employee_Type) VALUES('" + txt_employee_type.Text + "' )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Employee Type added successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_employee_type.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
            cmd = new SqlCommand("UPDATE Employee Type Table SET Employee Type = '" + txt employee type.Text + "'  " +
```

Figure 2. 108 Employee Type Form (For Admin) code - part 3



```
            con.Open();
            cmd = new SqlCommand("UPDATE Employee_Type_Table SET Employee_Type = '" + txt_employee_type.Text + "'  " +
                "WHERE Employee_Type_ID = '" + txt_employee_typeID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Employee Type edit successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_employee_typeID.Text == "")
    {
        MessageBox.Show("Select Employee Type to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Employee_Type_Table WHERE Employee_Type_ID = '" + txt_employee_typeID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Employee Type delete successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}
```

Figure 2. 109 Employee Type Form (For Admin) code - part 4



```
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_employee_typeID.Text == "")
    {
        MessageBox.Show("Select Employee Type to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Employee_Type_Table WHERE Employee_Type_ID = '" + txt_employee_typeID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Employee Type delete successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
    txt_employee_typeID.Text = "";
    txt_employee_type.Text = "";
}
}
}
```

Figure 2. 110 Employee Type Form (For Admin) code - part 5

Equipment Form (For Admin) →

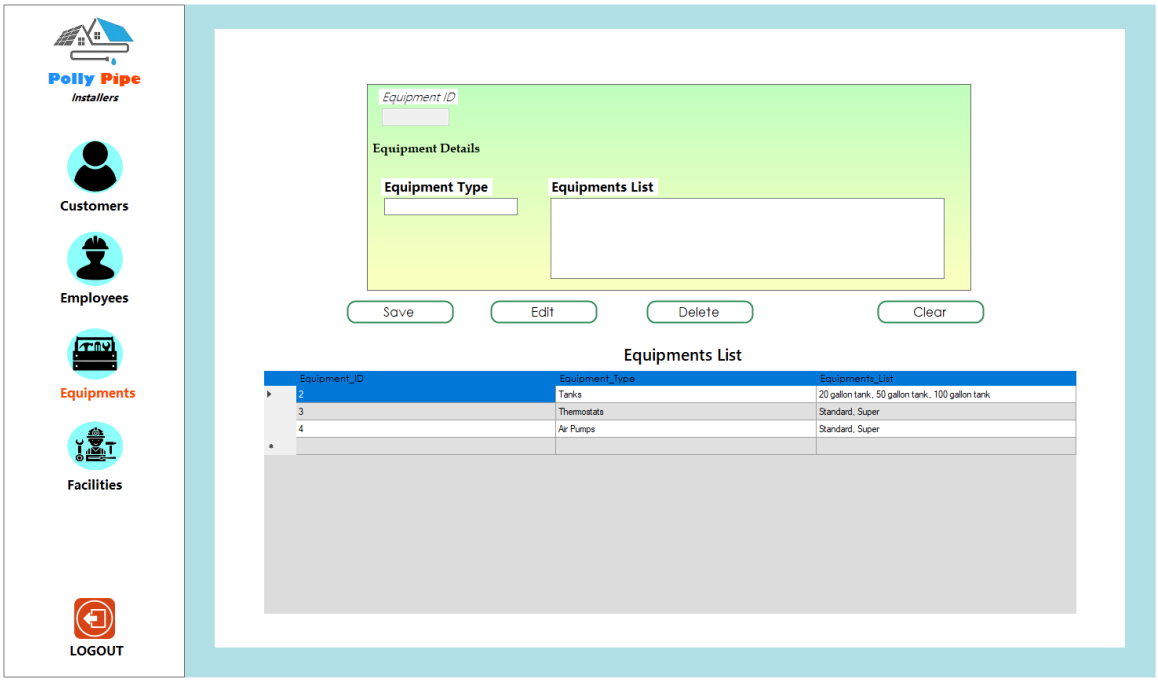
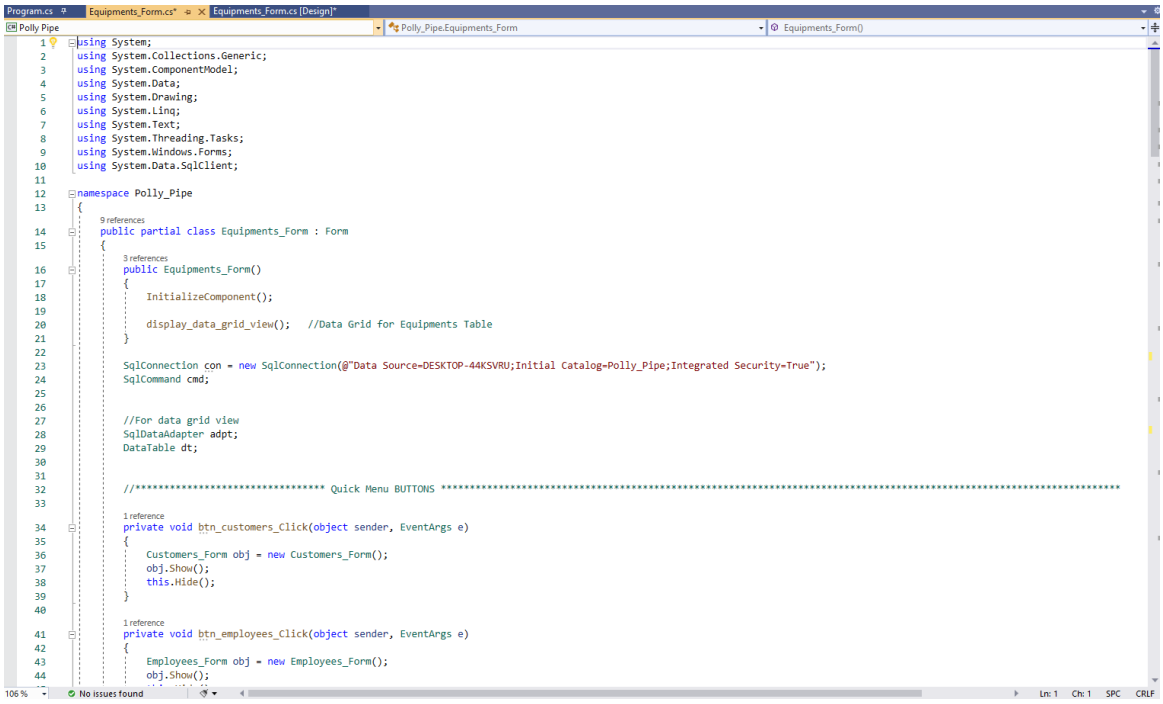
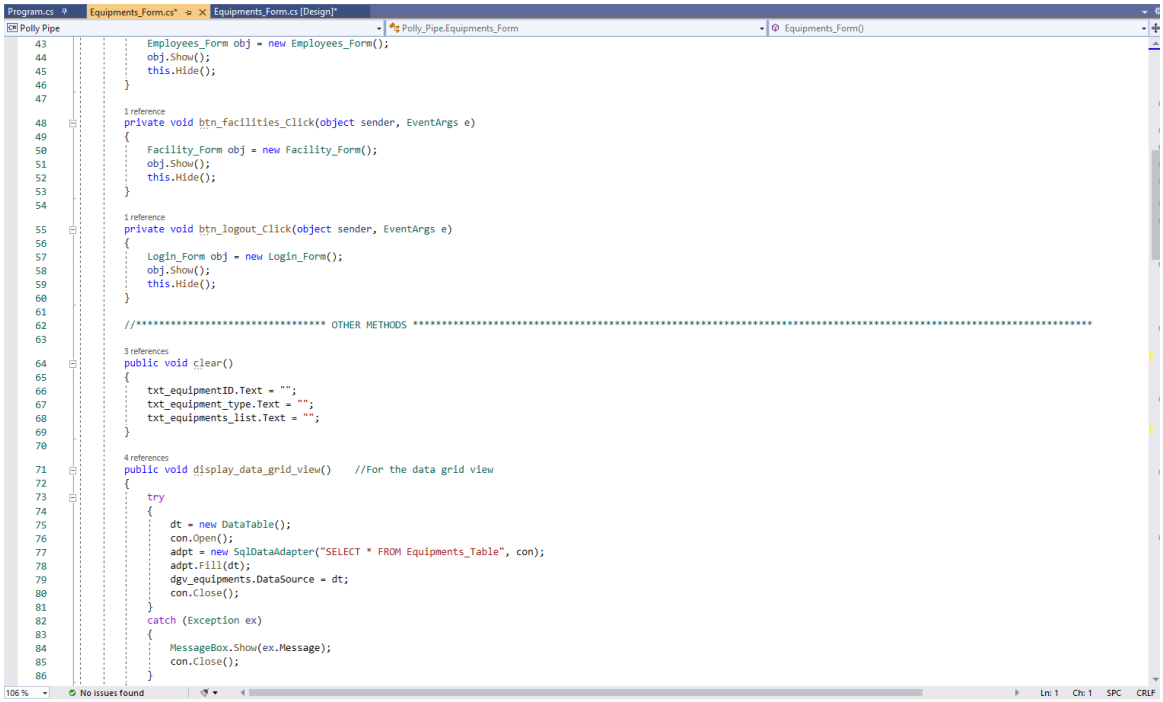


Figure 2. 111 Equipment Form (For Admin) interface



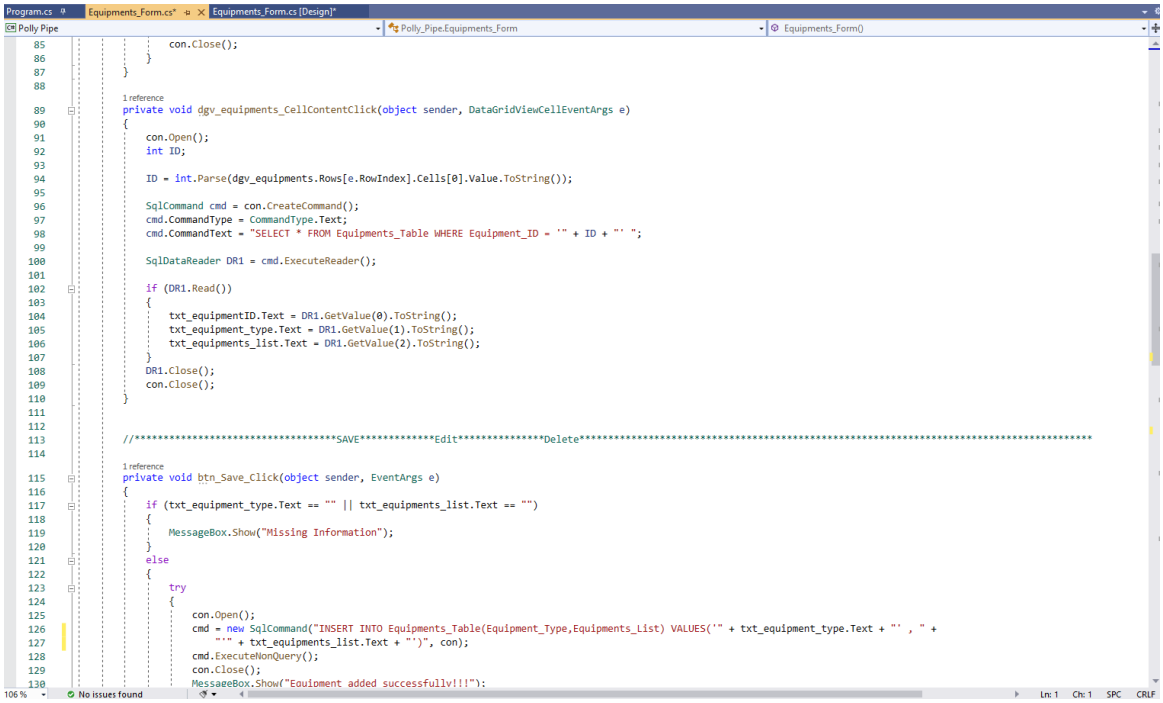
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    public partial class Equipments_Form : Form
    {
        public Equipments_Form()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Equipments Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************

        private void btn_customers_Click(object sender, EventArgs e)
        {
            Customers_Form obj = new Customers_Form();
            obj.Show();
            this.Hide();
        }

        private void btn_employees_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
```

Figure 2. 112 Equipment Form (For Admin) code - part 1



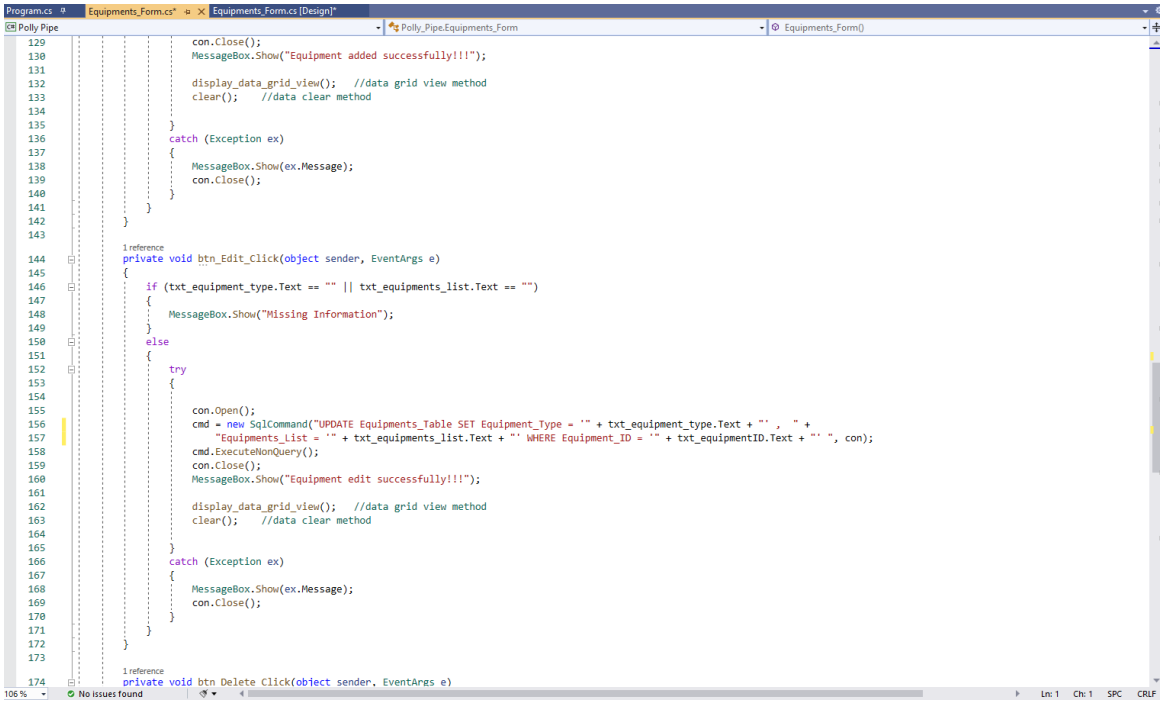
```
Employees_Form obj = new Employees_Form();
obj.Show();
this.Hide();
}

1 reference
private void btn_facilities_Click(object sender, EventArgs e)
{
Facility_Form obj = new Facility_Form();
obj.Show();
this.Hide();
}

1 reference
private void btn_logout_Click(object sender, EventArgs e)
{
Login_Form obj = new Login_Form();
obj.Show();
this.Hide();
}

//********************************* OTHER METHODS ***********************************************************************************************************

3 references
public void clear()
{
txt_equipmentID.Text = "";
txt_equipment_type.Text = "";
txt_equipments_list.Text = "";
}

4 references
public void display_data_grid_view()    //For the data grid view
{
try
{
dt = new DataTable();
con.Open();
adpt = new SqlDataAdapter("SELECT * FROM Equipments_Table", con);
adpt.Fill(dt);
dgv_equipments.DataSource = dt;
con.Close();
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
con.Close();
}
```

Figure 2. 113 Equipment Form (For Admin) code - part 2



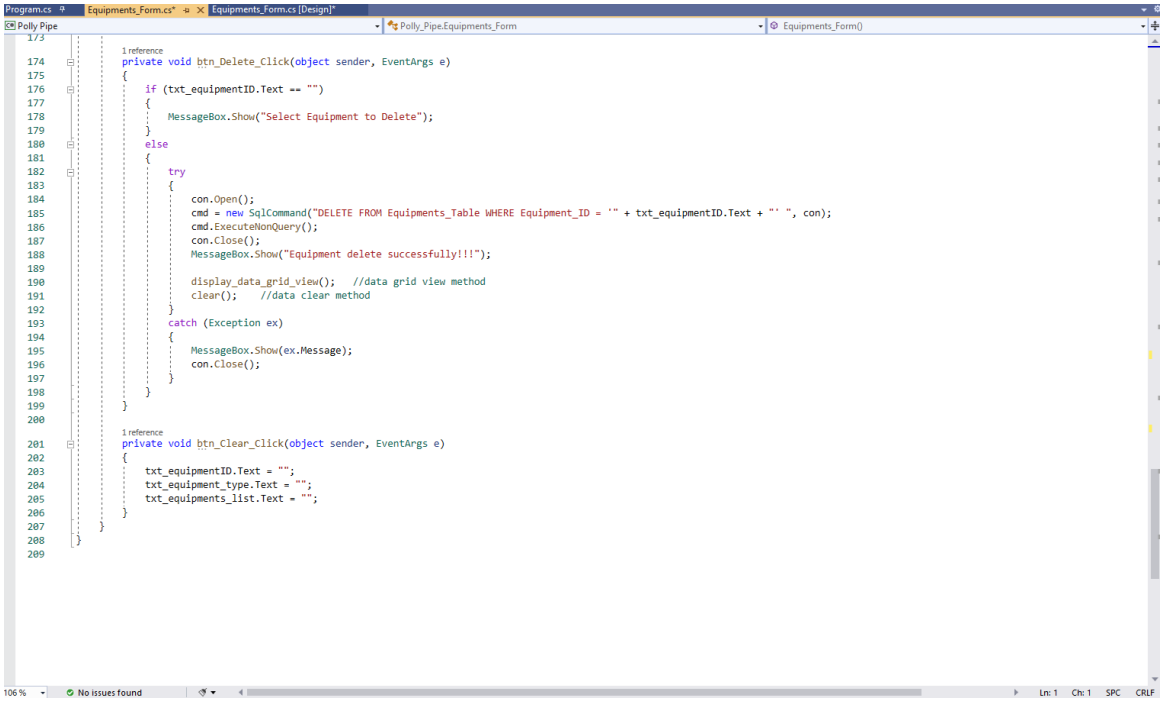
```
con.Close();
}
}

1 reference
private void dgv_equipments_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
con.Open();
int ID;

ID = int.Parse(dgv_equipments.Rows[e.RowIndex].Cells[0].Value.ToString());

SqlCommand cmd = con.CreateCommand();
cmd.CommandType = CommandType.Text;
cmd.CommandText = "SELECT * FROM Equipments_Table WHERE Equipment_ID = '" + ID + "' ";

SqlDataReader DR1 = cmd.ExecuteReader();

if (DR1.Read())
{
txt_equipmentID.Text = DR1.GetValue(0).ToString();
txt_equipment_type.Text = DR1.GetValue(1).ToString();
txt_equipments_list.Text = DR1.GetValue(2).ToString();
}
DR1.Close();
con.Close();
}


//*********************************SAVE************Edit**************Delete*****************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
if (txt_equipment_type.Text == "" || txt_equipments_list.Text == "")
{
MessageBox.Show("Missing Information");
}
else
{
try
{
con.Open();
cmd = new SqlCommand("INSERT INTO Equipments_Table(Equipment_Type,Equipments_List) VALUES('" + txt_equipment_type.Text + "' , " +
"'" + txt_equipments_list.Text + "')", con);
cmd.ExecuteNonQuery();
con.Close();
MessageBox.Show("Equipment added successfully!!!");
```

Figure 2. 114 Equipment Form (For Admin) code - part 3



```
                con.Close();
                MessageBox.Show("Equipment added successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Edit_Click(object sender, EventArgs e)
    {
        if (txt_equipment_type.Text == "" || txt_equipments_list.Text == "")
        {
            MessageBox.Show("Missing Information");
        }
        else
        {
            try
            {

                con.Open();
                cmd = new SqlCommand("UPDATE Equipments_Table SET Equipment_Type = '" + txt_equipment_type.Text + "' ,  " +
                    "Equipments_List = '" + txt_equipments_list.Text + "' WHERE Equipment_ID = '" + txt_equipmentID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Equipment edit successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Delete_Click(object sender, EventArgs e)
```

Figure 2. 115 Equipment Form (For Admin) code - part 4



```
    1 reference
    private void btn_Delete_Click(object sender, EventArgs e)
    {
        if (txt_equipmentID.Text == "")
        {
            MessageBox.Show("Select Equipment to Delete");
        }
        else
        {
            try
            {
                con.Open();
                cmd = new SqlCommand("DELETE FROM Equipments_Table WHERE Equipment_ID = '" + txt_equipmentID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Equipment delete successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Clear_Click(object sender, EventArgs e)
    {
        txt_equipmentID.Text = "";
        txt_equipment_type.Text = "";
        txt_equipments_list.Text = "";
    }
}
}
```

Figure 2. 116 Equipment Form (For Admin) code - part 5

Facility Form (For Admin) →

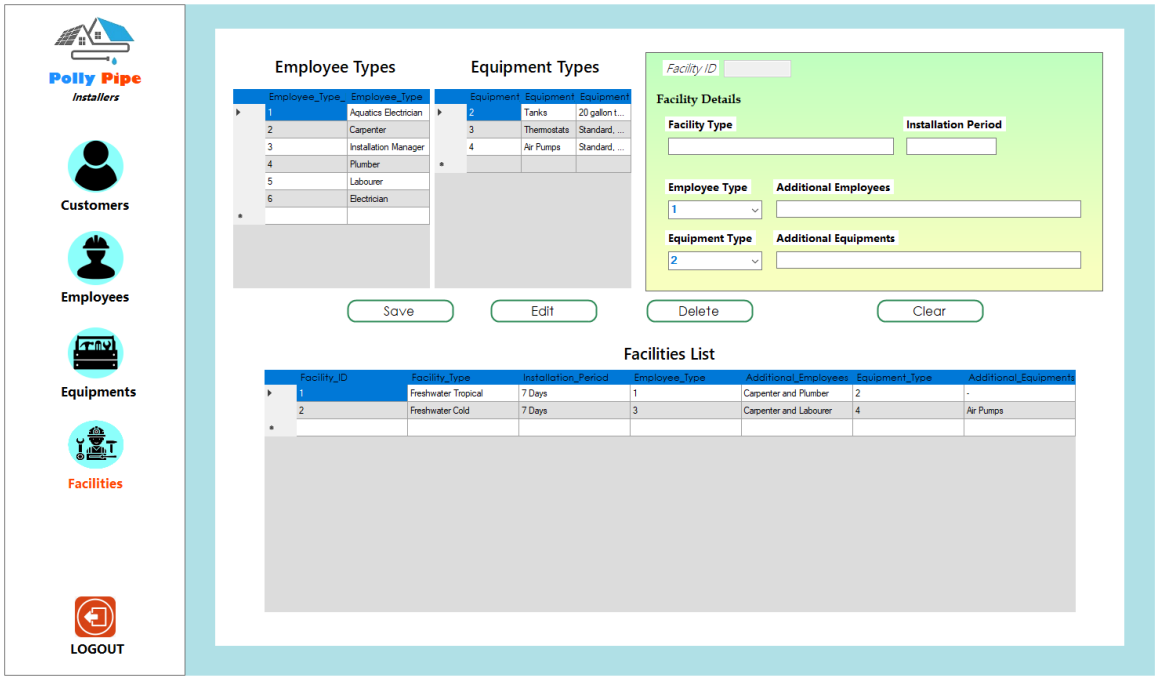
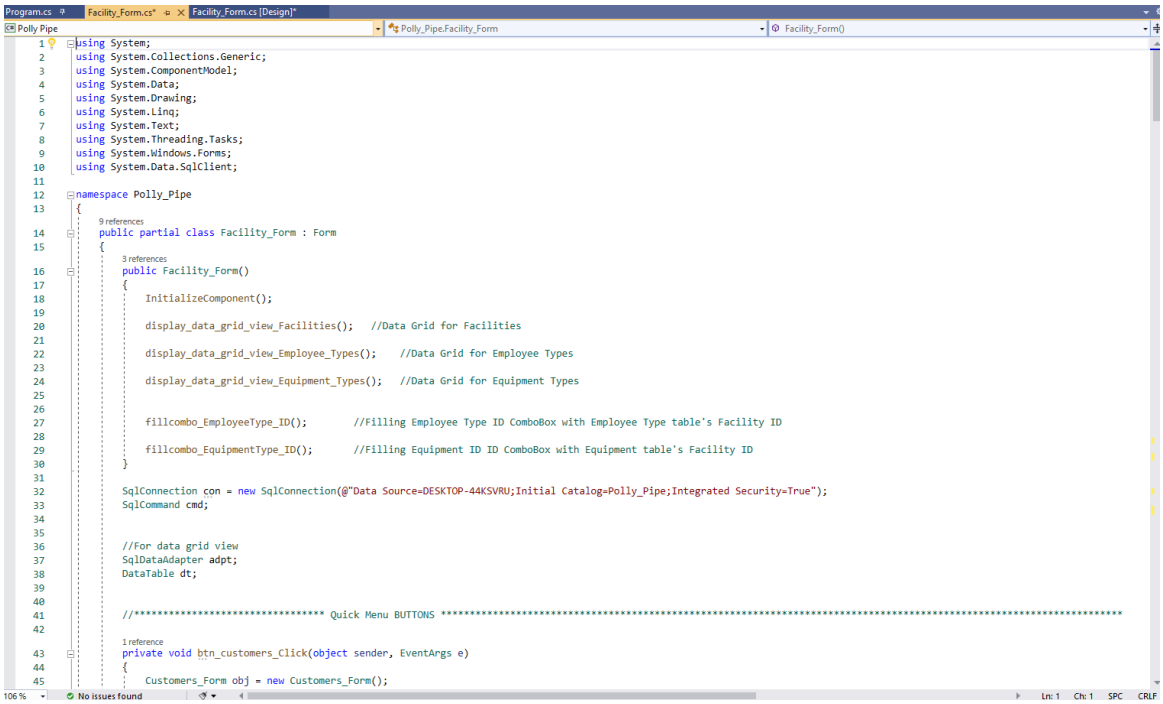
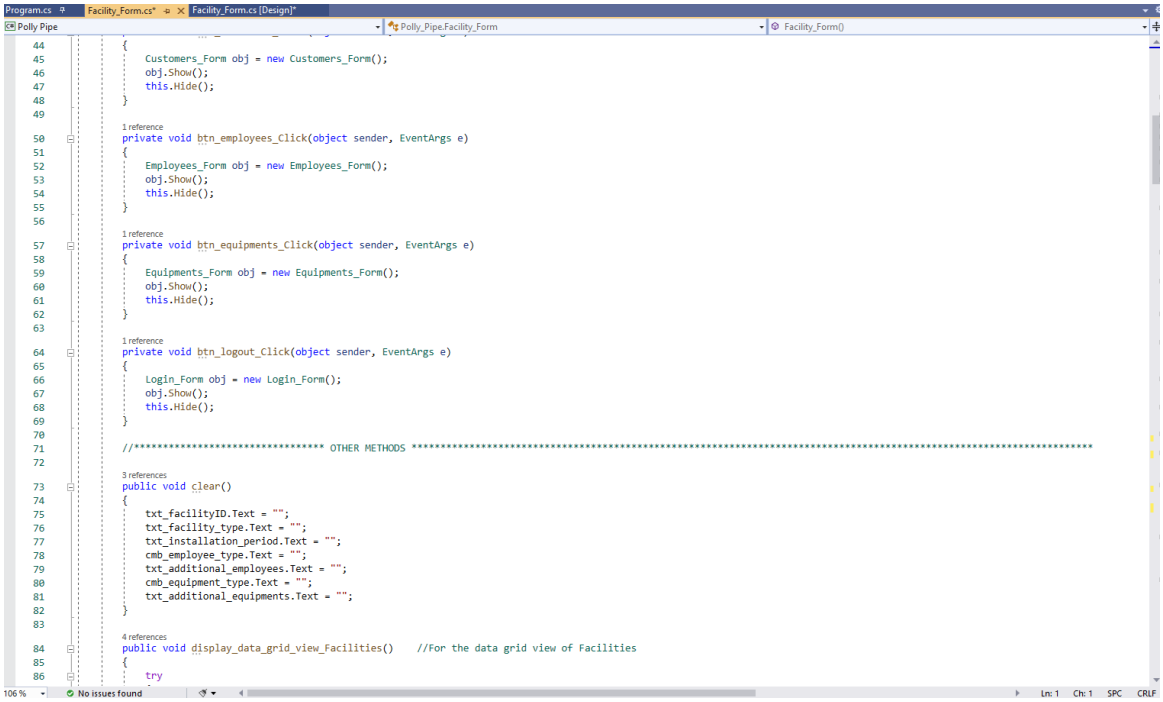


Figure 2. 117 Facility Form (For Admin) interface



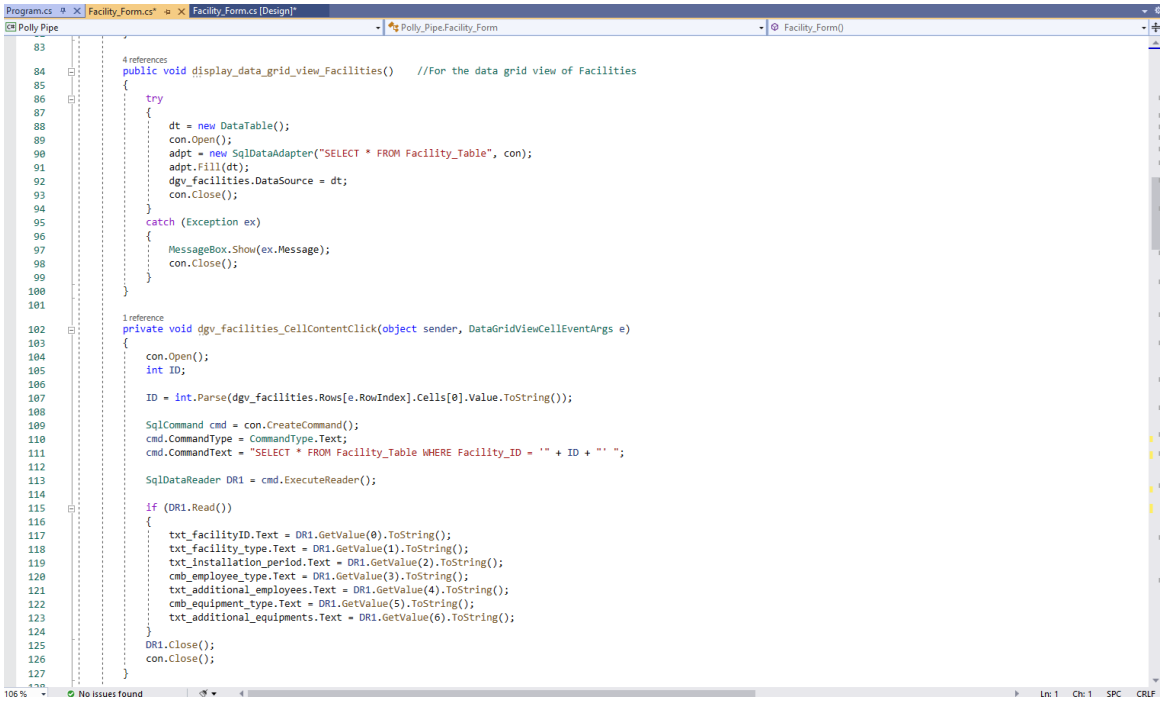
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    public partial class Facility_Form : Form
    {
        public Facility_Form()
        {
            InitializeComponent();

            display_data_grid_view_Facilities();   //Data Grid for Facilities

            display_data_grid_view_Employee_Types();    //Data Grid for Employee Types

            display_data_grid_view_Equipment_Types();   //Data Grid for Equipment Types


            fillcombo_EmployeeType_ID();        //Filling Employee Type ID ComboBox with Employee Type table's Facility ID

            fillcombo_EquipmentType_ID();       //Filling Equipment ID ID ComboBox with Equipment table's Facility ID
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***************************************************************************************************************

        private void btn_customers_Click(object sender, EventArgs e)
        {
            Customers_Form obj = new Customers_Form();
```

Figure 2. 118Facility Form (For Admin) code - part 1



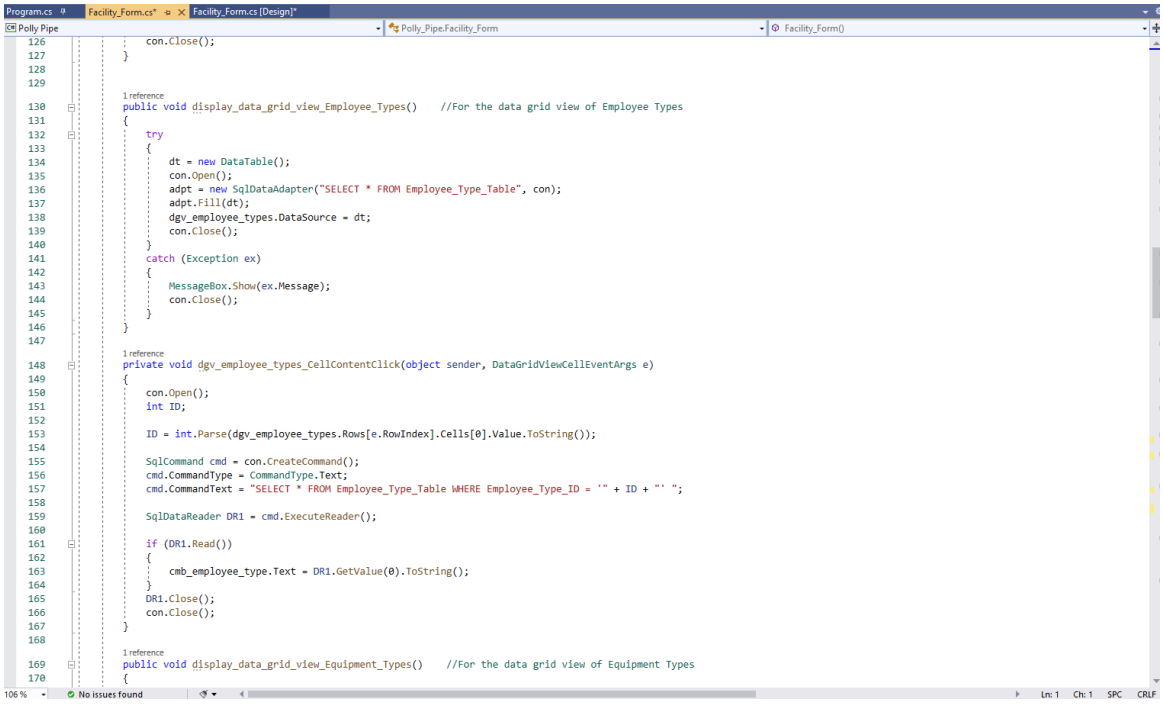
```
        {
            Customers_Form obj = new Customers_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_employees_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_equipments_Click(object sender, EventArgs e)
        {
            Equipments_Form obj = new Equipments_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        //********************************* OTHER METHODS ***********************************************************************************************************

        3 references
        public void clear()
        {
            txt_facilityID.Text = "";
            txt_facility_type.Text = "";
            txt_installation_period.Text = "";
            cmb_employee_type.Text = "";
            txt_additional_employees.Text = "";
            cmb_equipment_type.Text = "";
            txt_additional_equipments.Text = "";
        }

        4 references
        public void display_data_grid_view_Facilities()    //For the data grid view of Facilities
        {
            try
```

Figure 2. 119 Facility Form (For Admin) code - part 2



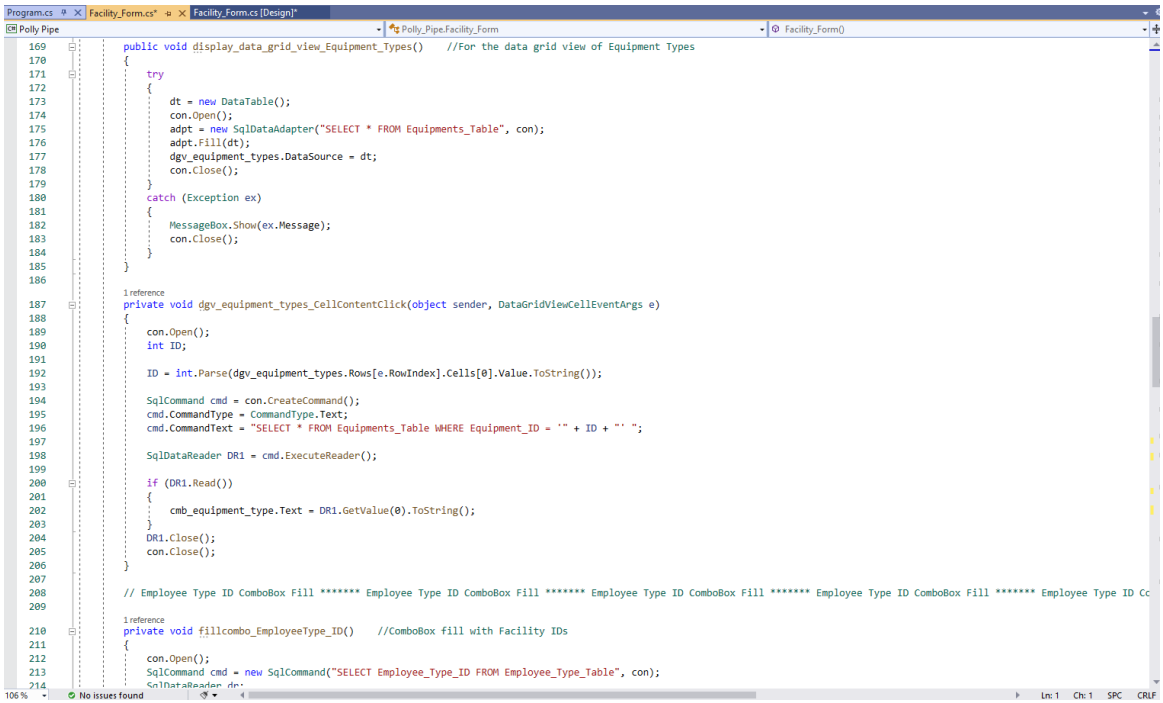
```
        4 references
        public void display_data_grid_view_Facilities()    //For the data grid view of Facilities
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Facility_Table", con);
                adpt.Fill(dt);
                dgv_facilities.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_facilities_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_facilities.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Facility_Table WHERE Facility_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                txt_facilityID.Text = DR1.GetValue(0).ToString();
                txt_facility_type.Text = DR1.GetValue(1).ToString();
                txt_installation_period.Text = DR1.GetValue(2).ToString();
                cmb_employee_type.Text = DR1.GetValue(3).ToString();
                txt_additional_employees.Text = DR1.GetValue(4).ToString();
                cmb_equipment_type.Text = DR1.GetValue(5).ToString();
                txt_additional_equipments.Text = DR1.GetValue(6).ToString();
            }
            DR1.Close();
            con.Close();
        }
```

Figure 2. 120 Facility Form (For Admin) code - part 3



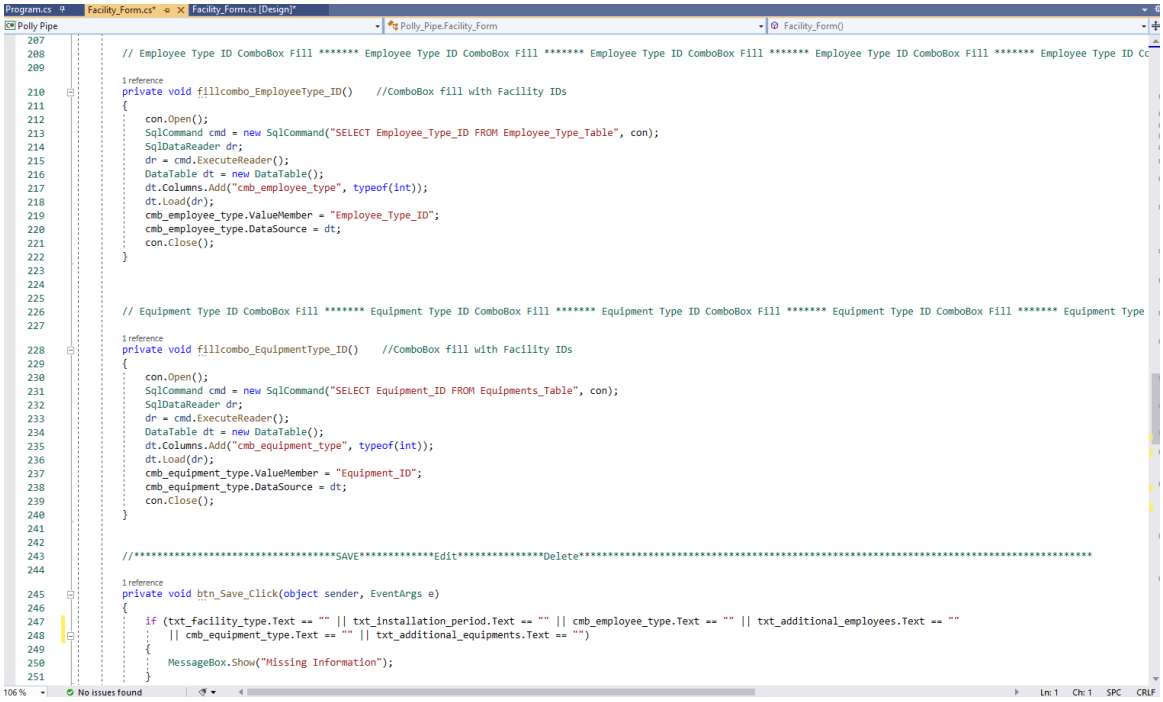
```
            con.Close();
        }


        1 reference
        public void display_data_grid_view_Employee_Types()    //For the data grid view of Employee Types
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Employee_Type_Table", con);
                adpt.Fill(dt);
                dgv_employee_types.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_employee_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_employee_types.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Employee_Type_Table WHERE Employee_Type_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                cmb_employee_type.Text = DR1.GetValue(0).ToString();
            }
            DR1.Close();
            con.Close();
        }

        1 reference
        public void display_data_grid_view_Equipment_Types()    //For the data grid view of Equipment Types
        {
```

Figure 2. 121 Facility Form (For Admin) code - part 4



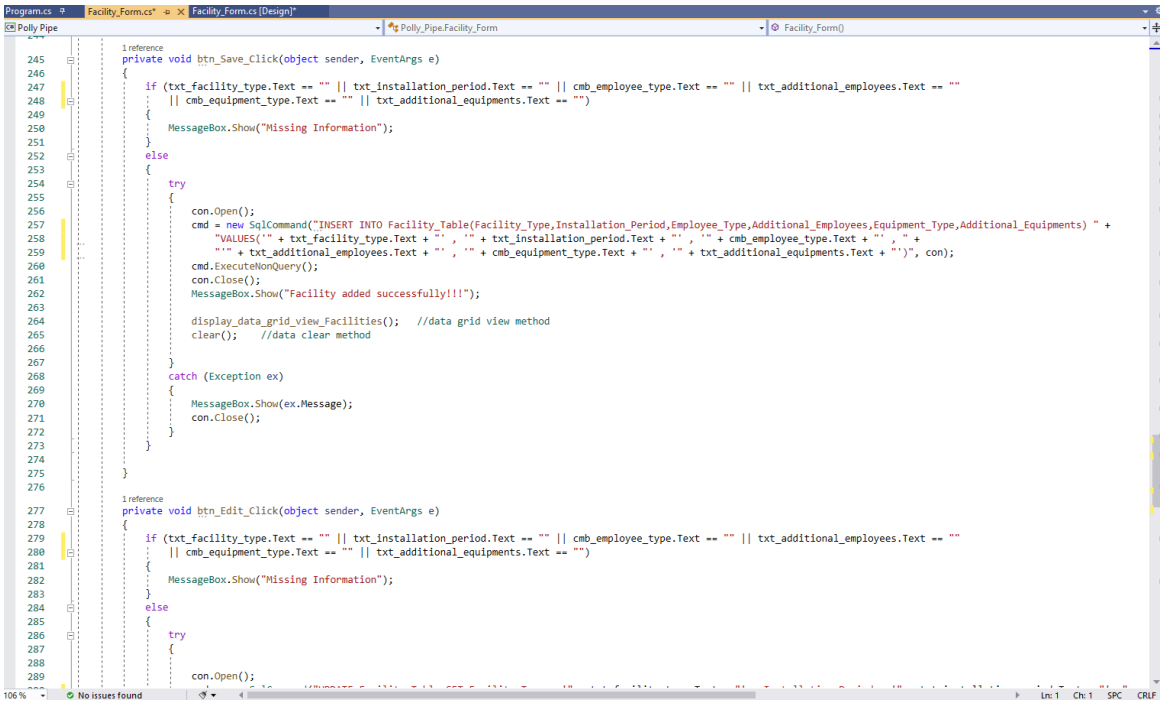
```
        public void display_data_grid_view_Equipment_Types()    //For the data grid view of Equipment Types
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Equipments_Table", con);
                adpt.Fill(dt);
                dgv_equipment_types.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_equipment_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_equipment_types.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Equipments_Table WHERE Equipment_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                cmb_equipment_type.Text = DR1.GetValue(0).ToString();
            }
            DR1.Close();
            con.Close();
        }

        // Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Cc

        1 reference
        private void fillcombo_EmployeeType_ID()    //ComboBox fill with Facility IDs
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT Employee_Type_ID FROM Employee_Type_Table", con);
```

Figure 2. 122 Facility Form (For Admin) code - part 5



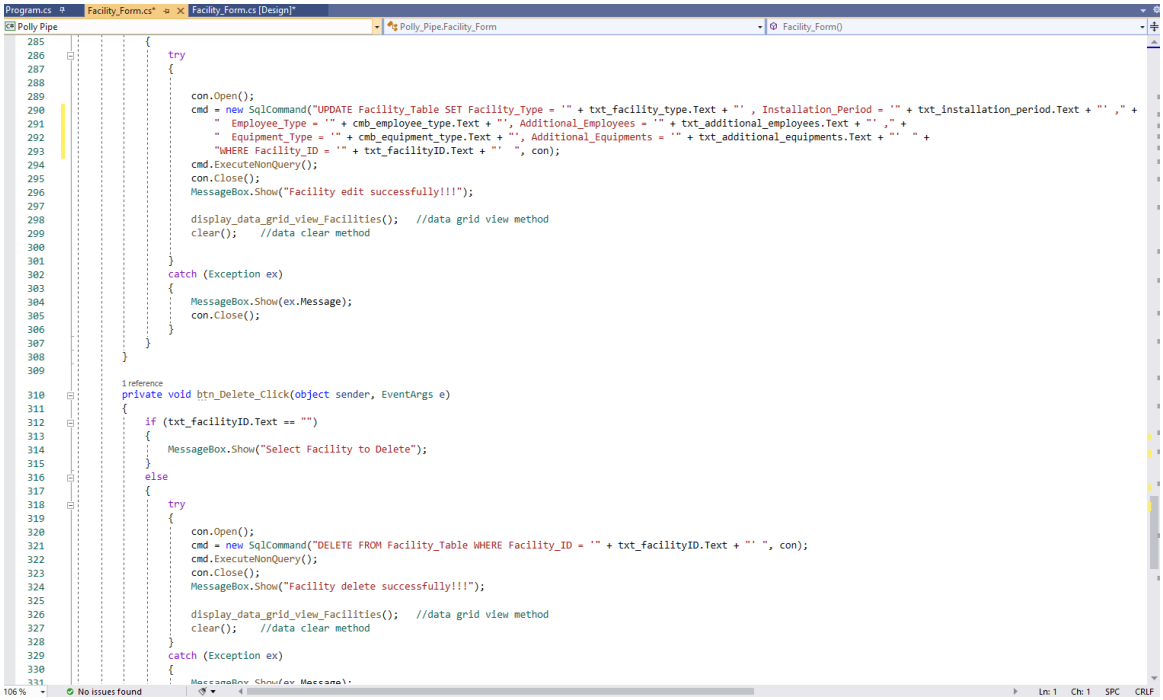
```
// Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Co

1 reference
private void fillcombo_EmployeeType_ID()    //ComboBox fill with Facility IDs
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Employee_Type_ID FROM Employee_Type_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_employee_type", typeof(int));
    dt.Load(dr);
    cmb_employee_type.ValueMember = "Employee_Type_ID";
    cmb_employee_type.DataSource = dt;
    con.Close();
}



// Equipment Type ID ComboBox Fill ******* Equipment Type ID ComboBox Fill ******* Equipment Type ID ComboBox Fill ******* Equipment Type ID ComboBox Fill ******* Equipment Type

1 reference
private void fillcombo_EquipmentType_ID()    //ComboBox fill with Facility IDs
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Equipment_ID FROM Equipments_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_equipment_type", typeof(int));
    dt.Load(dr);
    cmb_equipment_type.ValueMember = "Equipment_ID";
    cmb_equipment_type.DataSource = dt;
    con.Close();
}


//*********************************SAVE***********Edit**********Delete**********

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_facility_type.Text == "" || txt_installation_period.Text == "" || cmb_employee_type.Text == "" || txt_additional_employees.Text == ""
        || cmb_equipment_type.Text == "" || txt_additional_equipments.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
```

Figure 2. 123 Facility Form (For Admin) code - part 6



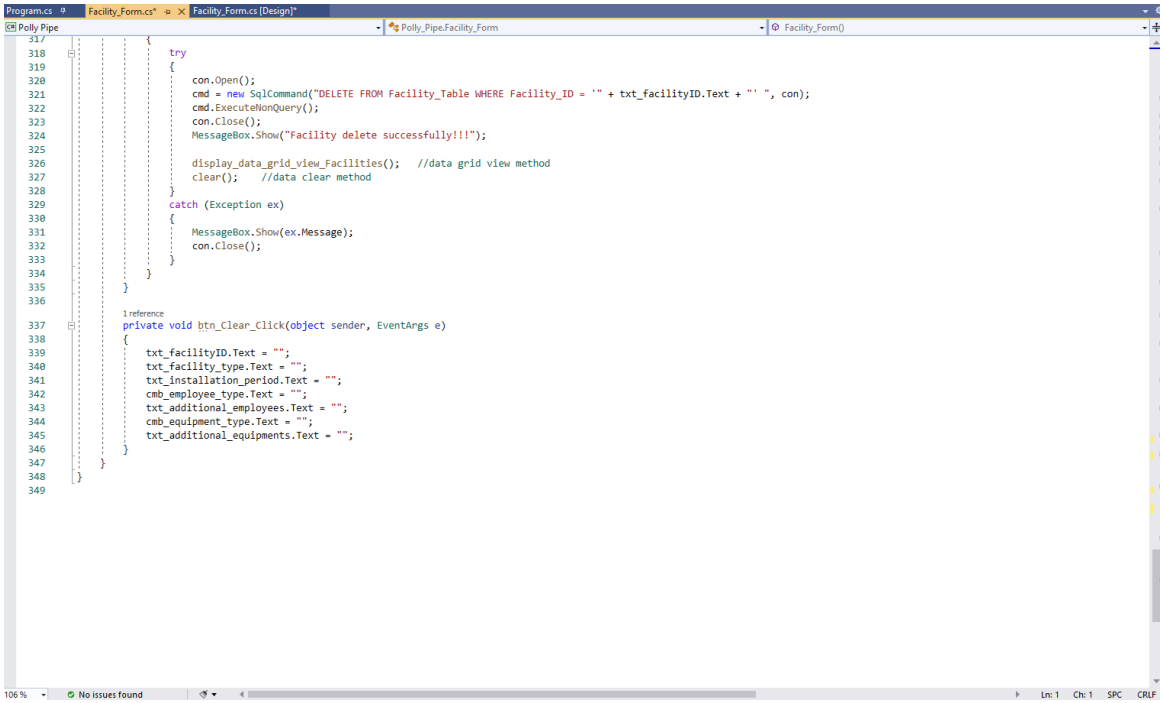
```
1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_facility_type.Text == "" || txt_installation_period.Text == "" || cmb_employee_type.Text == "" || txt_additional_employees.Text == ""
        || cmb_equipment_type.Text == "" || txt_additional_equipments.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Facility_Table(Facility_Type,Installation_Period,Employee_Type,Additional_Employees,Equipment_Type,Additional_Equipments) " +
                "VALUES('" + txt_facility_type.Text + "' , '" + txt_installation_period.Text + "' , '" + cmb_employee_type.Text + "' , " +
                "'" + txt_additional_employees.Text + "' , '" + cmb_equipment_type.Text + "' , '" + txt_additional_equipments.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Facility added successfully!!!");

            display_data_grid_view_Facilities();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_facility_type.Text == "" || txt_installation_period.Text == "" || cmb_employee_type.Text == "" || txt_additional_employees.Text == ""
        || cmb_equipment_type.Text == "" || txt_additional_equipments.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
```

Figure 2. 124 Facility Form (For Admin) code - part 7



```
            {
                try
                {

                    con.Open();
                    cmd = new SqlCommand("UPDATE Facility_Table SET Facility_Type = '" + txt_facility_type.Text + "' , Installation_Period = '" + txt_installation_period.Text + "' ," +
                        "  Employee_Type = '" + cmb_employee_type.Text + "', Additional_Employees = '" + txt_additional_employees.Text + "' ," +
                        "  Equipment_Type = '" + cmb_equipment_type.Text + "', Additional_Equipments = '" + txt_additional_equipments.Text + "'  " +
                        "WHERE Facility_ID = '" + txt_facilityID.Text + "'  ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Facility edit successfully!!!");

                    display_data_grid_view_Facilities();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (txt_facilityID.Text == "")
            {
                MessageBox.Show("Select Facility to Delete");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("DELETE FROM Facility_Table WHERE Facility_ID = '" + txt_facilityID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Facility delete successfully!!!");

                    display_data_grid_view_Facilities();   //data grid view method
                    clear();    //data clear method
                }
                catch (Exception ex)
                {
```

Figure 2. 125 Facility Form (For Admin) code - part 8



```
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("DELETE FROM Facility_Table WHERE Facility_ID = '" + txt_facilityID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Facility delete successfully!!!");

                    display_data_grid_view_Facilities();   //data grid view method
                    clear();    //data clear method
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Clear_Click(object sender, EventArgs e)
        {
            txt_facilityID.Text = "";
            txt_facility_type.Text = "";
            txt_installation_period.Text = "";
            cmb_employee_type.Text = "";
            txt_additional_employees.Text = "";
            cmb_equipment_type.Text = "";
            txt_additional_equipments.Text = "";
        }
    }
}
```

Figure 2. 126 Facility Form (For Admin) code - part 9

Installation Form (For Sales Representative) →

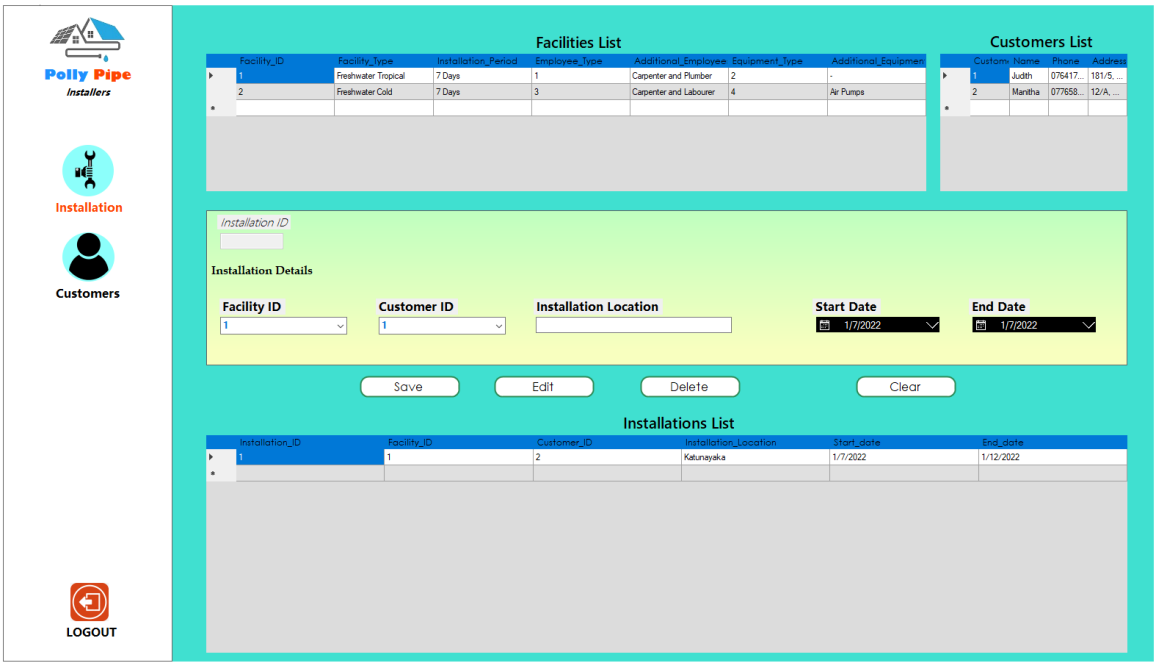


Figure 2. 127 Installation Form (For Sales Representative) interface

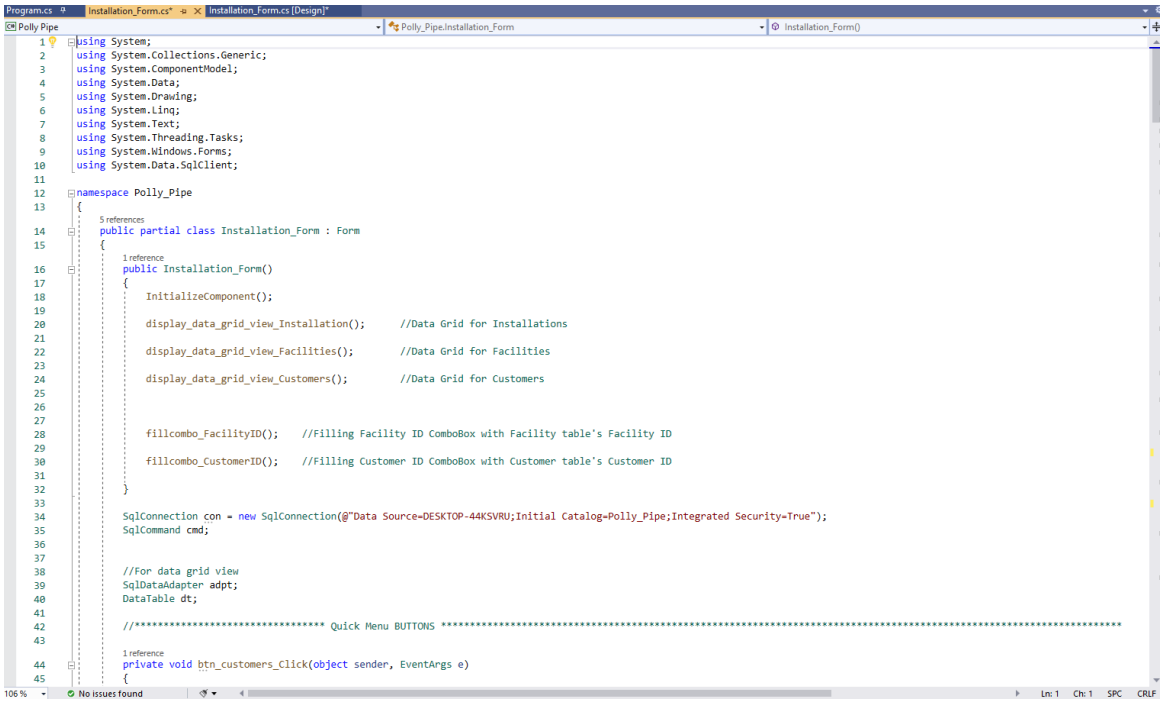
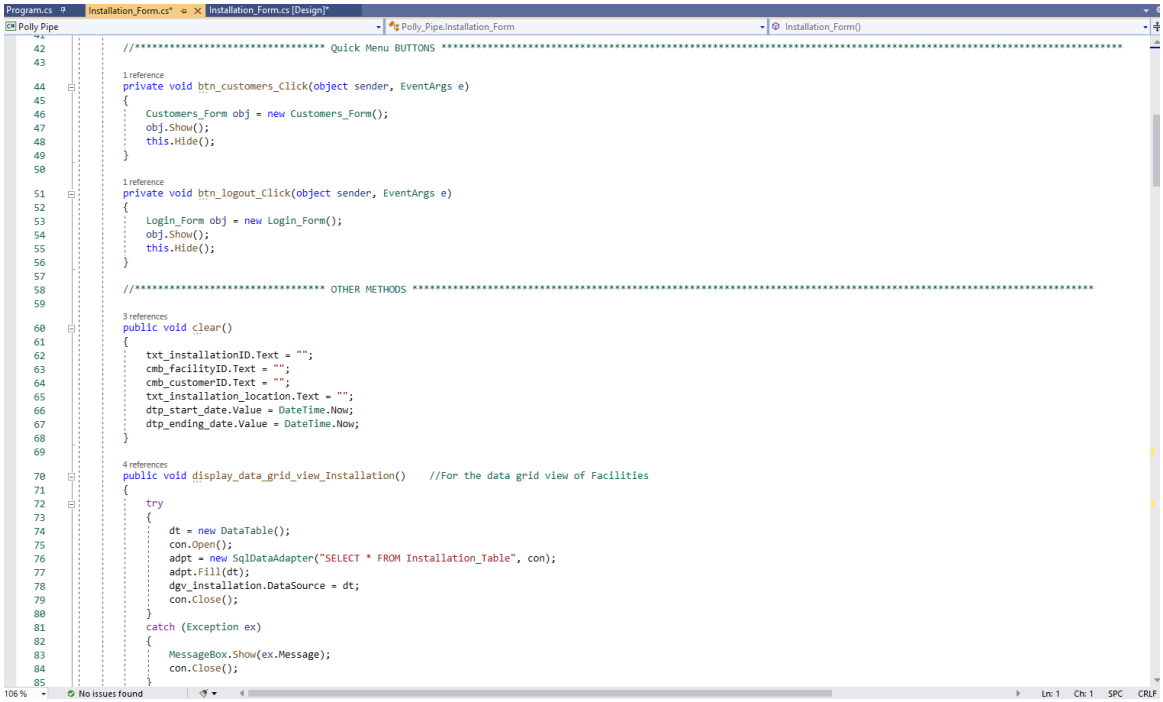
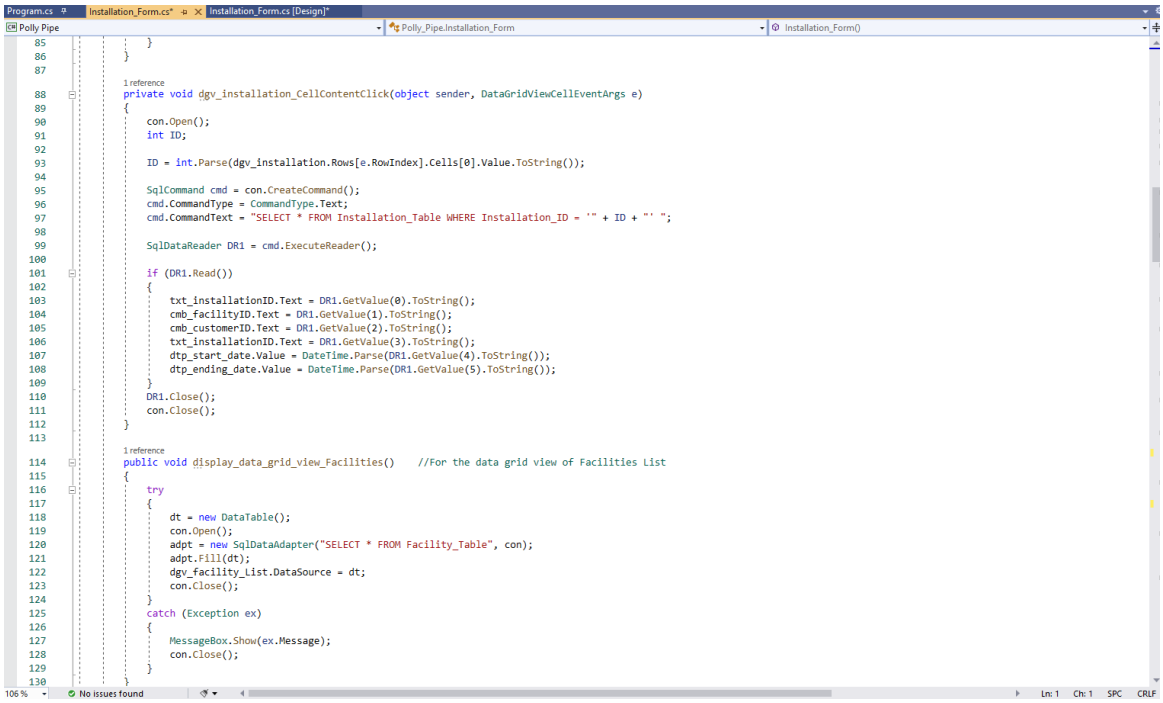


Figure 2. 128 Installation Form (For Sales Representative) code - part 1



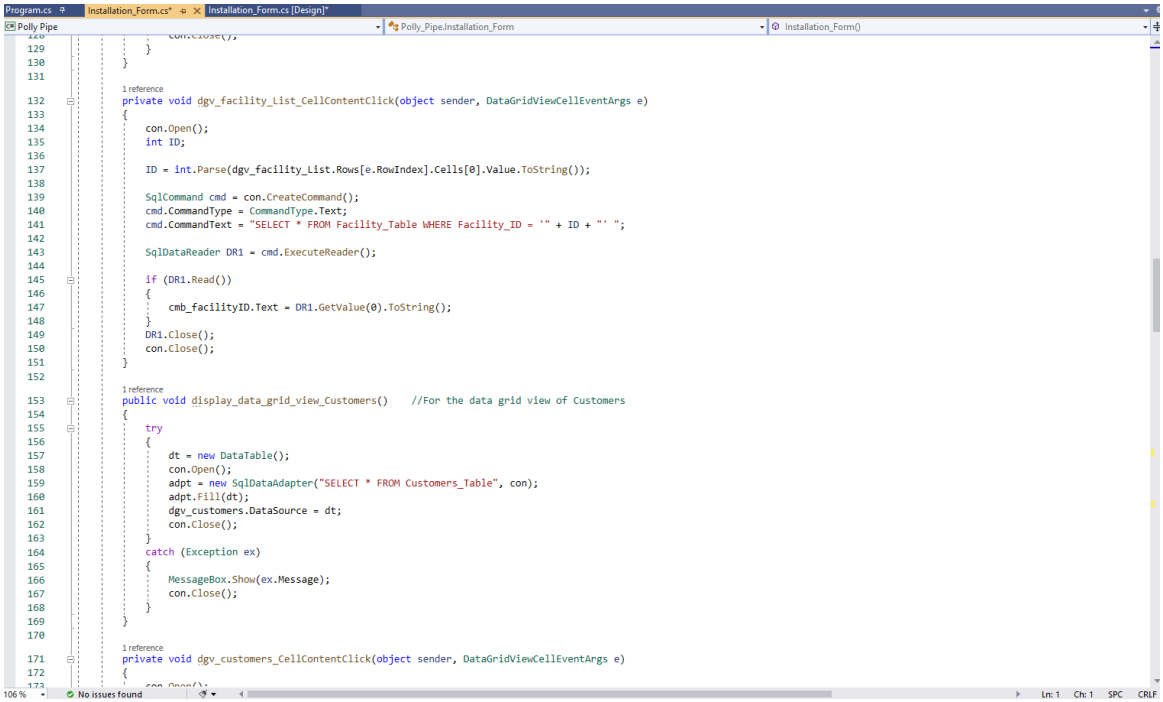
```
//********************************* Quick Menu BUTTONS ***********************************************************************************************************

1 reference
private void btn_customers_Click(object sender, EventArgs e)
{
    Customers_Form obj = new Customers_Form();
    obj.Show();
    this.Hide();
}

1 reference
private void btn_logout_Click(object sender, EventArgs e)
{
    Login_Form obj = new Login_Form();
    obj.Show();
    this.Hide();
}

//********************************* OTHER METHODS ****************************************************************************************************************

3 references
public void clear()
{
    txt_installationID.Text = "";
    cmb_facilityID.Text = "";
    cmb_customerID.Text = "";
    txt_installation_location.Text = "";
    dtp_start_date.Value = DateTime.Now;
    dtp_ending_date.Value = DateTime.Now;
}

4 references
public void display_data_grid_view_Installation()    //For the data grid view of Facilities
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Installation_Table", con);
        adpt.Fill(dt);
        dgv_installation.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
```

Figure 2. 129 Installation Form (For Sales Representative) code - part 2



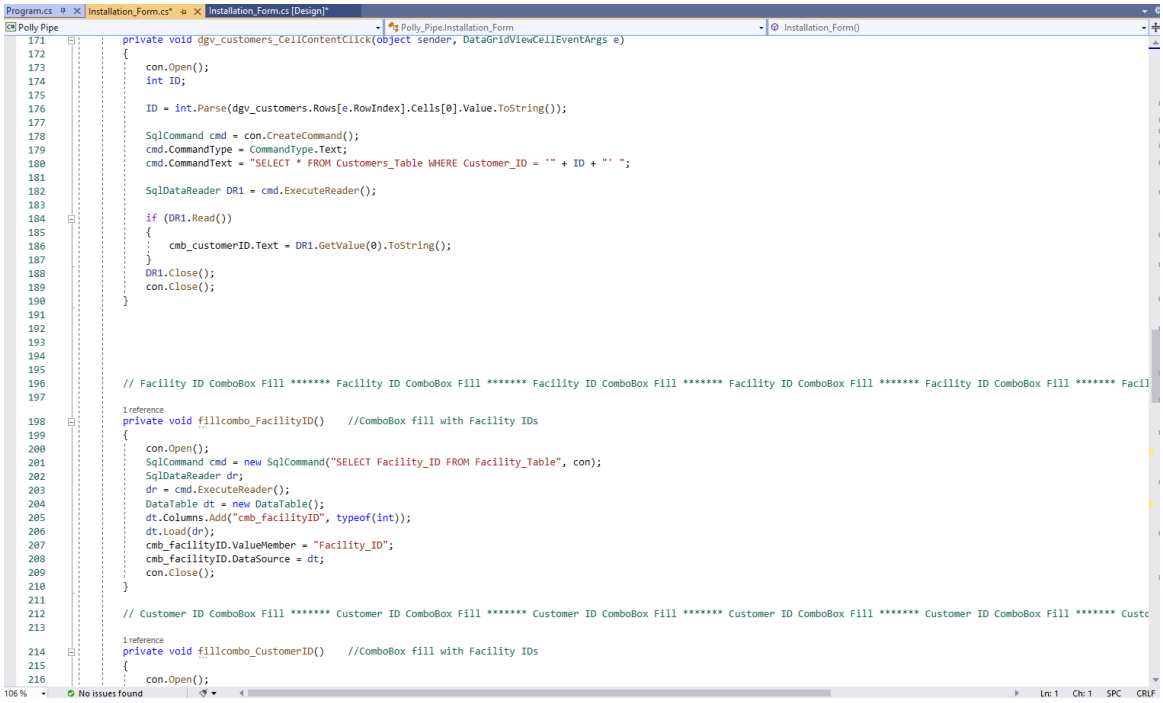
```
    }
}

1 reference
private void dgv_installation_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_installation.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Installation_Table WHERE Installation_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_installationID.Text = DR1.GetValue(0).ToString();
        cmb_facilityID.Text = DR1.GetValue(1).ToString();
        cmb_customerID.Text = DR1.GetValue(2).ToString();
        txt_installationID.Text = DR1.GetValue(3).ToString();
        dtp_start_date.Value = DateTime.Parse(DR1.GetValue(4).ToString());
        dtp_ending_date.Value = DateTime.Parse(DR1.GetValue(5).ToString());
    }
    DR1.Close();
    con.Close();
}

1 reference
public void display_data_grid_view_Facilities()    //For the data grid view of Facilities List
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Facility_Table", con);
        adpt.Fill(dt);
        dgv_facility_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}
```

Figure 2. 130 Installation Form (For Sales Representative) code - part 3



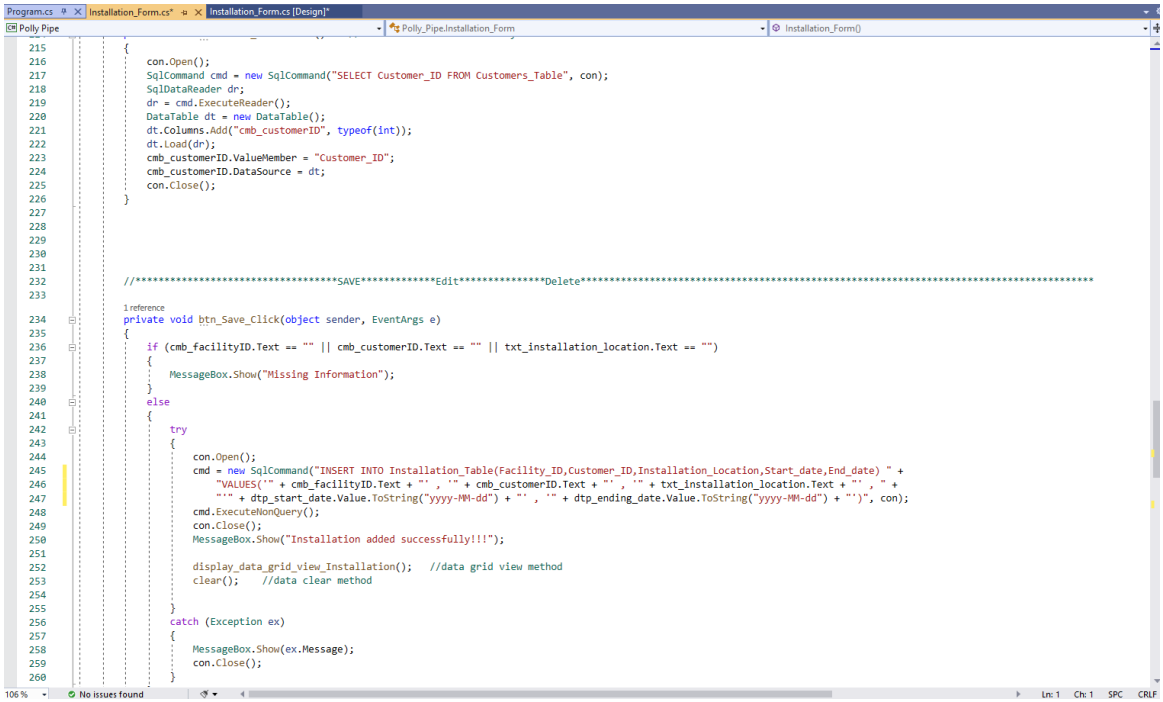
```
Program.cs | Installation_Form.cs* | Installation_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Installation_Form | Installation_Form()

            con.Close();
        }
    }

    1 reference
    private void dgv_facility_List_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_facility_List.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Facility_Table WHERE Facility_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            cmb_facilityID.Text = DR1.GetValue(0).ToString();
        }
        DR1.Close();
        con.Close();
    }

    1 reference
    public void display_data_grid_view_Customers()    //For the data grid view of Customers
    {
        try
        {
            dt = new DataTable();
            con.Open();
            adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
            adpt.Fill(dt);
            dgv_customers.DataSource = dt;
            con.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

    1 reference
    private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();

106 %  No issues found  Ln: 1  Ch: 1  SPC  CRLF
```

Figure 2. 131 Installation Form (For Sales Representative) code - part 4



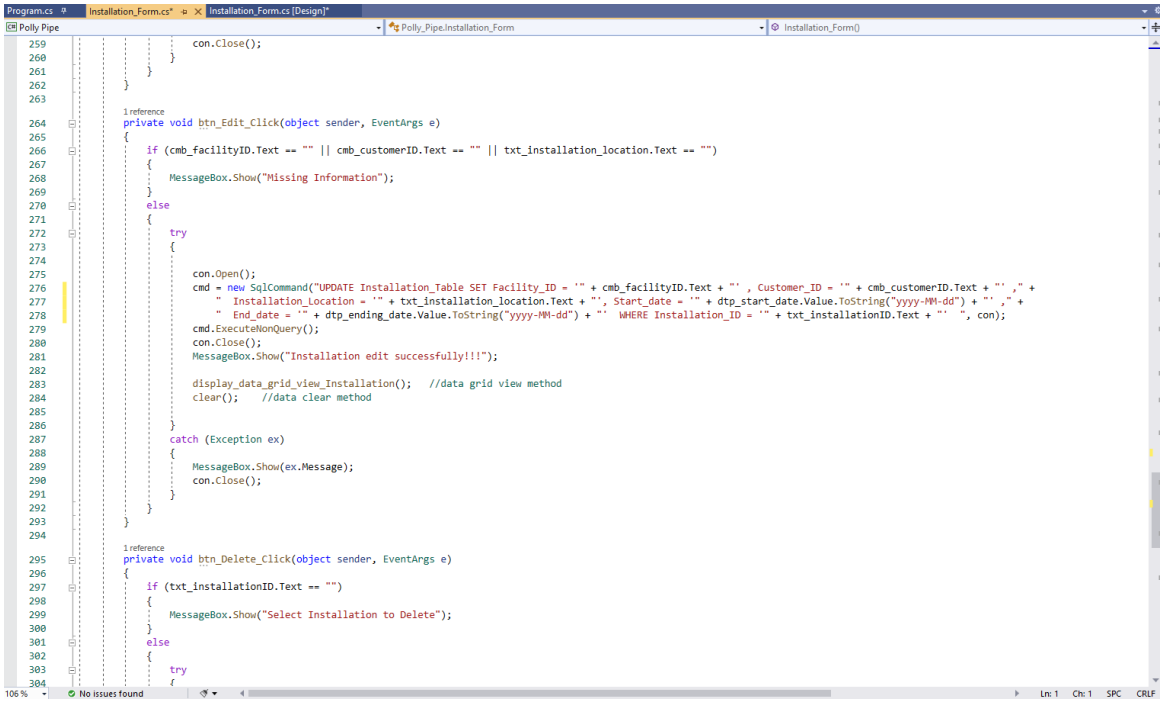
```
Program.cs | Installation_Form.cs* | Installation_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Installation_Form | Installation_Form()

    private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        con.Open();
        int ID;

        ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());

        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Customers_Table WHERE Customer_ID = '" + ID + "' ";

        SqlDataReader DR1 = cmd.ExecuteReader();

        if (DR1.Read())
        {
            cmb_customerID.Text = DR1.GetValue(0).ToString();
        }
        DR1.Close();
        con.Close();
    }




    // Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facil

    1 reference
    private void fillcombo_FacilityID()    //ComboBox fill with Facility IDs
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("SELECT Facility_ID FROM Facility_Table", con);
        SqlDataReader dr;
        dr = cmd.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Columns.Add("cmb_facilityID", typeof(int));
        dt.Load(dr);
        cmb_facilityID.ValueMember = "Facility_ID";
        cmb_facilityID.DataSource = dt;
        con.Close();
    }

    // Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Custo

    1 reference
    private void fillcombo_CustomerID()    //ComboBox fill with Facility IDs
    {
        con.Open();

106 %  No issues found  Ln: 1  Ch: 1  SPC  CRLF
```

Figure 2. 132 Installation Form (For Sales Representative) code - part 5



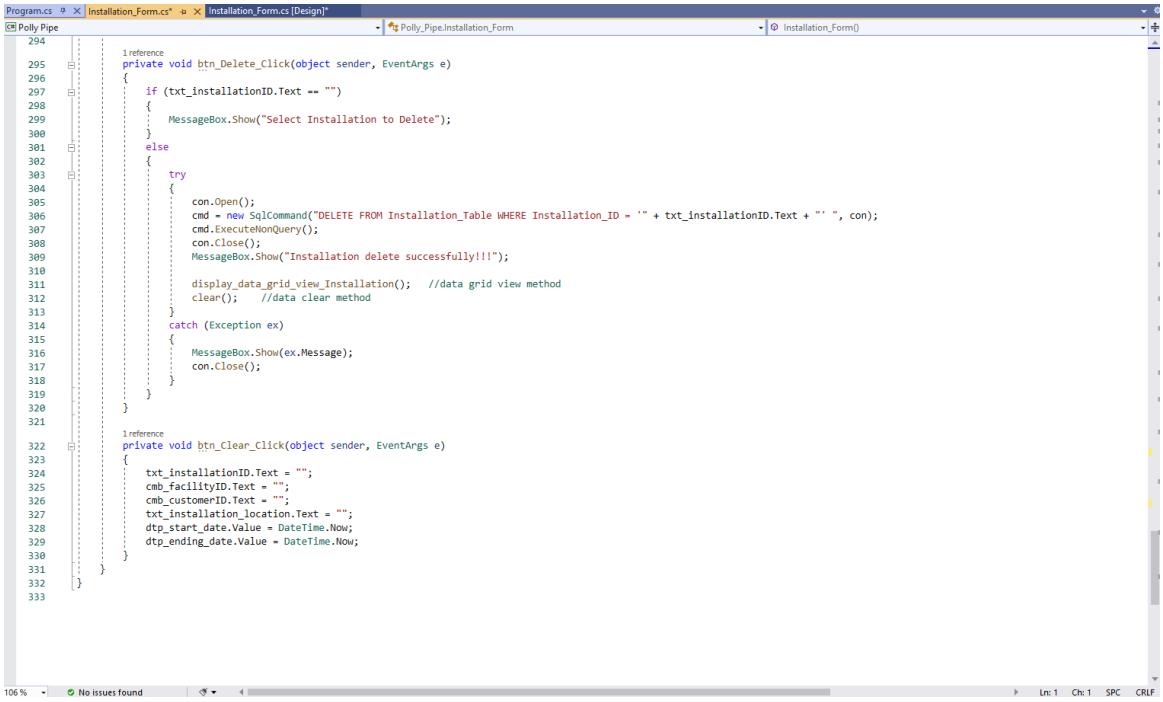
```
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Customer_ID FROM Customers_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_customerID", typeof(int));
    dt.Load(dr);
    cmb_customerID.ValueMember = "Customer_ID";
    cmb_customerID.DataSource = dt;
    con.Close();
}

//**********************************SAVE************Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (cmb_facilityID.Text == "" || cmb_customerID.Text == "" || txt_installation_location.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Installation_Table(Facility_ID,Customer_ID,Installation_Location,Start_date,End_date) " +
                "VALUES('" + cmb_facilityID.Text + "' , '" + cmb_customerID.Text + "' , '" + txt_installation_location.Text + "' , " +
                "'" + dtp_start_date.Value.ToString("yyyy-MM-dd") + "' , '" + dtp_ending_date.Value.ToString("yyyy-MM-dd") + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Installation added successfully!!!");

            display_data_grid_view_Installation();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
```

Figure 2. 133 Installation Form (For Sales Representative) code - part 6



```
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (cmb_facilityID.Text == "" || cmb_customerID.Text == "" || txt_installation_location.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
            cmd = new SqlCommand("UPDATE Installation_Table SET Facility_ID = '" + cmb_facilityID.Text + "' , Customer_ID = '" + cmb_customerID.Text + "' ," +
                "  Installation_Location = '" + txt_installation_location.Text + "', Start_date = '" + dtp_start_date.Value.ToString("yyyy-MM-dd") + "' ," +
                "  End_date = '" + dtp_ending_date.Value.ToString("yyyy-MM-dd") + "'  WHERE Installation_ID = '" + txt_installationID.Text + "'  ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Installation edit successfully!!!");

            display_data_grid_view_Installation();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_installationID.Text == "")
    {
        MessageBox.Show("Select Installation to Delete");
    }
    else
    {
        try
        {
```

Figure 2. 134 Installation Form (For Sales Representative) code - part 7



```
1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_installationID.Text == "")
    {
        MessageBox.Show("Select Installation to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Installation_Table WHERE Installation_ID = '" + txt_installationID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Installation delete successfully!!!");

            display_data_grid_view_Installation();   //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
    txt_installationID.Text = "";
    cmb_facilityID.Text = "";
    cmb_customerID.Text = "";
    txt_installation_location.Text = "";
    dtp_start_date.Value = DateTime.Now;
    dtp_ending_date.Value = DateTime.Now;
}
}
}
```

Figure 2. 135 Installation Form (For Sales Representative) code - part 8

Customer Form (For Sales Representative) →

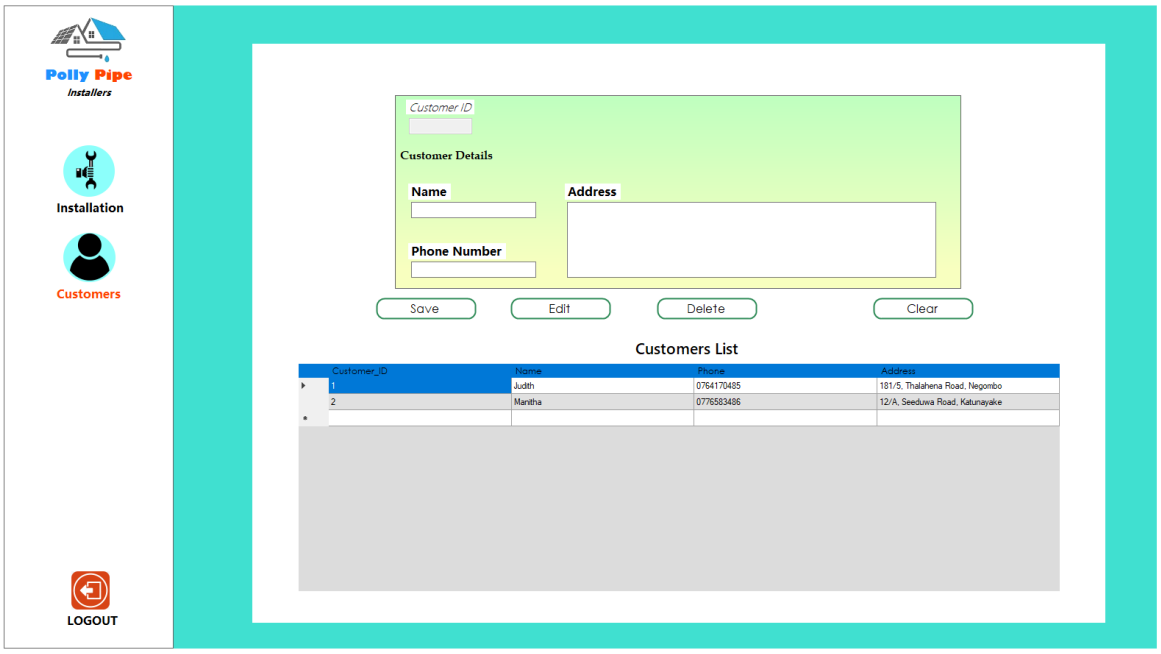


Figure 2. 136 Customer Form (For Sales Representative) interface

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    public partial class Customers_Form2 : Form
    {
        public Customers_Form2()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Customers Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***********************************************************************************************************

        private void btn_installation_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        private void btn_logout_Click(object sender, EventArgs e)
        {
```

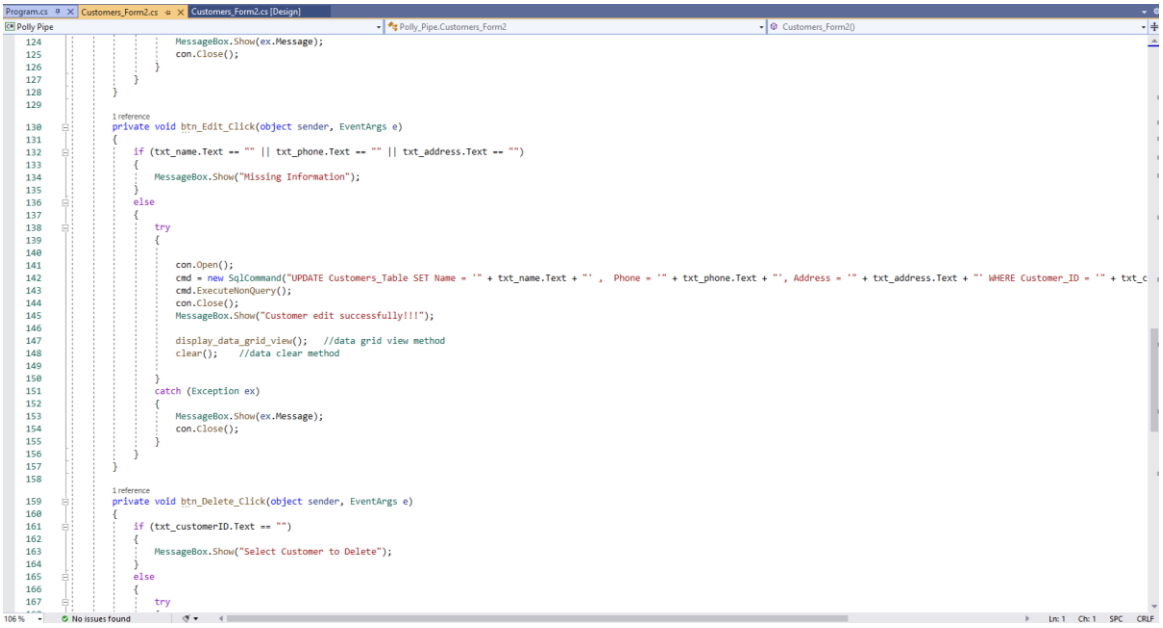
Figure 2. 137 Customer Form (For Sales Representative) code - part 1

```
Program.cs    Customers_Form2.cs    Customers_Form2.cs [Design]
Polly Pipe    Polly_Pipe.Customers_Form2    Customers_Form2()

        1 reference
        private void btn_logout_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }


        3 references
        public void clear()
        {
            txt_customerID.Text = "";
            txt_name.Text = "";
            txt_phone.Text = "";
            txt_address.Text = "";
        }

        4 references
        public void display_data_grid_view()    //For the data grid view
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
                adpt.Fill(dt);
                dgv_customers.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
```

Figure 2. 138 Customer Form (For Sales Representative) code - part 2

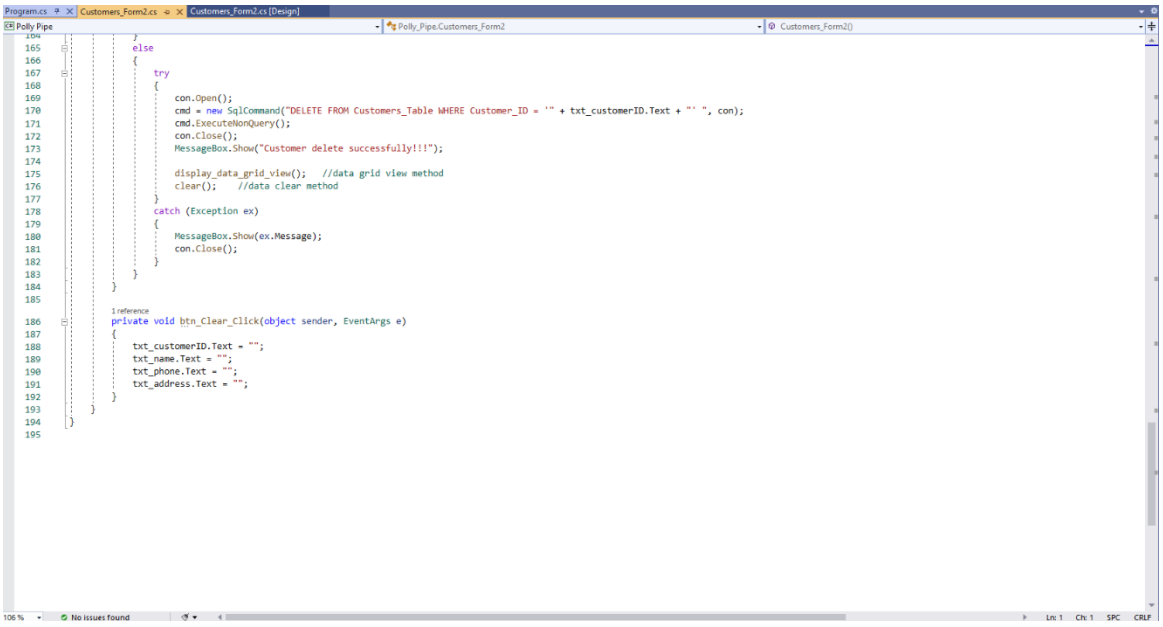
```
Program.cs    Customers_Form2.cs    Customers_Form2.cs [Design]
Polly Pipe    Polly_Pipe.Customers_Form2    Customers_Form2()

            ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Customers_Table WHERE Customer_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                txt_customerID.Text = DR1.GetValue(0).ToString();
                txt_name.Text = DR1.GetValue(1).ToString();
                txt_phone.Text = DR1.GetValue(2).ToString();
                txt_address.Text = DR1.GetValue(3).ToString();
            }
            DR1.Close();
            con.Close();
        }


        //**********************************SAVE************Edit**************Delete******************************************************************************************

        1 reference
        private void btn_Save_Click(object sender, EventArgs e)
        {
            if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("INSERT INTO Customers_Table(Name,Phone,Address) VALUES('" + txt_name.Text + "' , '" + txt_phone.Text + "' , '" + txt_address.Text + "')", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Customer added successfully!!!");

                    display_data_grid_view();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
106 %    No issues found    Ln: 1  Ch: 1  SPC  CRLF
```

Figure 2. 139 Customer Form (For Sales Representative) code - part 3



```
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Edit_Click(object sender, EventArgs e)
    {
        if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
        {
            MessageBox.Show("Missing Information");
        }
        else
        {
            try
            {

                con.Open();
                cmd = new SqlCommand("UPDATE Customers_Table SET Name = '" + txt_name.Text + "' ,  Phone = '" + txt_phone.Text + "', Address = '" + txt_address.Text + "' WHERE Customer_ID = '" + txt_c
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Customer edit successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Delete_Click(object sender, EventArgs e)
    {
        if (txt_customerID.Text == "")
        {
            MessageBox.Show("Select Customer to Delete");
        }
        else
        {
            try
```

Figure 2. 140 Customer Form (For Sales Representative) code - part 4



```
        else
        {
            try
            {
                con.Open();
                cmd = new SqlCommand("DELETE FROM Customers_Table WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Customer delete successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Clear_Click(object sender, EventArgs e)
    {
        txt_customerID.Text = "";
        txt_name.Text = "";
        txt_phone.Text = "";
        txt_address.Text = "";
    }
}
```

Figure 2. 141 Customer Form (For Sales Representative) code - part 5

2.7 User and System requirements to designed Polly Pipe System

Summery	
User Requirements	**System Requirements**
Login Form	
Login form to access the system.	Select job role (Admin or Representative).
Hide the password characters.	Enter login credentials for each job role.
Allowing admin to access only for Customer, Employee, Employee Types, Equipment and Facility forms.	
Allowing sales representative to access only for Installation and Customer forms.	
Customers Form	
A form to record customer details.	Entering customer details.
Ability to save, edit and delete records.	
Ability to clear fields values.	
Ability to see inserted records.	
Ability to retrieve inserted records to text boxes when needed.	
Employees Form	
A form to record employee details.	Entering employee details.
Ability to save, edit and delete records.	Select employee type id from combo box.
Ability to clear fields values.	
Ability to see inserted records.	

Ability to retrieve inserted records to text boxes when needed.	
To see Employee Type details before entering records.	
To enter Employee Type ID by choosing values.	
Employee Types Form	
A form to record employee details.	Entering employee type details.
Ability to save, edit and delete records.	
Ability to clear fields values.	
Ability to see inserted records.	
Ability to retrieve inserted records to text boxes when needed.	
Equipment Form	
A form to record equipment details.	Entering equipment details.
Ability to save, edit and delete records.	
Ability to clear fields values.	
Ability to see inserted records.	
Ability to retrieve inserted records to text boxes when needed.	
Facility Form	
A form to record facility details.	Entering facility details.
Ability to save, edit and delete records.	Select employee type id from combo box.
Ability to clear fields values.	Select equipment id from combo box.
Ability to see inserted records.	
Ability to retrieve inserted records to text boxes when needed.	

To see Employee Type details before entering records.	
To enter Employee Type ID by choosing values.	
To see Equipment details before entering records.	
To enter Equipment ID by choosing values.	
Entering employee list for each facility when necessary.	
Entering equipment list for each facility when necessary.	
Installation Form	
A form to record installation details.	Entering installation details.
Ability to save, edit and delete records.	Select facility id from combo box.
Ability to clear fields values.	Select customer id from combo box.
Ability to see inserted records.	Enter dates for installation.
Ability to retrieve inserted records to text boxes when needed.	
To see Facility details before entering records.	
To enter Facility ID by choosing values.	
To see Customer details before entering records.	
To enter Customer ID by choosing values.	

Table 2. 3 User and System requirements in brief

User Requirements	
User Requirement	**How I managed to fulfill user requirements**
Login Form	
Login form to access the system.	Creating “Login Form” as a doorway to the system.
Hide the password characters.	For this text box I’ve given password character “*” mark. Hence when user typing password in this text box it appears in “*” mark which provide better safety.
Allowing admin to access only for Customer, Employee, Employee Types, Equipment and Facility forms.	When admin give right security credentials in Login form, he/she is directed to Customer form. From there is a quick menu to and admin only have access to go to forms which he is allowed to.
Allowing sales representative to access only for Installation and Customer forms.	When sales representative gives right security credentials in Login form, he/she is directed to Installation form. From there is a quick menu to and representative only have access to go to forms which he is allowed to. For this I had to make a duplicate Customer form and named as “Customer Forms 2”. Because original Customer form quick menu panel has access to all Admin authorized forms. Since it is compromising the security access, I had to make another customer form but linked with same SQL database table. Only changed part is the quick menu panel which giving access to only Installation and Customer Form 2.

Customers Form	
A form to record customer details.	Creating “Customers Form” to record customer details.
Ability to save, edit and delete records.	Adding Save, Edit, Delete buttons to the form.
Ability to clear fields values.	Adding Clear button to the form.
Ability to see inserted records.	Adding DataGrid view to the form.
Ability to retrieve inserted records to text boxes when needed.	Insert cell double click function to retrieve inserted data into textboxes.
Employees Form	
A form to record employee details.	Creating “Employees Form” to record employee details.
Ability to save, edit and delete records.	Adding Save, Edit, Delete buttons to the form.
Ability to clear fields values.	Adding Clear button to the form.
Ability to see inserted records.	Adding DataGrid view to the form.
Ability to retrieve inserted records to text boxes when needed.	Insert cell double click function to retrieve inserted data into textboxes.
To see Employee Type details before entering records.	Adding DataGrid for Employee Types to see the inserted Employees Types form values.
To enter Employee Type ID by choosing given values.	I’ve added a combo box and made it a foreign key to Employee Types form to retrieve Employee Type ID list in the combo box.
Employee Types Form	
A form to record employee type details.	Creating “Employees Types Form” to record employee type details.
Ability to save, edit and delete records.	Adding Save, Edit, Delete buttons to the form.

Ability to clear fields values.	Adding Clear button to the form.
Ability to see inserted records.	Adding DataGrid view to the form.
Ability to retrieve inserted records to text boxes when needed.	Insert cell double click function to retrieve inserted data into textboxes.
Equipment Form	
A form to record equipment details.	Creating “Equipment Form” to record employee details.
Ability to save, edit and delete records.	Adding Save, Edit, Delete buttons to the form.
Ability to clear fields values.	Adding Clear button to the form.
Ability to see inserted records.	Adding DataGrid view to the form.
Ability to retrieve inserted records to text boxes when needed.	Insert cell double click function to retrieve inserted data into textboxes.
Facility Form	
A form to record facility details.	Creating “Facility Form” to record facility details.
Ability to save, edit and delete records.	Adding Save, Edit, Delete buttons to the form.
Ability to clear fields values.	Adding Clear button to the form.
Ability to see inserted records.	Adding DataGrid view to the form.
Ability to retrieve inserted records to text boxes when needed.	Insert cell double click function to retrieve inserted data into textboxes.
To see Employee Type details before entering records.	Adding DataGrid for Employee Types to see the inserted Employee Types form values.
To enter Employee Type ID by choosing values.	I’ve added a combo box and made it a foreign key to Employee Types form to retrieve Employee Type ID list in the combo box.

To see Equipment details before entering records.	Adding DataGrid for Equipment to see the inserted Equipment form values.
To enter Equipment ID by choosing values.	I've added a combo box and made it a foreign key to Equipment form to retrieve Equipment ID list in the combo box.
Entering employee list for each facility when necessary.	I've added text box to add the employee who is mainly handle task. Then I've added an additional text box to get a list of employee list for each facility when necessary. This is added as a description.
Entering equipment list for each facility when necessary.	I've added text box to add the equipment which is mainly handle task. Then I've added an additional text box to get a list of equipment list for each facility when necessary. This is added as a description.
Installation Form	
A form to record installation details.	Creating "Installation Form" to record installation details.
Ability to save, edit and delete records.	Adding Save, Edit, Delete buttons to the form.
Ability to clear fields values.	Adding Clear button to the form.
Ability to see inserted records.	Adding DataGrid view to the form.
Ability to retrieve inserted records to text boxes when needed.	Insert cell double click function to retrieve inserted data into textboxes.
To see Facility details before entering records.	Adding DataGrid for Facility to see the inserted Facility form values.
To enter Facility ID by choosing values.	I've added a combo box and made it a foreign key to Facility form to retrieve Facility ID list in the combo box.
To see Customer details before entering records.	Adding DataGrid for Customer to see the inserted Customer form values.

To enter Customer ID by choosing values.	I've added a combo box and made it a foreign key to Customer form to retrieve Customer ID list in the combo box.

Table 2. 4 User requirements with a full description How I managed to fulfill those requirements

System Requirements	
System Requirement	**How I managed to fulfill system requirements**
Login Form	
Select job role (Admin or Representative).	For this I've added combo box to select the job role.
Enter login credentials for each job role.	For this I've added text boxes to give login credentials.
Customers Form	
Entering customer details.	I've added text boxes to enter customer details. I've disabled the Customer ID text box since the database system automatically assign a ID value and it shouldn't be change since the is identity value is true.
Employees Form	
Entering employee details.	I've added text boxes to enter employee details. I've disabled the Employee ID text box since the database system automatically assign a ID value and it shouldn't be change since the is identity value is true.
Select employee type id from combo box	I've added a combo box and made it a foreign key to Employee Types form to retrieve Employee Type ID list in the combo box.
Employees Types Form	
Entering customer details.	I've added text boxes to enter employee type details.

	I've disabled the Employee Type ID text box since the database system automatically assign a ID value and it shouldn't be change since the is identity value is true.
Equipment Form	
Entering equipment details.	I've added text boxes to enter equipment details. I've disabled the Equipment ID text box since the database system automatically assign a ID value and it shouldn't be change since the is identity value is true.
Facility Form	
Entering facility details.	I've added text boxes to enter facility details. I've disabled the Facility ID text box since the database system automatically assign a ID value and it shouldn't be change since the is identity value is true.
Select employee type id from combo box.	I've added a combo box and made it a foreign key to Employee Types form to retrieve Employee Type ID list in the combo box.
Select equipment id from combo box.	I've added a combo box and made it a foreign key to Equipment form to retrieve Equipment ID list in the combo box.
Installation Form	
Entering installation details.	I've added text boxes to enter installation details. I've disabled the Installation ID text box since the database system automatically assign a ID value and it shouldn't be change since the is identity value is true.

Select facility id from combo box.	I've added a combo box and made it a foreign key to Facility form to retrieve Facility ID list in the combo box.
Select customer id from combo box.	I've added a combo box and made it a foreign key to Customer form to retrieve Customer ID list in the combo box.
Enter period for installation.	I've added date time picker to enter installation details.

Table 2. 5 System requirements with a full description How I managed to fulfill those requirements

Activity 3

3.1 Test the system against user and system requirements

In this subtopic I've documented 19 Test Cases according to above state User and System requirements. Instead of documenting all the test cases of above, I choose unique 19 test cases which is different to each other. I avoided to documenting similar type test cases to save time and space.

Ex :- Rather than documenting Save button testing for all forms I tested only Customer Form Save button.

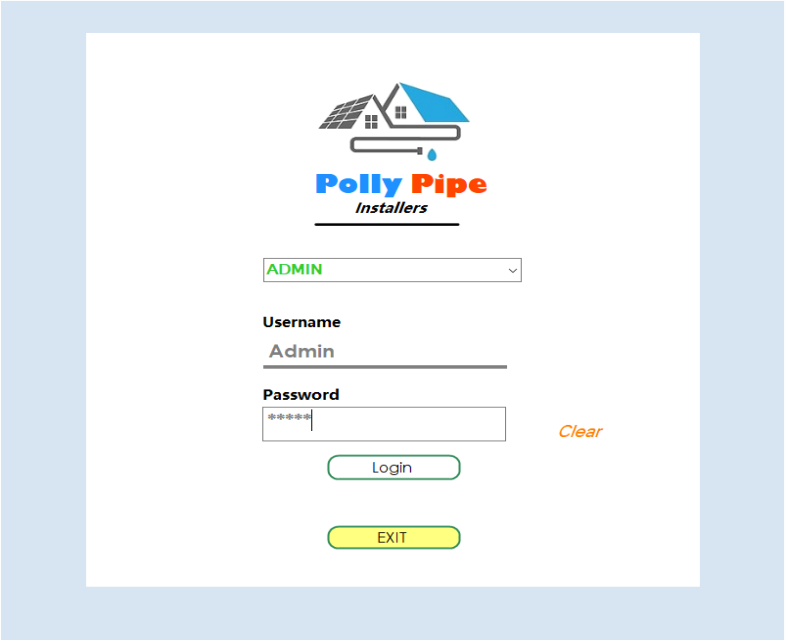
Test Case 1			
Function	Login Form :- Hide the password characters.		
Requirement Type	User Requirement		
Inputs	Admin Password : Admin Rep Password : Rep		
Expected Output	*****	**Actual Output**	*****
Result	Success		
Screenshot			

Table 3. 1 Test Case 1

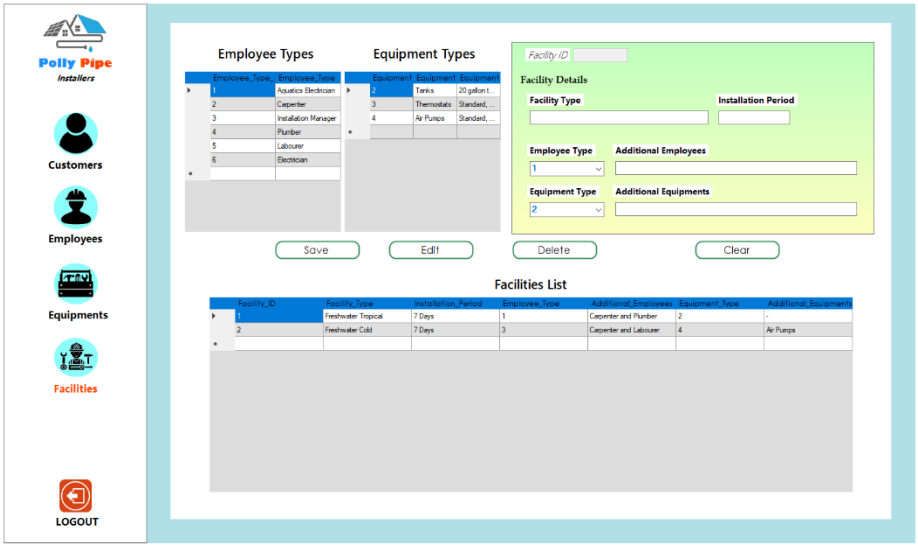
Test Case 2			
Function	Login Form :- Allowing admin to access only for Customer, Employee, Employee Types, Equipment and Facility forms.		
Requirement Type	User Requirement		
Inputs	In login form select ADMIN in combo box then enter username and password as “Admin”. Then hit login button. After redirecting to Customer Form, at the left side corner there is Quick Menu panel. Click any button to go Admin authorized forms. For this test case click on Facility button.		
Expected Output	Facilities Form	**Actual Output**	Facility Form
Result	Success		
Screenshot			

Table 3. 2 Test Case 2

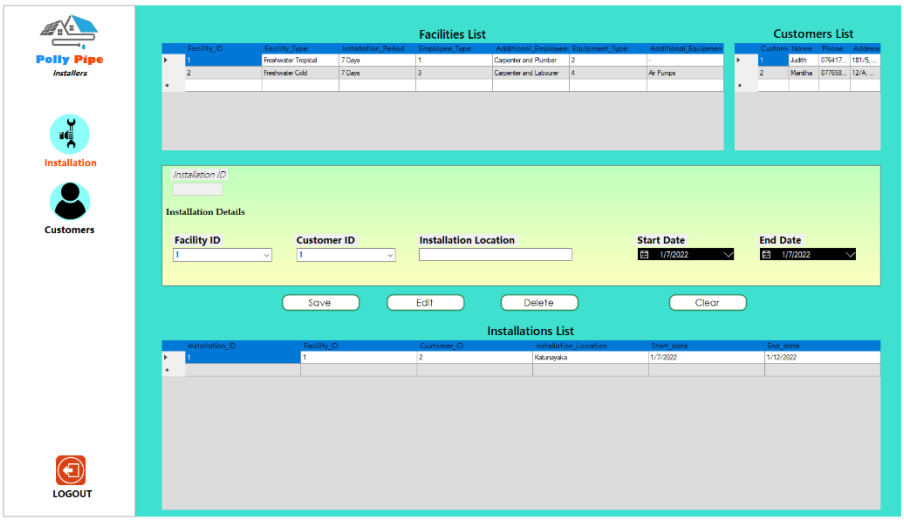
Test Case 3			
Function	Login Form :- Allowing sales representative to access only for Installation and Customer forms.		
Requirement Type	User Requirement		
Inputs	In login form select REPRESENTATIVE in combo box then enter username and password as “Rep”. Then hit login button. After redirecting to Installation Form, at the left side corner there is Quick Menu panel. Click any button to go Sales representative authorized forms. For this test case click on Customer button.		
Expected Output	Customers Form	**Actual Output**	Customers Form
Result	Success		
Screenshot			

Table 3. 3 Test Case 3

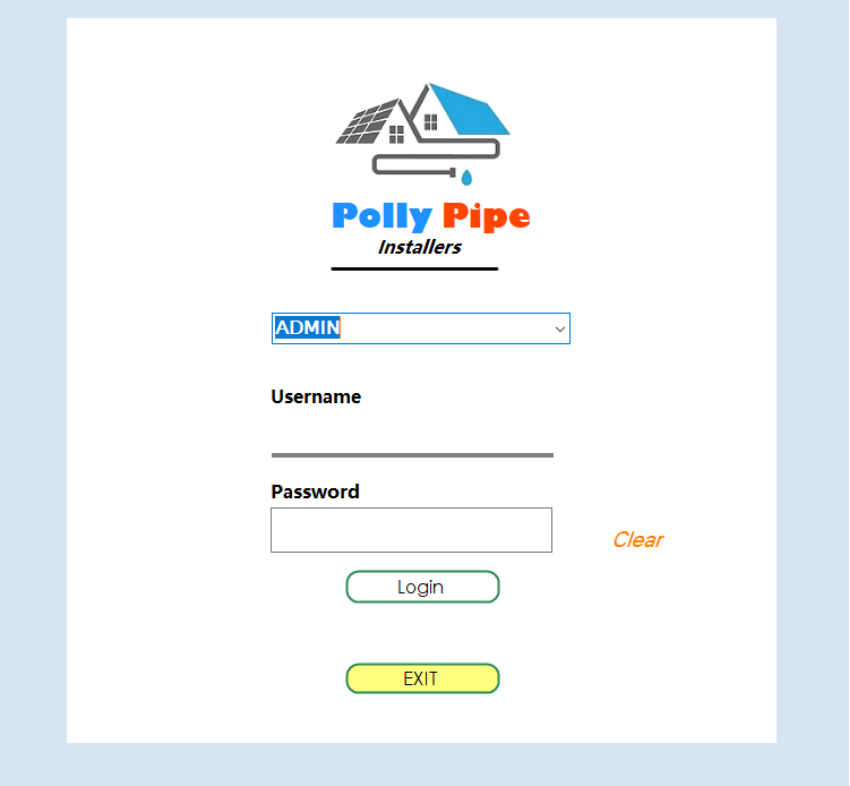
Test Case 4			
Function	Login Form :- Select job role (Admin or Representative.		
Requirement Type	System Requirement		
Inputs	Click on “Select a Role” combo box to see selection menu. Click on ADMIN.		
Expected Output	Combo box characters “ADMIN”	**Actual Output**	Combo box characters “ADMIN”
Result	Success		
Screenshot			

Table 3. 4 Test Case 4

Test Case 5			
Function	Login Form :- Enter login credentials for each job role.		
Requirement Type	System Requirement		
Inputs	Click on “Select a Role” combo box to see selection menu. Click on ADMIN. Then give inputs for following text boxes as following. Username : Admin Password : Admin		
Expected Output	Combo box value : ADMIN Username text box value : Admin Password text box value : *****	**Actual Output**	Combo box value : ADMIN Username text box value : Admin Password text box value : *****
Result	Success		
Screenshot			

Table 3. 5 Test Case 5

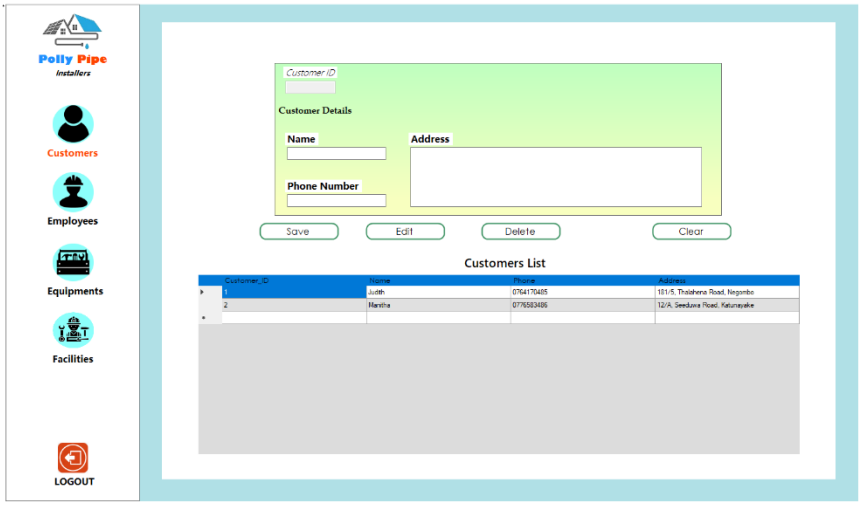
Test Case 6			
Function	Customers Form :- A form to record customer details.		
Requirement Type	User Requirement		
Inputs	After giving right login credential for admin in Login form hit Login button to go Customers Form.		
Expected Output	Customers Form	**Actual Output**	Customers Form
Result	Success		
Screenshot			

Table 3. 6 Test Case 6

Test Case 7			
Function	Customers Form :- A form to record customer details.		
Requirement Type	User Requirement		
Inputs	After giving wrong login credential for admin in Login form hit Login button to go Customers Form.		
Expected Output	Customers Form	**Expected Output**	Login Form message box error
Result	Failed		
Screenshot			

Table 3. 7 Test Case 7

<table>
<tr><th colspan="4">Test Case 8</th></tr>
<tr><td>Function</td><td colspan="3">Customers Form :- Ability to save, edit and delete records.</td></tr>
<tr><td>Requirement Type</td><td colspan="3">User Requirement</td></tr>
<tr><td>Inputs</td><td colspan="3">Save :- Fill text boxes fields and hit Save button to save.

Edit :- Double click on any row on DataGrid view which want to be edited. Then make changes and hit Edit button to edit.

Delete :- Double click on any row on DataGrid view which want to be deleted. Then hit Delete button to delete.</td></tr>
<tr><td>Expected Output</td><td>Save :- Message box “Customer added successfully!!!”

Edit :- Message box “Customer edit successfully!!!”

Delete :- Message box “Customer delete successfully!!!”</td><td>Actual Output</td><td>Save :- Message box “Customer added successfully!!!”

Edit :- Message box “Customer edit successfully!!!”

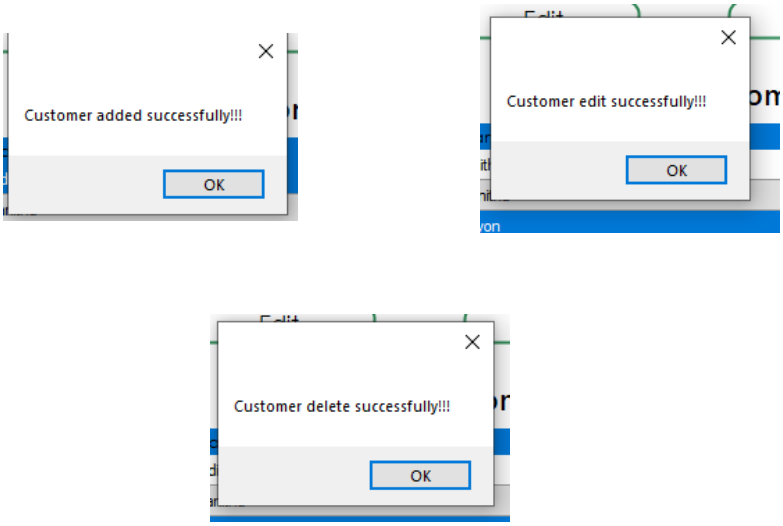
Delete :- Message box “Customer delete successfully!!!”</td></tr>
<tr><td>Result</td><td colspan="3">Success</td></tr>
<tr><td>Screenshot</td><td colspan="3"> 
</td></tr>
</table>

Table 3. 8 Test Case 8

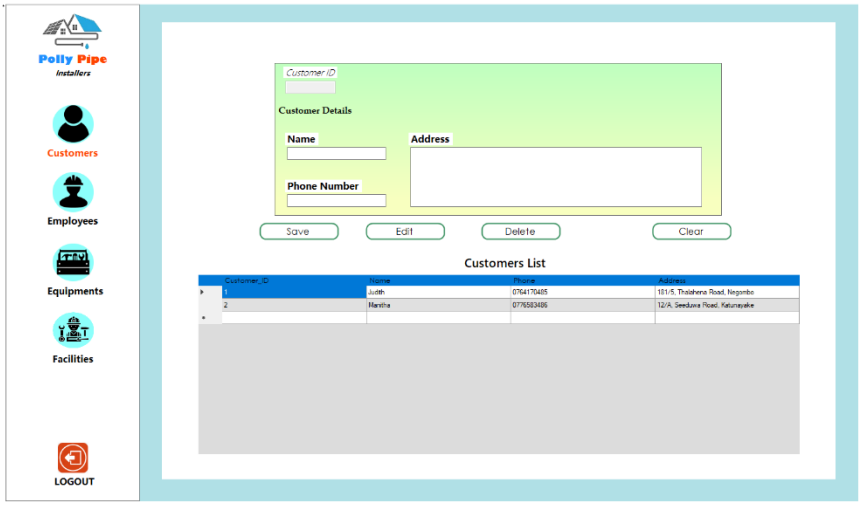
Test Case 9			
Function	Customers Form :- Ability to clear fields values.		
Requirement Type	User Requirement		
Inputs	After filling text box fields hit Clear button		
Expected Output	Empty text boxes	**Actual Output**	Empty text boxes
Result	Success		
Screenshot			

Table 3. 9 Test Case 9

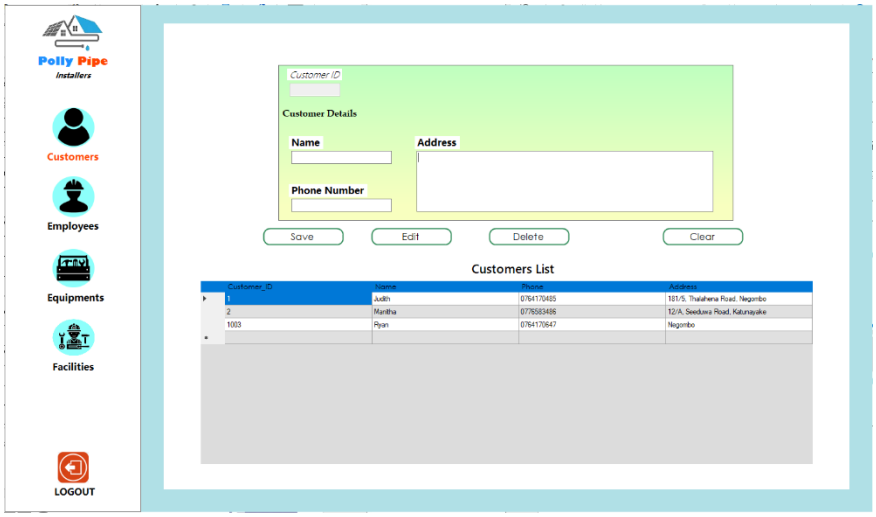
Test Case 10			
Function	Customers Form :- Ability to see inserted records.		
Requirement Type	User Requirement		
Inputs	After saving a new record click “ok” button on Saved Successful message box.		
Expected Output	DataGrid view with the new record.	**Actual Output**	DataGrid view with the new record.
Result	Success		
Screenshot			

Table 3. 10 Test Case 10

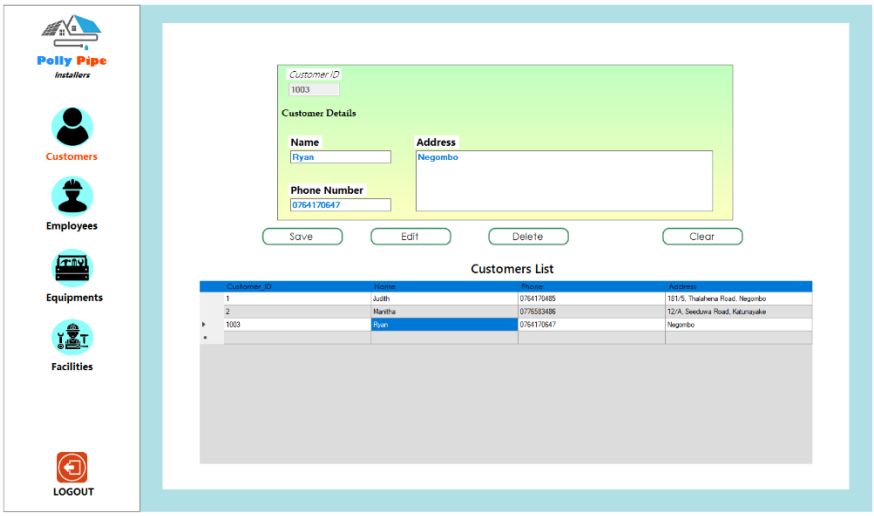
Test Case 11			
Function	Customers Form :- Ability to retrieve inserted records to text boxes when needed.		
Requirement Type	User Requirement		
Inputs	Click on cell content “Ryan” of DataGrid view		
Expected Output	Customer Ryan’s information filled in text boxes fields.	**Actual Output**	Customer Ryan’s information filled in text boxes fields.
Result	Success		
Screenshot			

Table 3. 11 Test Case 11

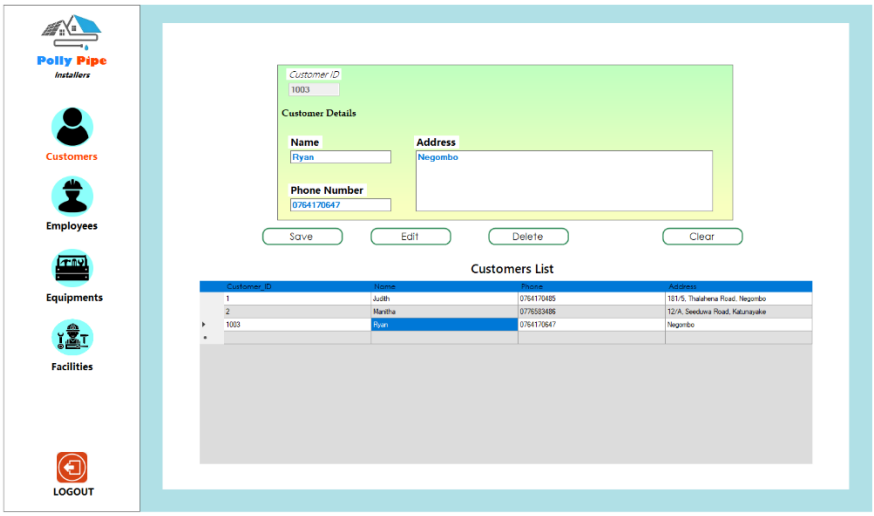
Test Case 12			
Function	Customers Form :- Entering customer details.		
Requirement Type	System Requirement		
Inputs	Fill the Customer Form text boxes fields		
Expected Output	Customer Form text boxes fields filled with given inputs.	**Actual Output**	Customer Form text boxes fields filled with given inputs.
Result	Success		
Screenshot			

Table 3. 12 Test Case 12

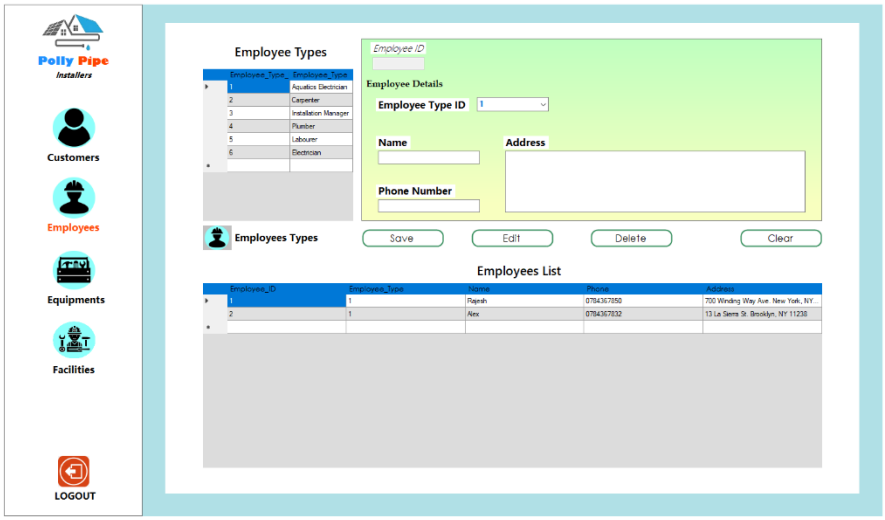
Test Case 13			
Function	Employees Form :- To see Employee Type details before entering records.		
Requirement Type	User Requirement		
Inputs	Open the Employees form by click on Employee Button on quick menu panel		
Expected Output	Employees Form with Employee Type DataGrid view.	**Actual Output**	Employees Form with Employee Type DataGrid view.
Result	Success		
Screenshot			

Table 3. 13 Test Case 13

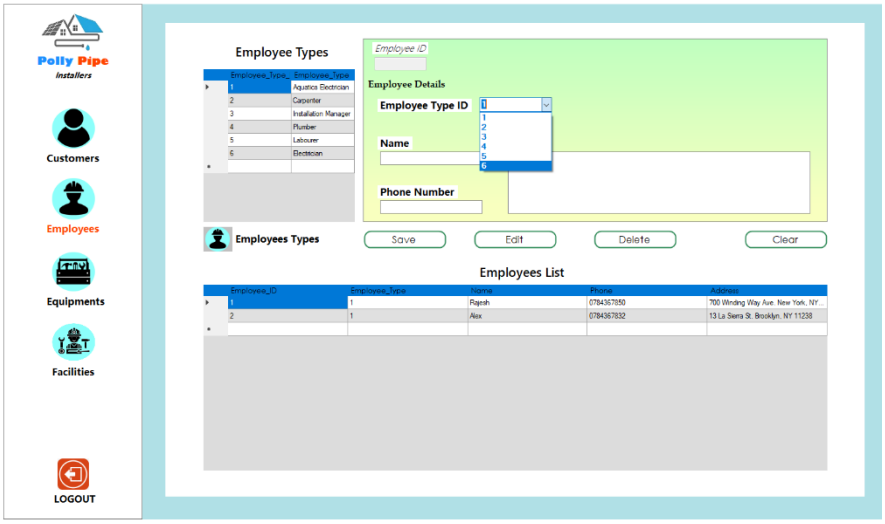
Test Case 14			
Function	Employees Form :- To enter Employee Type ID by choosing values.		
Requirement Type	User Requirement		
Inputs	Click on Employee Type ID combo box to see selection menu.		
Expected Output	List of Employee Type IDs which can be seen in DataGrid view.	**Actual Output**	List of Employee Type IDs which can be seen in DataGrid view.
Result	Success		
Screenshot			

Table 3. 14 Test Case 14

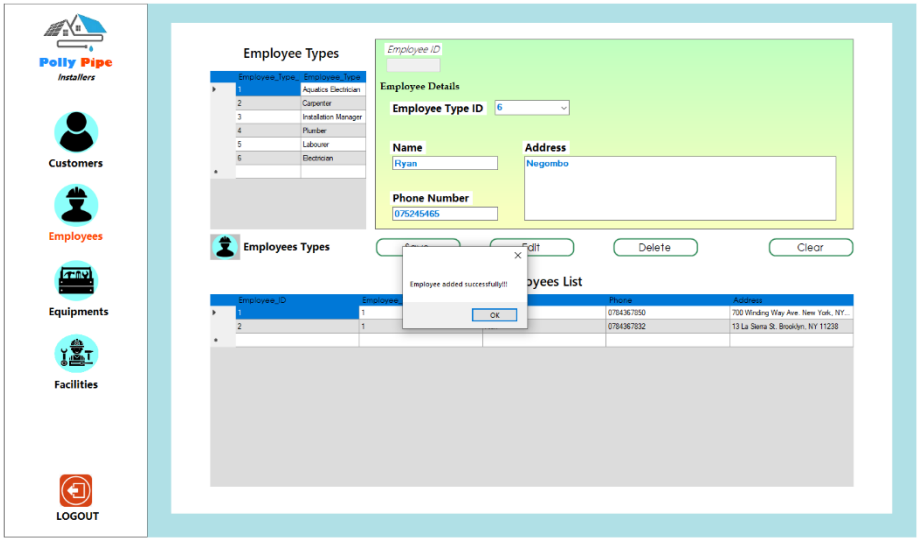
Test Case 15			
Function	Employees Form :- Select employee type id from combo box.		
Requirement Type	System Requirement		
Inputs	Click on Employee Type ID combo box to see selection menu. Then click on number 6 then enter some Employee details then hit Save button.		
Expected Output	Message box “Employee added successfully!!!”	**Actual Output**	Message box “Employee added successfully!!!”
Result	Success		
Screenshot			

Table 3. 15 Test Case 15

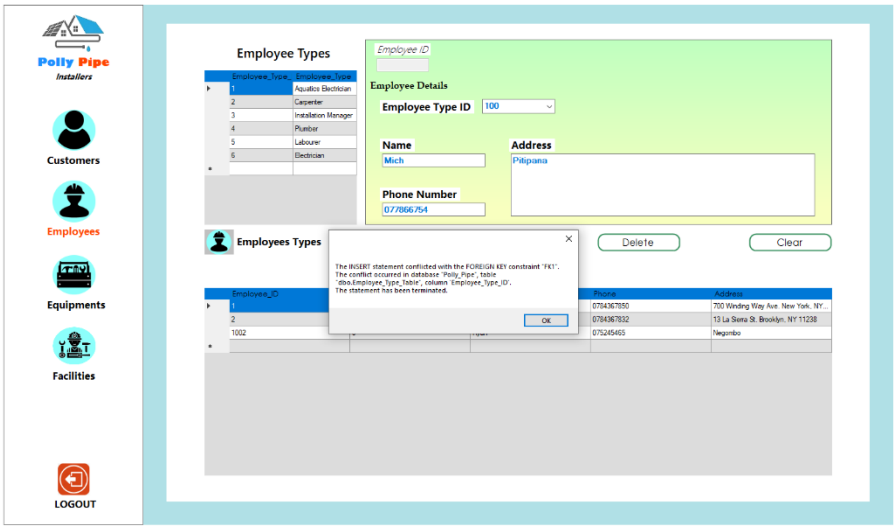
Test Case 16			
Function	Employees Form :- Select employee type id from combo box.		
Requirement Type	System Requirement		
Inputs	Click on Employee Type ID combo box to see selection menu. Then click on number 100 then enter some Employee details then hit Save button.		
Expected Output	Message box "Employee added successfully!!!"	**Actual Output**	Message box which describes the Insert recording to database has been terminated since the value 100 is out of range of Foreign Key (Employee Type ID) values which has been given.
Result	Failed		
Screenshot			

Table 3. 16 Test Case 16

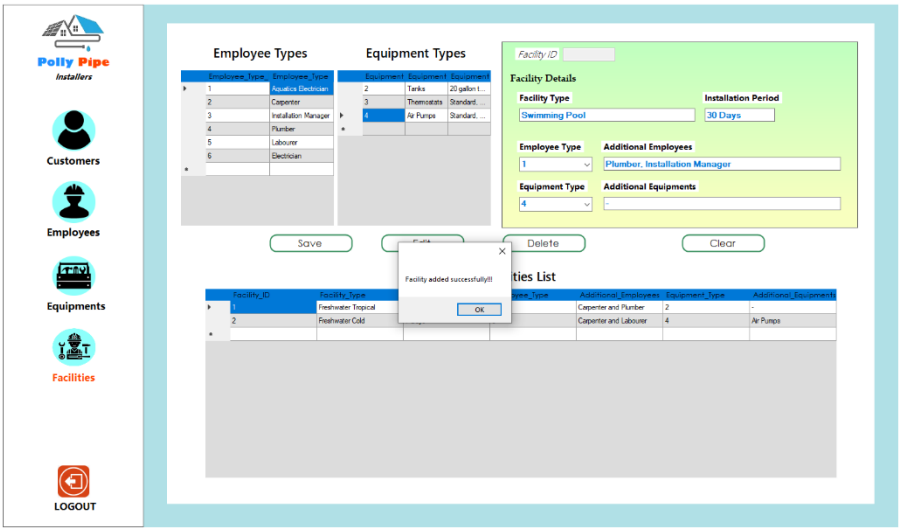
Test Case 17			
Function	Facilities Form :- Entering employee list for each facility when necessary.		
Requirement Type	User Requirement		
Inputs	Fill text boxes values in Facilities form as usual and in “Additional Employees” text box give Employee list which below 50 varchar characters.		
Expected Output	Message box “Facility added successfully!!!”	**Actual Output**	Message box “Facility added successfully!!!”
Result	Success		
Screenshot			

Table 3. 17 Test Case 17

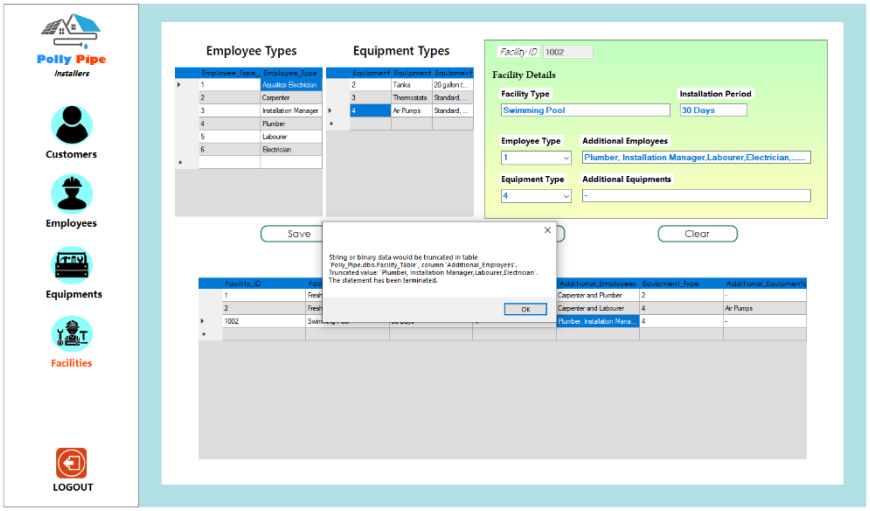
Test Case 18			
Function	Facilities Form :- Entering employee list for each facility when necessary.		
Requirement Type	User Requirement		
Inputs	Fill text boxes values in Facilities form as usual and in “Additional Employees” text box give Employee list which above 50 varchar characters.		
Expected Output	Message box “Facility added successfully!!!”	**Actual Output**	Message box which describes the Insert recording to database has been terminated. Because the given varchar characters is out of range SQL table varchar character limit which is 50.
Result	Failed		
Screenshot			

Table 3. 18 Test Case 18

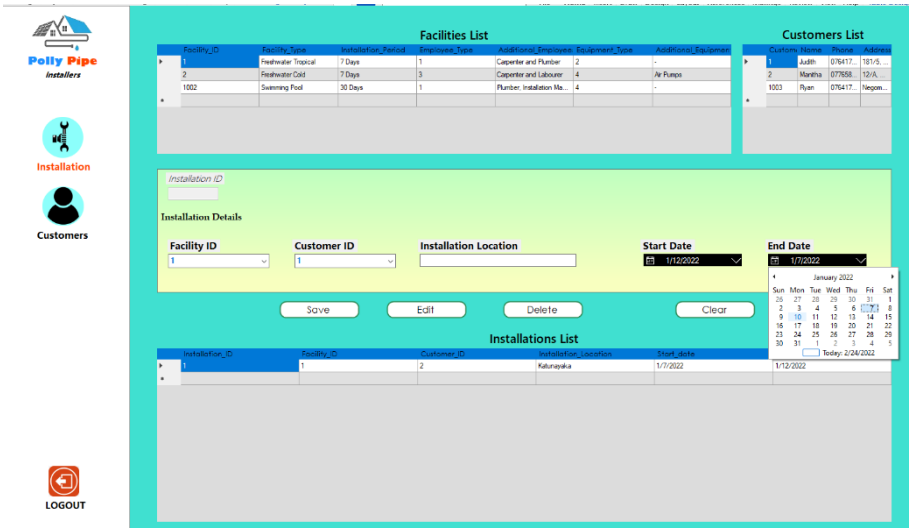
Test Case 19			
Function	Installation Form :- Enter dates for installation.		
Requirement Type	System Requirement		
Inputs	Click on Date Time Picker in the form and select a date.		
Expected Output	Can be select a date	**Actual Output**	Can select a date
Result	Success		
Screenshot			

Table 3. 19 Test Case 19

3.1.1 Effectiveness of the testing

For testing a software product, we have a wide variety of software testing approaches at our fingertips. In software testing, the method used has an effect on both the process and the quality of the final result. Developers and testers can use test cases to find faults that happened during development or flaws that were missed.

As a result, we must develop an effective and efficient testing method. However, comparing testing procedures just on fault detection ability is insufficient. They should also be assessed to see which one improves reliability the greatest. To develop a meaningful testing theory, we must assess existing and new testing procedures for their effectiveness, as well as their ability to improve software reliability.

Because of the testing the system while creating, it helped me to create a quality software with minimal flaws in the system. When finishing each form of the system, I intended to use test cases to detect faults and most of times I was happened to face different types of bugs. Sometimes the flaws were related to database or sometimes they were related to the system coding. Despite of encountering lot of bugs while testing, I was able to create a sophisticated application.

3.2 Feedback Form

3.2.1 Feedback form of sample survey

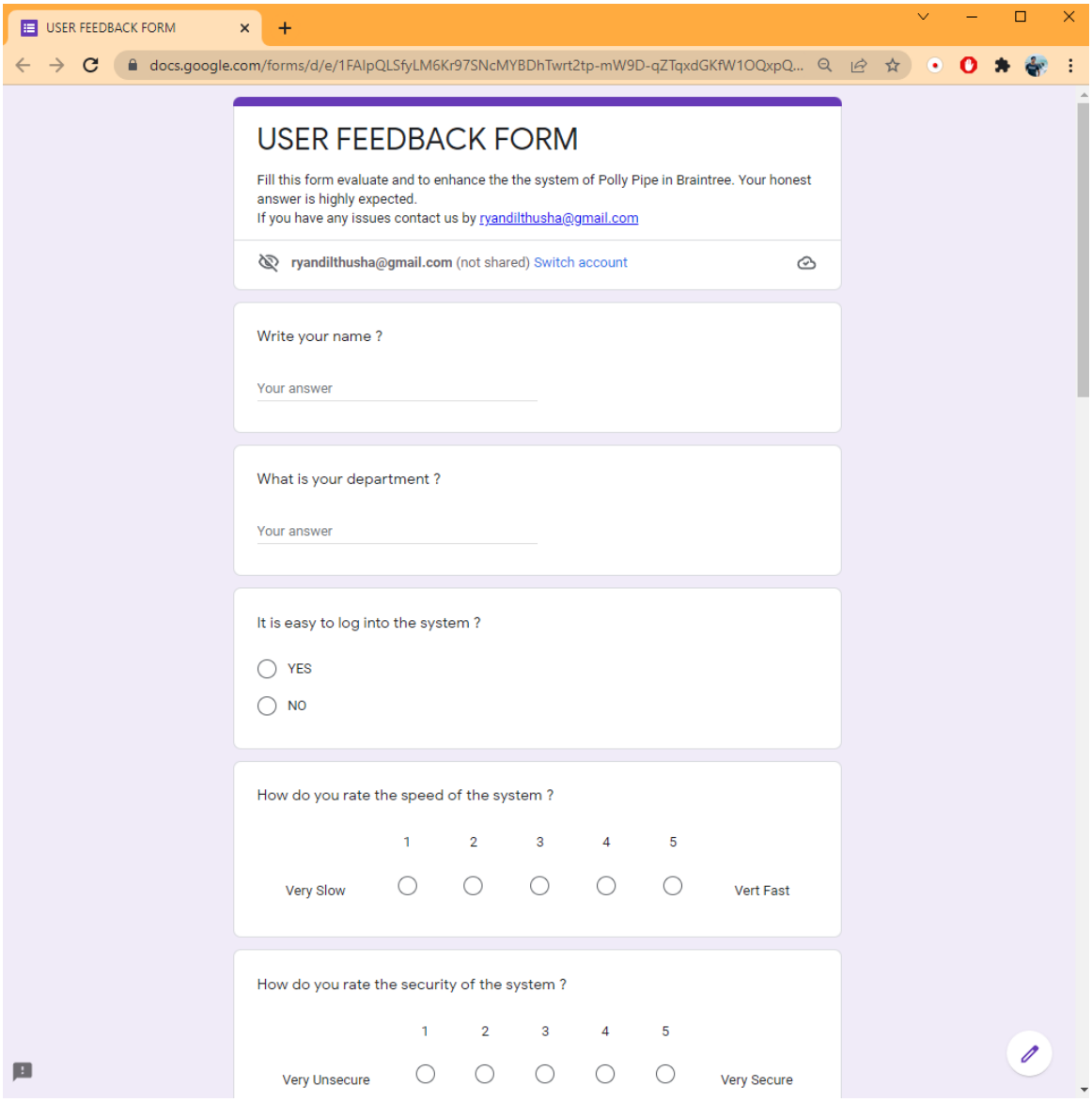


Figure 3. 1 Feedback form part 1

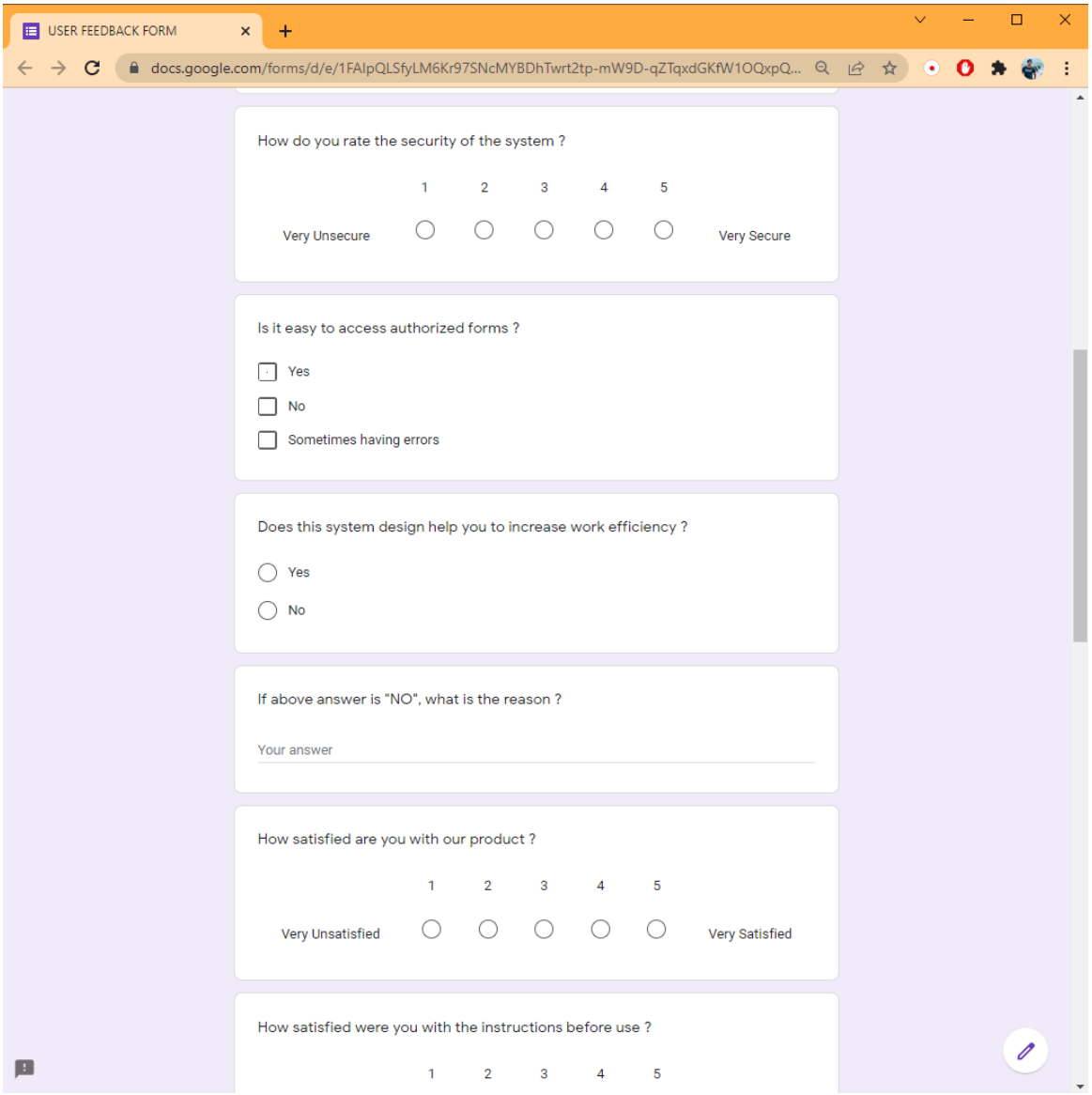


Figure 3. 2 Feedback form part 2

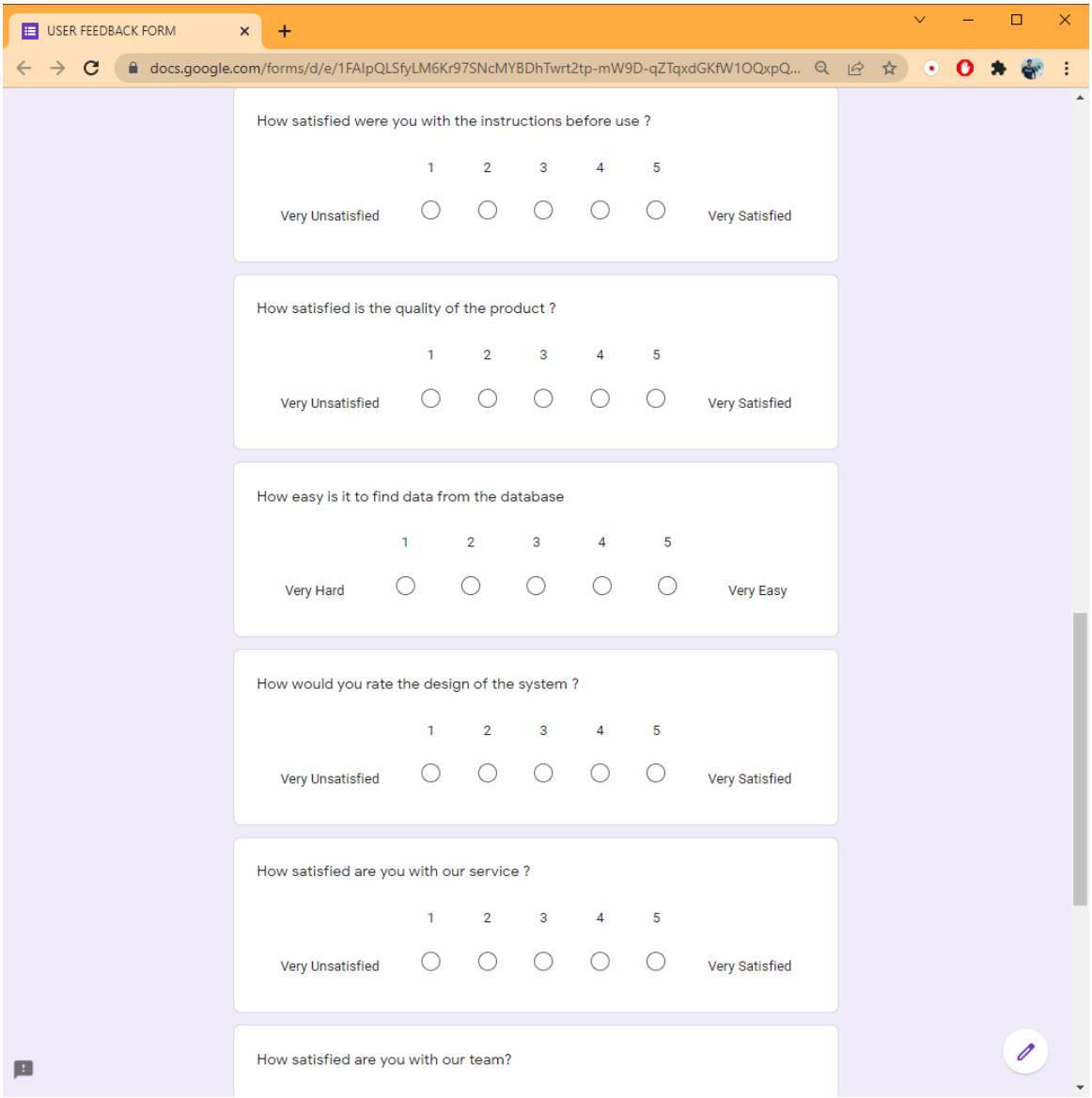


Figure 3. 3 Feedback form part 3

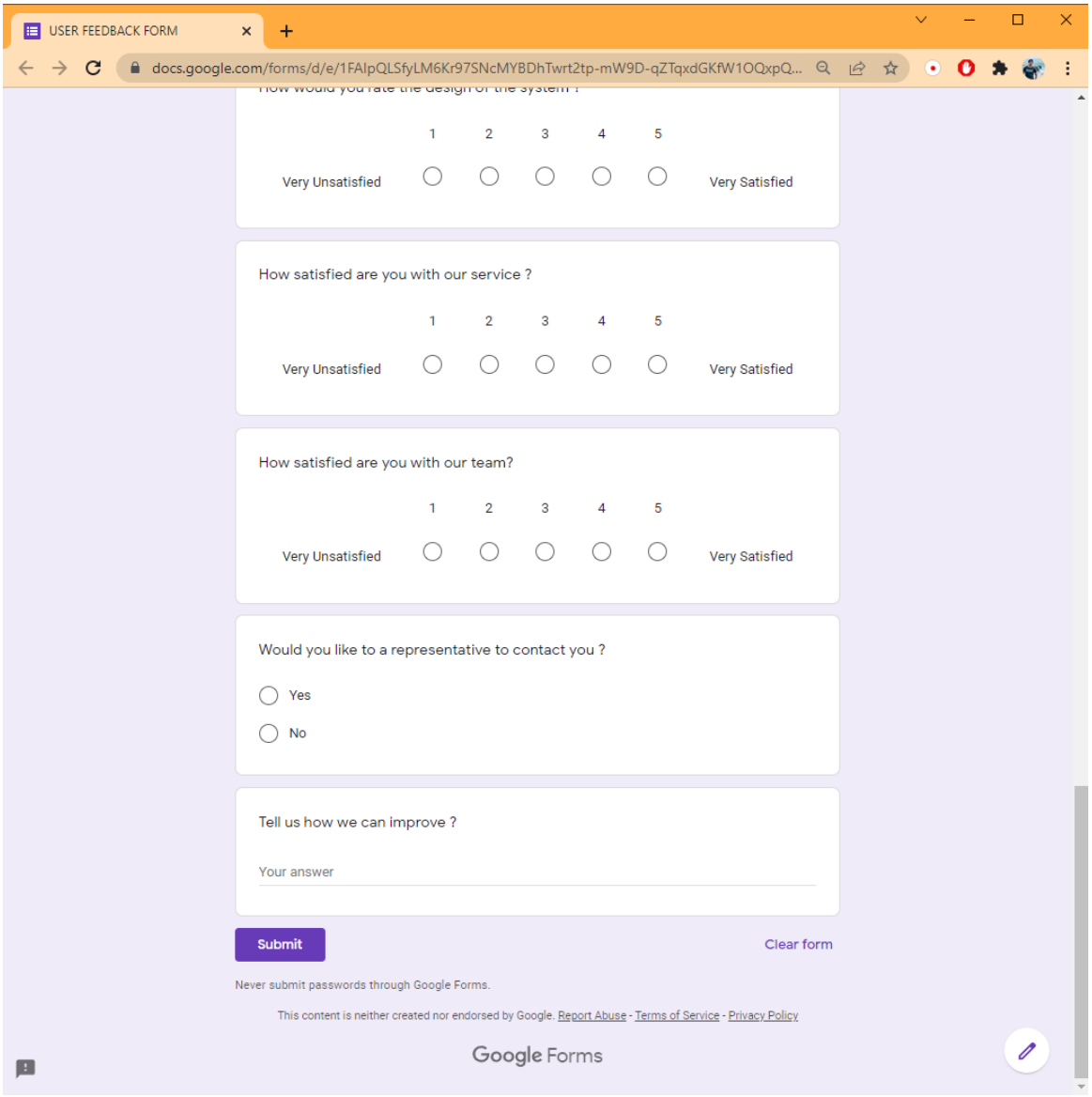


Figure 3. 4 Feedback form part 4

3.2.2 How we can improve the feedback collection form

Customer feedback is crucial since it acts as a direction for the company's future development. Companies are always interested in learning what they're doing correctly and wrong in the perspective of their customers. Positive solutions can be found in both the good and the poor, making it easier to change and adapt the client experience over time. In a brief, feedback is how we ensure that our community is at the center of everything we do.

1) Using simple images for customer rating scores

When rating a characteristic of something it is easy to rate it by selecting simple images rather than reading a numeral score. And it is very attractive way of keep customer adhere to the feedback form with an interest.

Figure 3. 5 Using simple images for customer rating scores

2) Making a quick survey

However, people are more likely to complete our survey if it is brief. That doesn't mean we have to compromise on the survey's quality. All we have to do now is ask the correct questions and ask for a quick response. That demanding response should have the potential to lead to a great, in-depth response.

3) Aim for a beautiful design that is also simple to use.

On whatever device, the feedback form should be professional, tidy, and easy to navigate. Consider including a corporate logo and using company colors to make it more attractive. A basic, tasteful design will always provide an excellent first impression, and it makes the survey more interesting to complete.

4) Side feedback button at the edge of the system

We can adjust the side button, which sits at the system's edge, waiting for a consumer to click it, which will launch the feedback form and prompt them to take action.

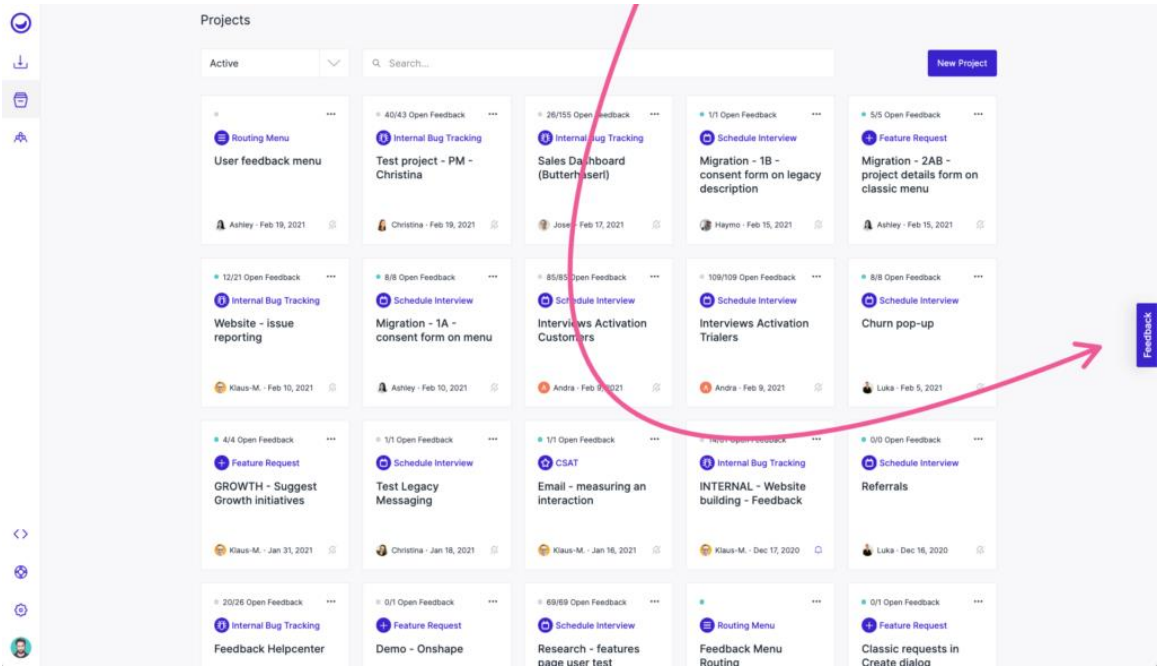


Figure 3. 6 Side feedback button at the edge of the system

5) Instant feedback from cooperate website

We can get fast consumer feedback without asking any questions using an embeddable on-site widget.

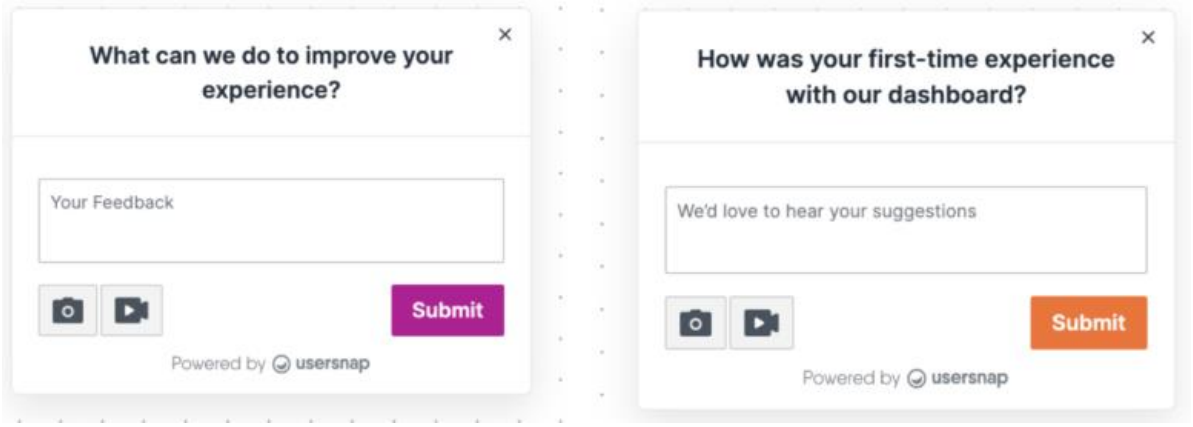


Figure 3. 7 Instant feedback sample from cooperate website

Activity 4

4.1 Produce Technical documentation and User documentation for developed System

4.1.1 Why software documentation

All written documentation and resources dealing with the development and use of a software product. Whether developed by a small team or a large business, all software development products require some form of documentation. Documentation exists to describe product functionality, organize project-related data, and allow stakeholders and developers to discuss any significant questions that arise. The major goal of good documentation is to make sure that developers and stakeholders are on the same page in order to achieve the project's goals.

There are two basic types of documentation for software products.

1. User Documentation - User documentation refers to documents written primarily for product end-users and system administrators. Tutorials, user guides, troubleshooting manuals, installation, and reference manuals are all examples of user documentation.
2. Technical Documentation - This document contains information about the system as well as its technological components. It includes documents such as requirements, design decisions, architecture descriptions, program source code, and help guides.

4.2 User Documentation

4.2.1 What is user documentation

End-user documentation, often known as user documentation, refers to the documentation provided to end-users for a product or service. Its purpose is to provide product information. Also known as "user help," this is a term that is frequently used. It could be based on a delivered product or service, or a complete end-to-end project documentation that directly benefits the end-user.

Technically, a useful user document should have all relevant documentation and necessities for the whole product life cycle. The user documentation is part of the end-to-end product that is supplied to the consumer.

4.2.2 Types of User Documentation

1) Description Document :- The Product Description Document provides a full summary of the product, including all of the services it provides. End-users read this page and decide whether or not this is the product they are looking for.

2) Installation and Setup :- Provides thorough instructions on how to install and set up the product, as well as how to use it.

3) Product or User Manual :- It explains the product's basic functionalities with illustrations. It contains all of the 'How-to' instructions for using the product on a regular basis. We can write a fantastic Product handbook in no time.

4.2.3 User Documentation

Contents

1) System Description
2) System References
3) System Features
4) System Requirements
5) User Interfaces

1) System Description

This system is made for both Administrators and Sales Representatives to use. Both of them having different login credentials to access the system hence it enhances the security of the system For admin this system is providing to store details of a customer base and employee details. And also, with this system admin can classify the employee types separately. Not only that, but also admin can store details about your facilities which expecting to install and provide. Admin can do that easily by looking at the stored data in the database.

But for Sales Representatives they only have access to install a facility with stored data by admins and adding new customers to database. Sales representative can give installment data by easily by looking at the admin stored data in the database.

2) System References

We've used latest platforms, plugins and technologies when creating this system

- Microsoft Visual Studio 2019
- Microsoft SQL Server Management Studio 2018

3) System Features

This system is coming with small file containing the database base, source code along with the setup. User can easily install the setup by double clicking the “SETUP.exe” file to any location user want.

4) System Requirements

This system can be installed to any basic computers which having Windows Operating System.

5) User Interfaces

Login Form →

After loading the system 1st thing all users can see is the login form. User must select the job role they are having in the combo box. Then user should give login credentials in the Username and Password fields which system developers have given separately. After entering login credentials user can simply enter to the system by clicking Login button. If user want to erase given data, user can do it by clicking on the clear button. If user want to exit, he/she can do it by clicking Exit button.

Figure 4. 1 Login Form

Customer Form →

- Both Admin and Sales Representative have the access to log into Customers Form.
- User can't edit the Customer ID field because the ID it's automatically given by the system.
- User can simply save new records by filling the text box fields and hitting the Save button.
- When updating a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. User can make changes on the fields then by hitting the Edit button the record will be updated.
- When deleting a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. Then by hitting the Delete button the record will be deleted.
- To clear the text box fields user can hit on the Clear button.
- User can exit from the system by click on the logout button.
- User can switch to different form by left hand panel buttons.

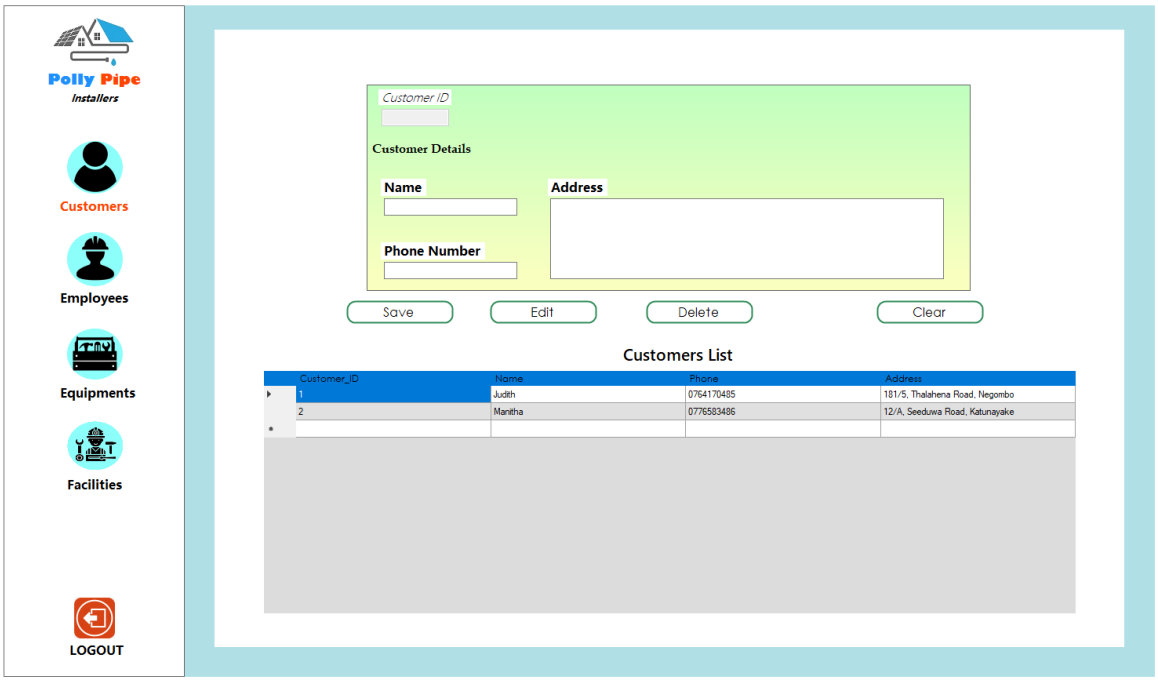


Figure 4. 2 Customer Form (For Admin)

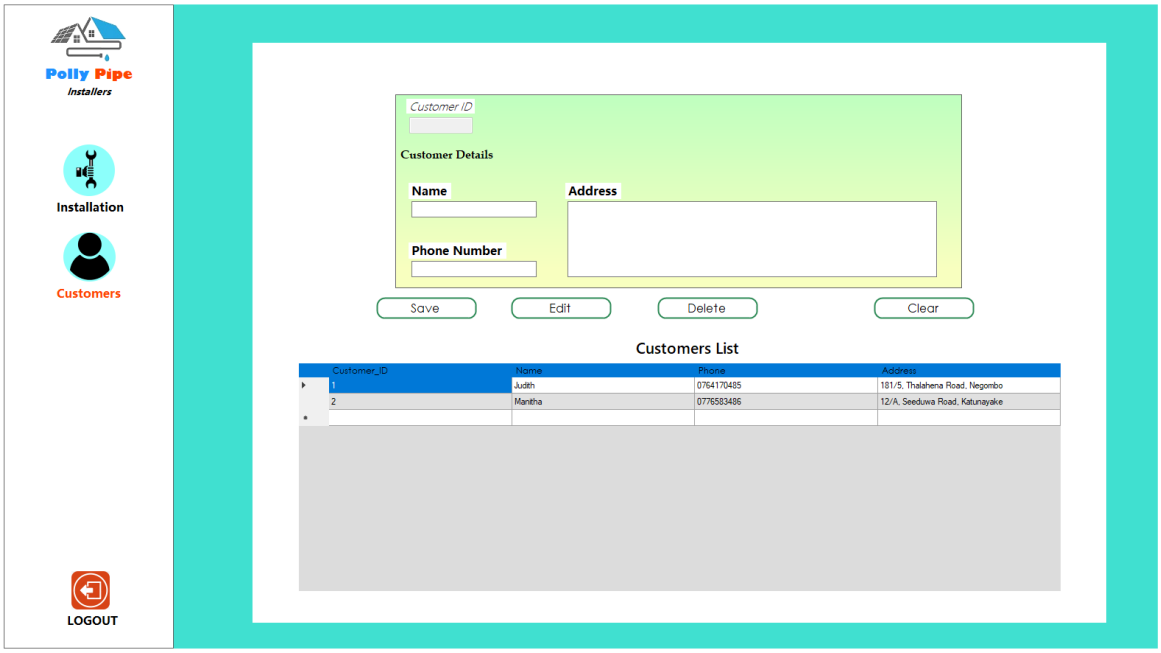


Figure 4. 3 Customer Form (For Rep)

Employee Form →

- Only admin has the access to log into Employee Form.
- User can't edit the Employee ID field because the ID it's automatically given by the system.
- User can fill the Employee Type ID text box by selecting the Employee Type ID from combo box or clicking on the Employee Type DataGrid view relevant cell.
- If user want to make changes of Employee Type details, he/she can log into to relevant form by clicking on Employee Types button at the button of the DataGrid view.
- User can simply save new records by filling the text box fields and hitting the Save button.
- When updating a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. User can make changes on the fields then by hitting the Edit button the record will be updated.
- When deleting a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. Then by hitting the Delete button the record will be deleted.
- To clear the text box fields user can hit on the Clear button.
- User can exit from the system by click on the logout button.
- User can switch to different form by left hand panel buttons.

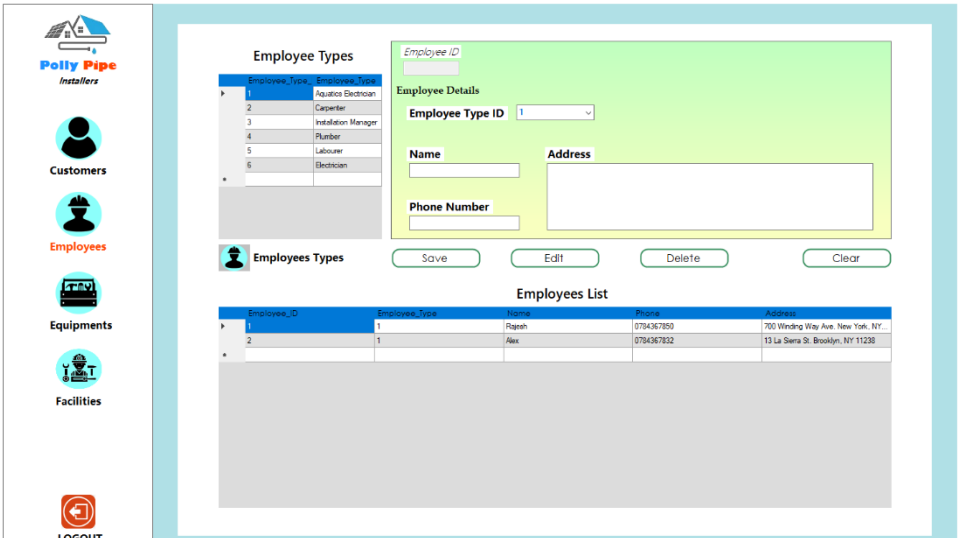


Figure 4. 4 Employee Form

Employee Types Form →

- Only admin has the access to log into Employee Types Form.
- User can't edit the Employee Type ID field because the ID it's automatically given by the system.
- User can simply save new records by filling the text box fields and hitting the Save button.
- When updating a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. User can make changes on the fields then by hitting the Edit button the record will be updated.
- When deleting a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. Then by hitting the Delete button the record will be deleted.
- To clear the text box fields user can hit on the Clear button.
- User can exit from the system by click on the logout button.

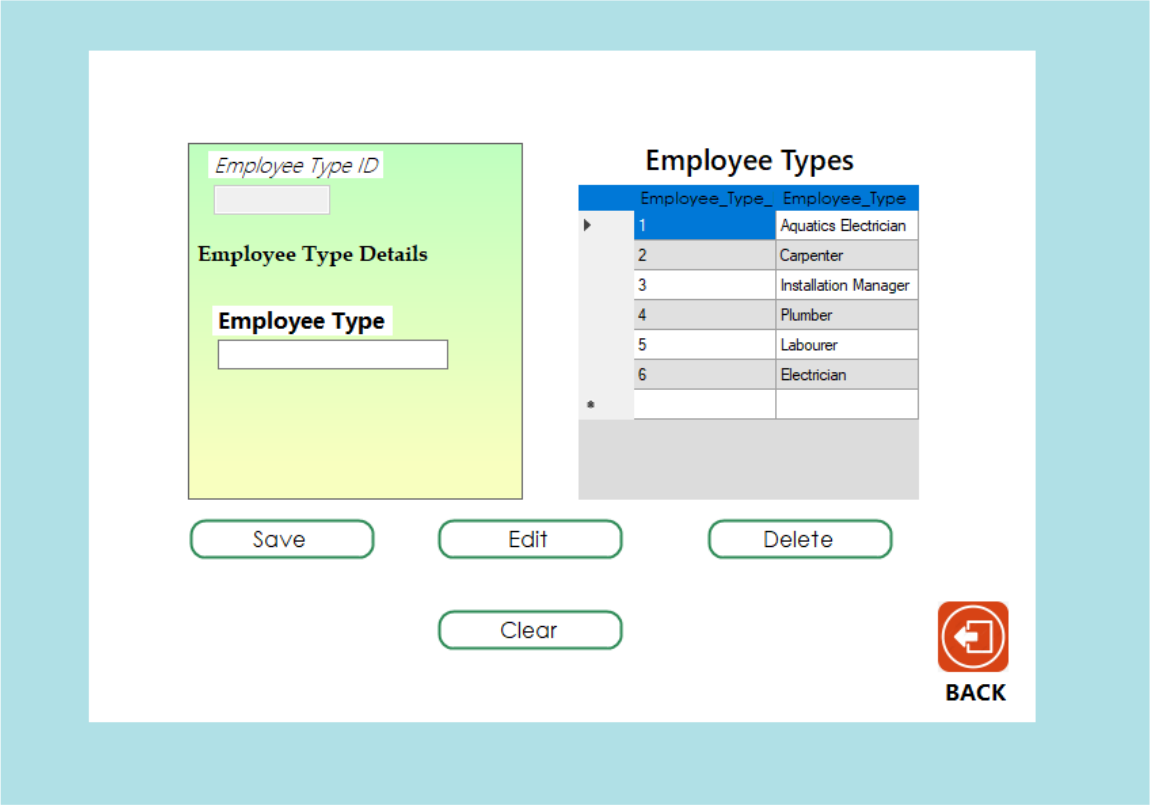


Figure 4. 5 Employee Types Form

Equipment Form →

- Only admin has the access to log into Equipment Form.
- User can't edit the Customer ID field because the ID it's automatically given by the system.
- User can simply save new records by filling the text box fields and hitting the Save button.
- When updating a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. User can make changes on the fields then by hitting the Edit button the record will be updated.
- When deleting a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. Then by hitting the Delete button the record will be deleted.
- To clear the text box fields user can hit on the Clear button.
- User can exit from the system by click on the logout button.
- User can switch to different form by left hand panel buttons.

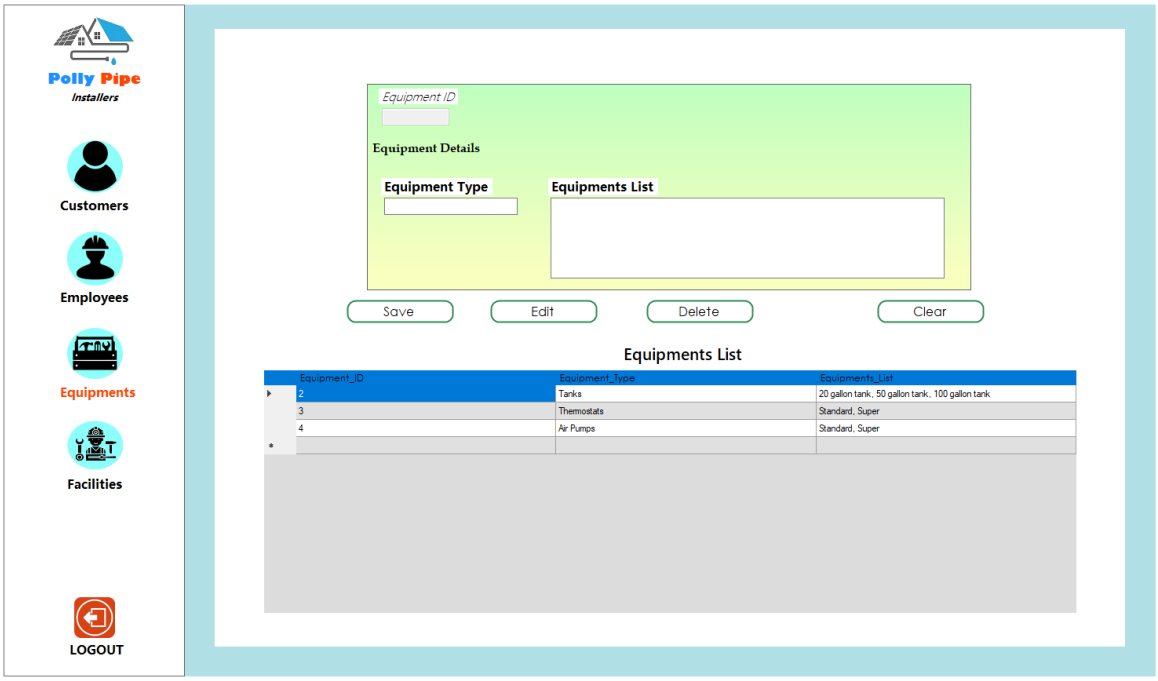


Figure 4. 6 Equipment Form

Facility Form →

- Only admin has the access to log into Facility Form.
- User can't edit the Facility ID field because the ID it's automatically given by the system.
- User can fill the Employee Type ID text box by selecting the Employee Type ID from combo box or clicking on the Employee Type DataGrid view relevant cell.
- User can fill the Equipment Type ID text box by selecting the Equipment Type ID from combo box or clicking on the Equipment Type DataGrid view relevant cell.
- User can simply save new records by filling the text box fields and hitting the Save button.
- When updating a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. User can make changes on the fields then by hitting the Edit button the record will be updated.
- When deleting a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. Then by hitting the Delete button the record will be deleted.
- To clear the text box fields user can hit on the Clear button.
- User can exit from the system by click on the logout button.
- User can switch to different form by left hand panel buttons.

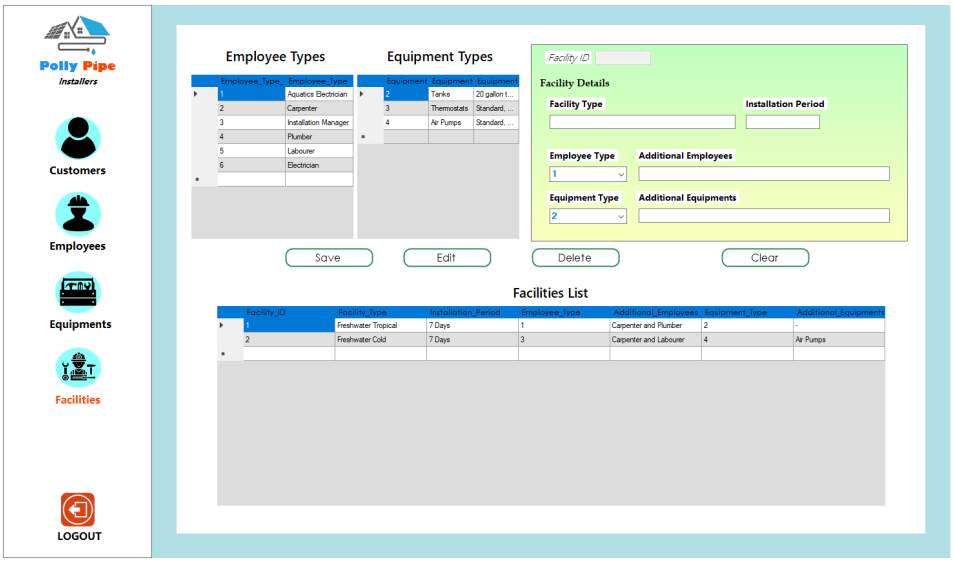


Figure 4. 7 Facility Form

Installation Form →

- Only admin has the access to log into Installation Form.
- User can't edit the Installation ID field because the ID it's automatically given by the system.
- User can fill the Facility ID text box by selecting the Facility ID from combo box or clicking on the Facility DataGrid view relevant cell.
- User can fill the Customer ID text box by selecting the Customer ID from combo box or clicking on the Customer DataGrid view relevant cell.
- User can simply save new records by filling the text box fields and hitting the Save button.
- When updating a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. User can make changes on the fields then by hitting the Edit button the record will be updated.
- When deleting a specific record, user should click on the selected record in the data grid view. By clicking it, the text box fields automatically getting filled with the relevant data. Then by hitting the Delete button the record will be deleted.
- To clear the text box fields user can hit on the Clear button.
- User can exit from the system by click on the logout button.
- User can switch to different form by left hand panel buttons.

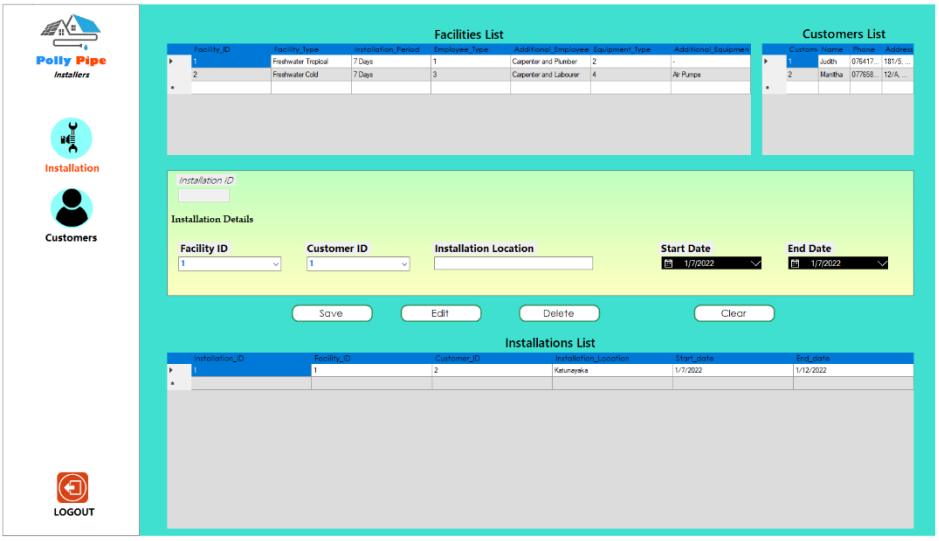


Figure 4. 8 Installation Form

4.3 Technical Documentation

Any document that explains how to use, operate, create, or architect a product is referred to as technical documentation. Consider it a "how to" guidebook for users such as new recruits, admins, and anybody else who needs to know how to operate the product. Technical documentation is more than just information capture. It's all about making it easy to read, understand, and use.

Technical documentation is important for end users because it enables them to use a product effectively. This is especially useful for the technical documentation provider because it reduces customer support hours and leads to happier users who can troubleshoot and get answers to their own queries. Technical documentation is essential from an internal point of view since it provides individuals with the information they need to work efficiently on a product. Products aren't always able to talk for themselves. That is why we want technical documentation to provide us with all of the information we require.

Contents

1) Overview
2) Design and Architecture
3) Source Code
4) System Requirements
5) Future Enhancements

4.3.1 Overview

This document is made it help technical users to understand the system build. And also, this can be used for future maintenance requirements and need. For well performance of the system and for the system updates as require, technical users can use and refer this documentation.

4.3.2 Design and Architecture

ER Diagram →

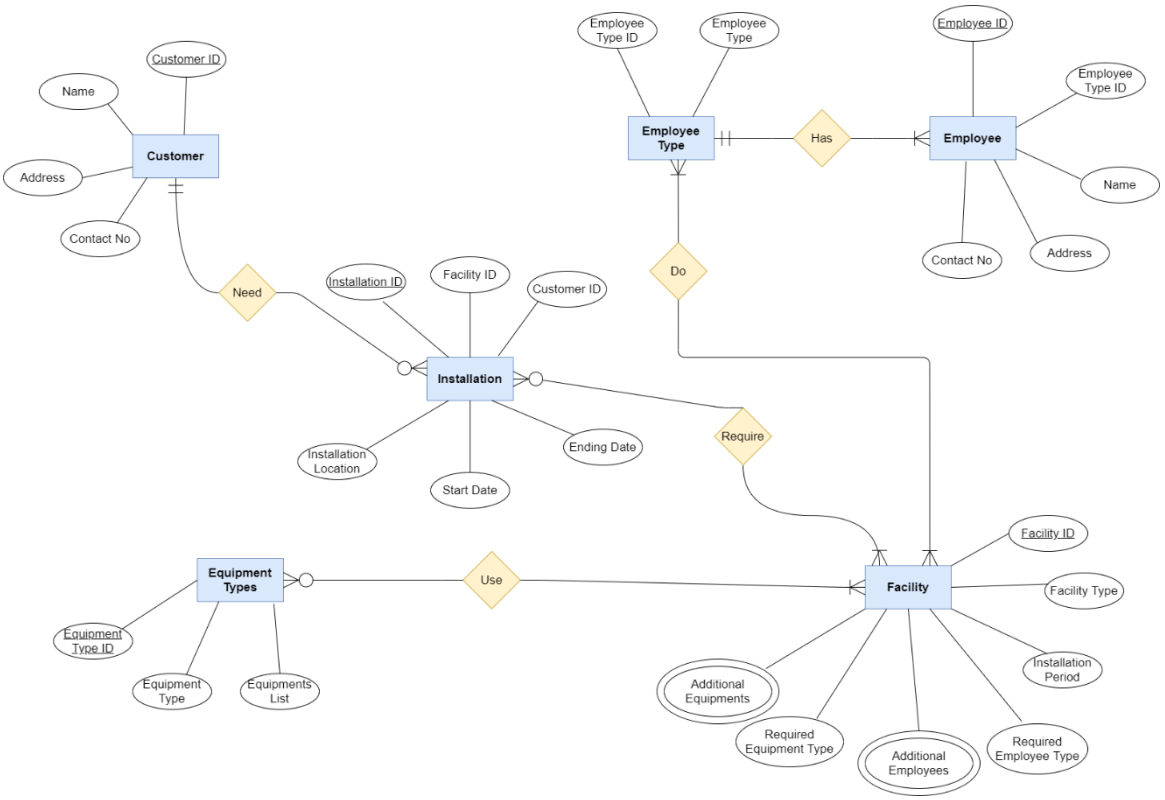


Figure 4. 9 ER Diagram

Logical Database →

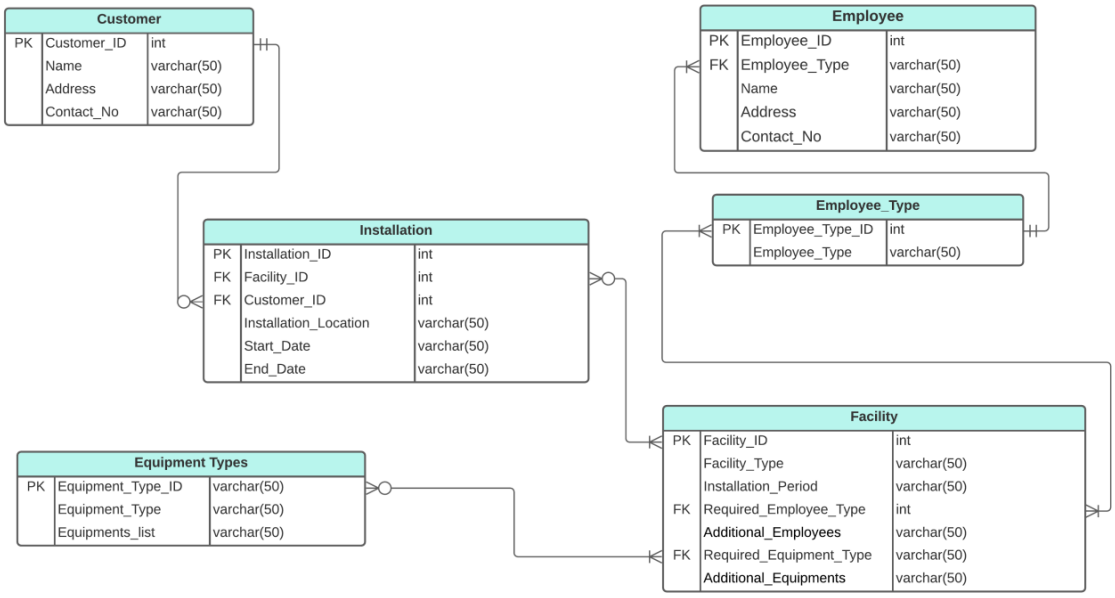


Figure 4. 10 Logical Database

Class Diagram →

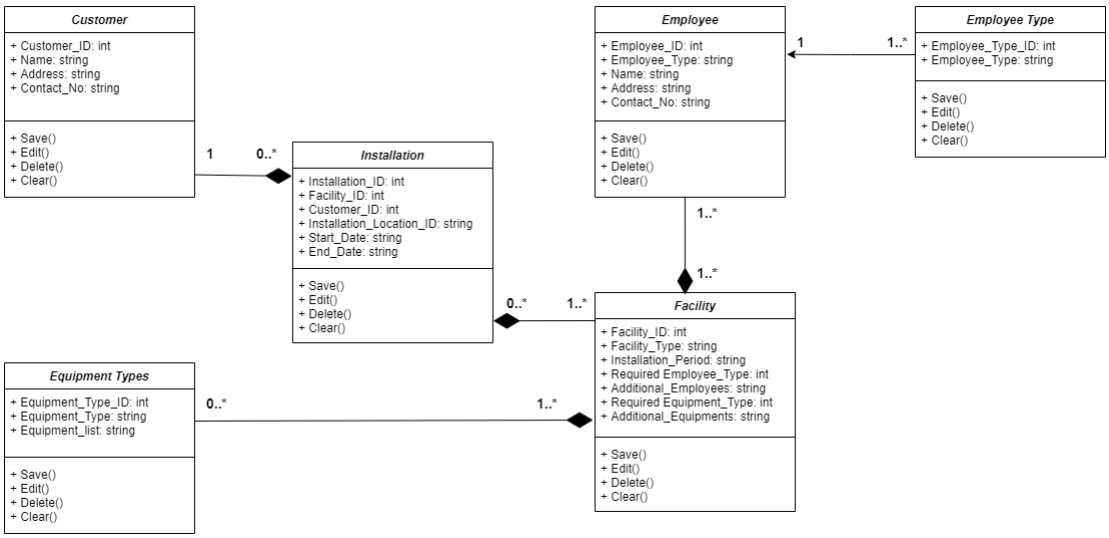


Figure 4. 11 Class Diagram

Use Case Diagram →

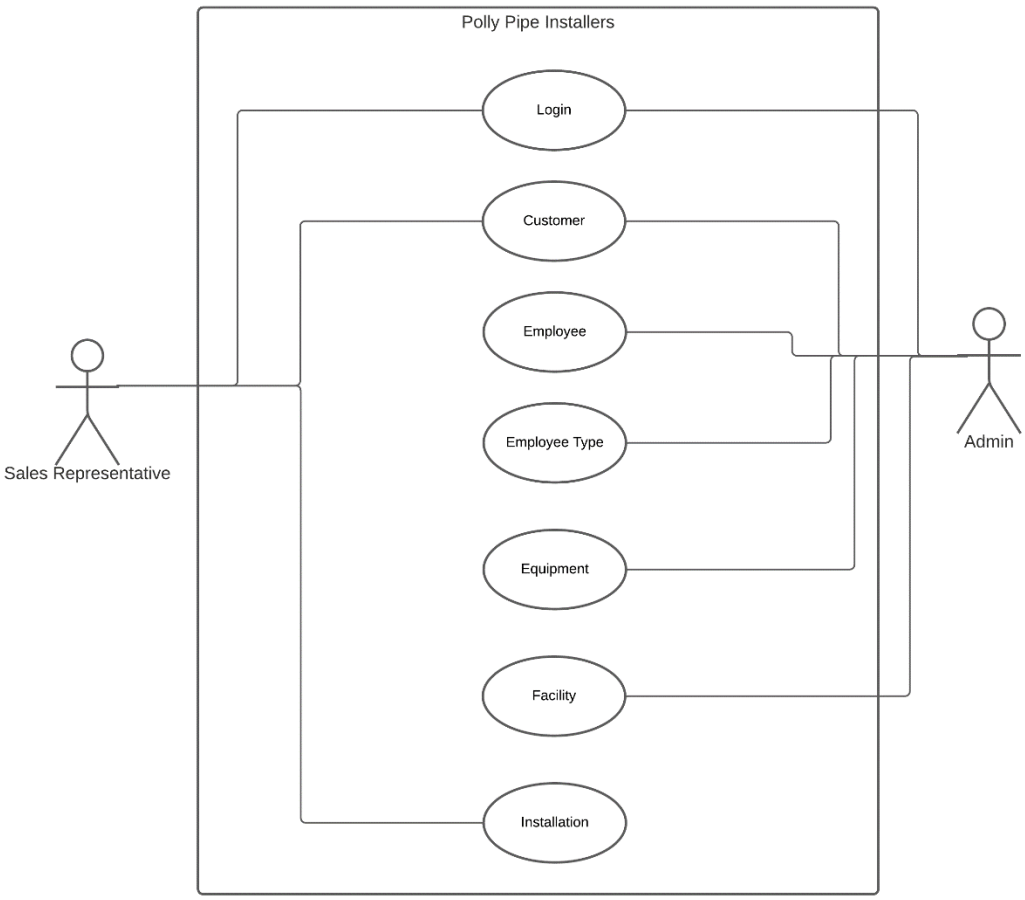


Figure 4. 12 Use Case Diagram

DFD Level 0 →

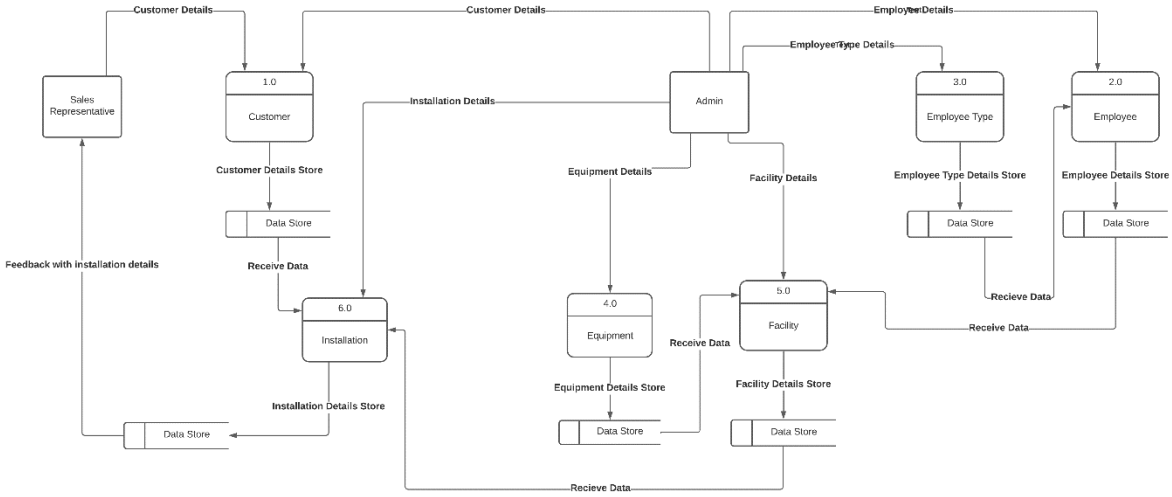


Figure 4. 13 DFD Level 0

DFD Level 1 →

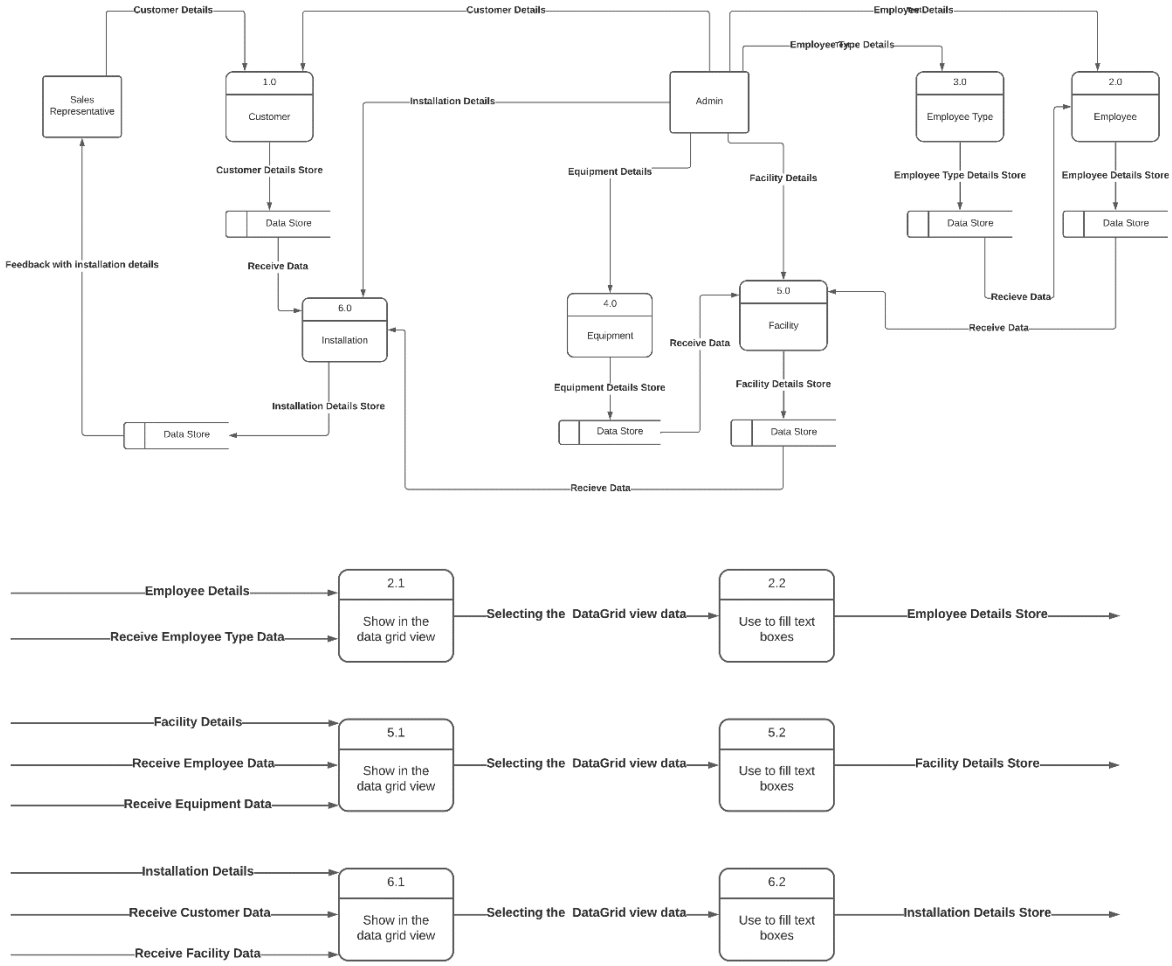


Figure 4. 14 DFD Level 1

Flow Chart for Login Button Function →

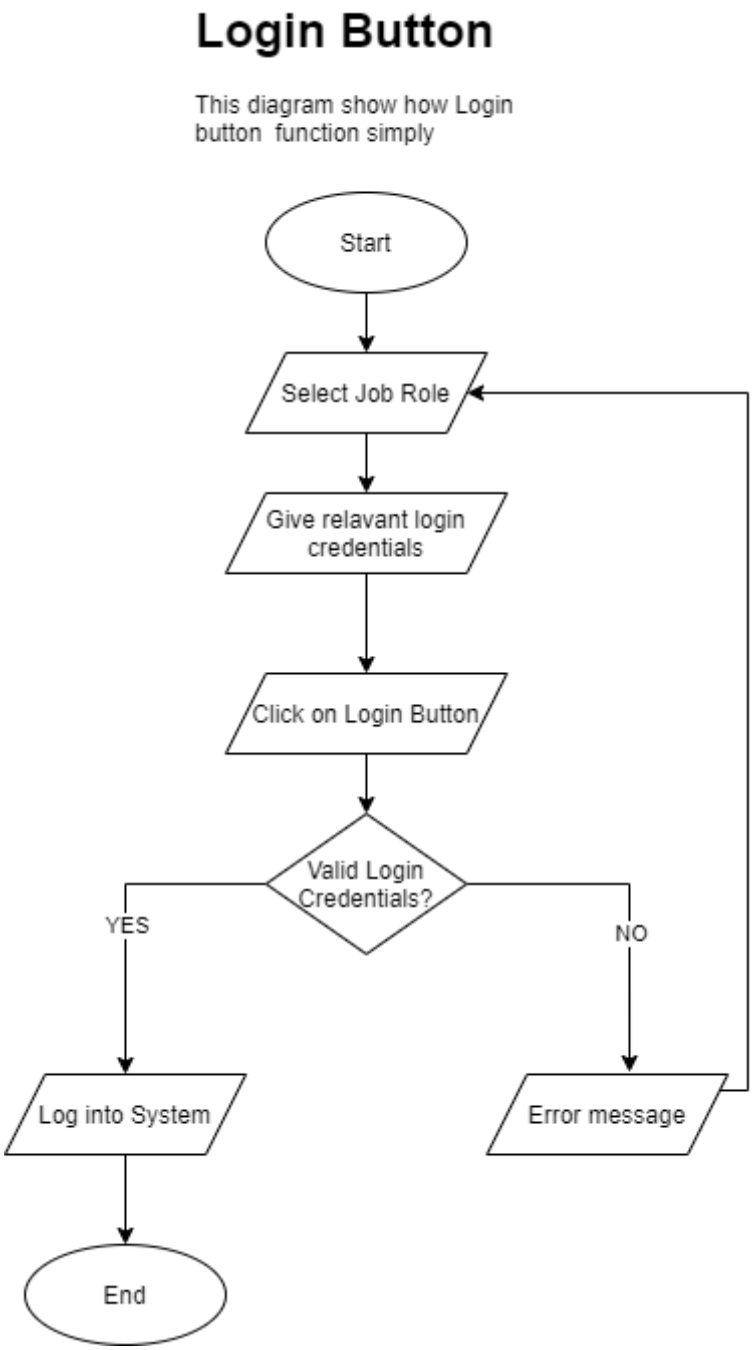


Figure 4. 15 Flow Chart for Login Button Function

Flow Chart for Save Button Function →

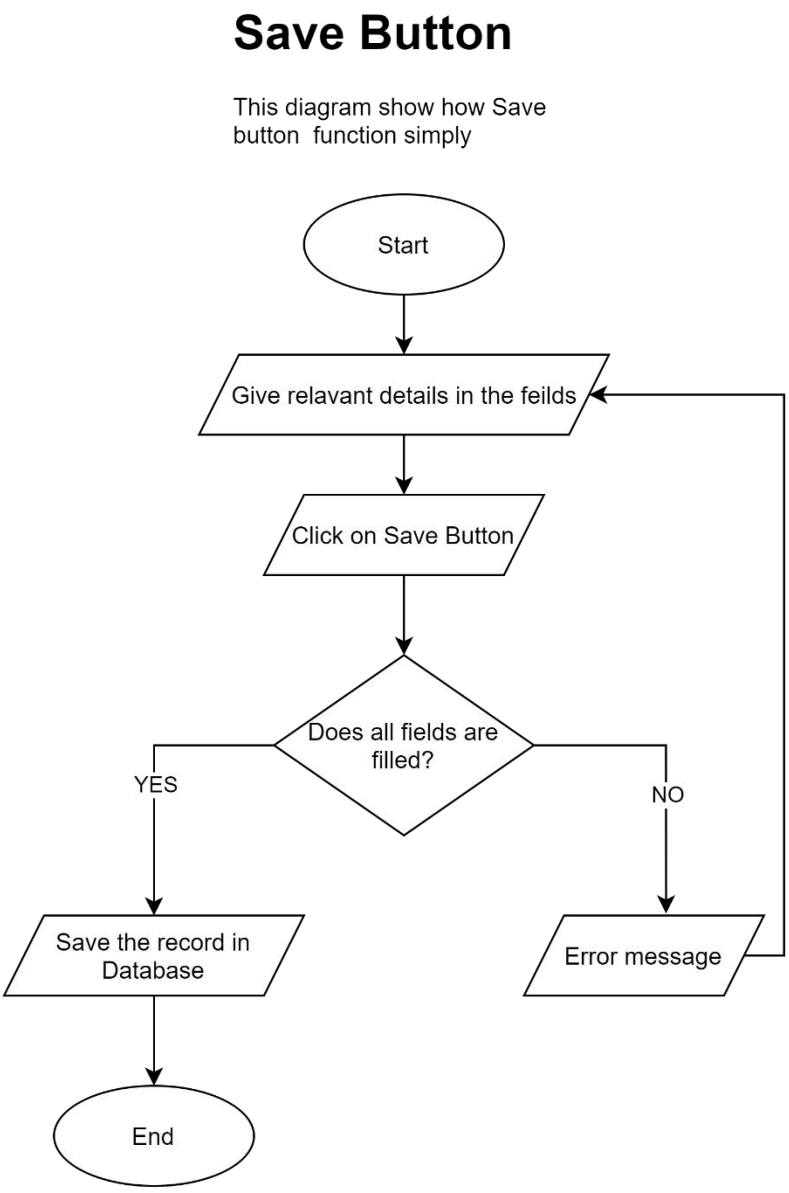


Figure 4. 16 Flow Chart for Save Button Function

Flow Chart for Edit Button Function →

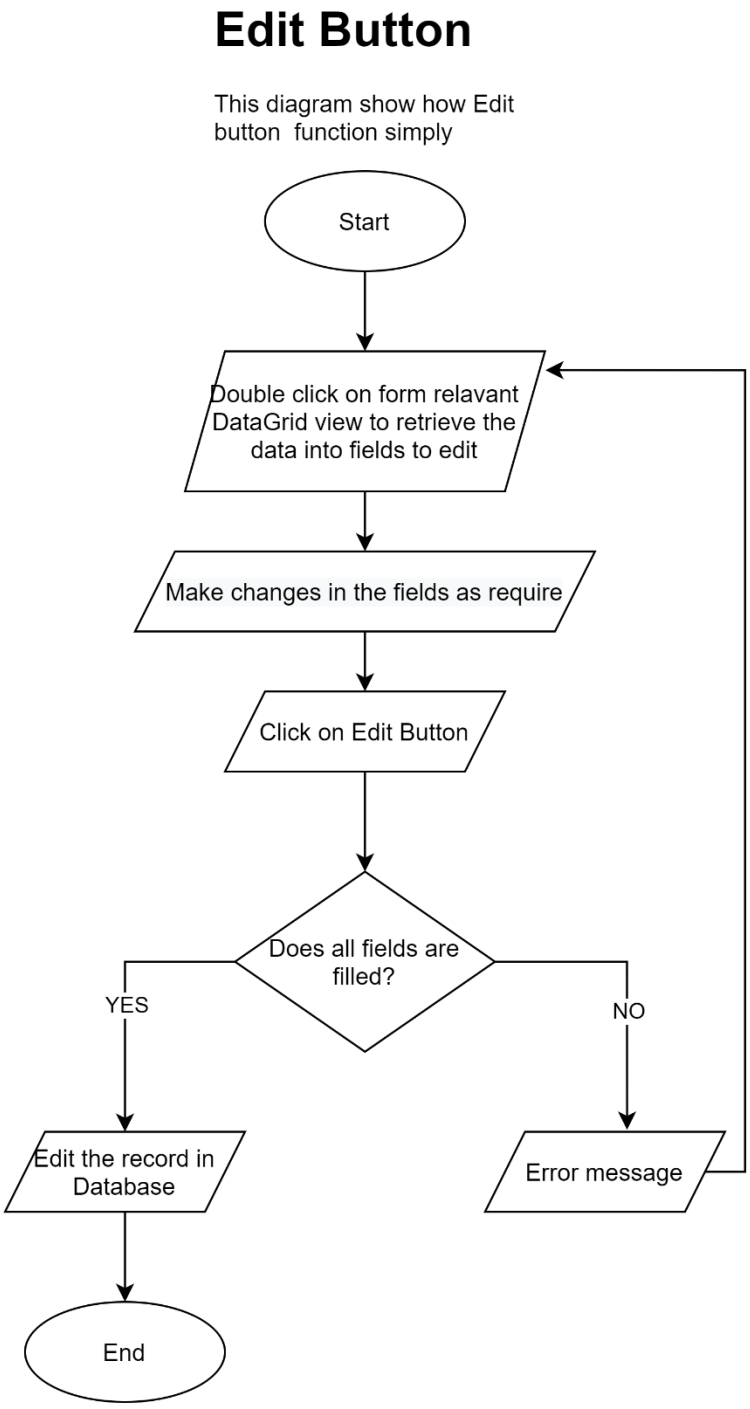


Figure 4. 17 Flow Chart for Edit Button Function

Flow Chart for Delete Button Function →

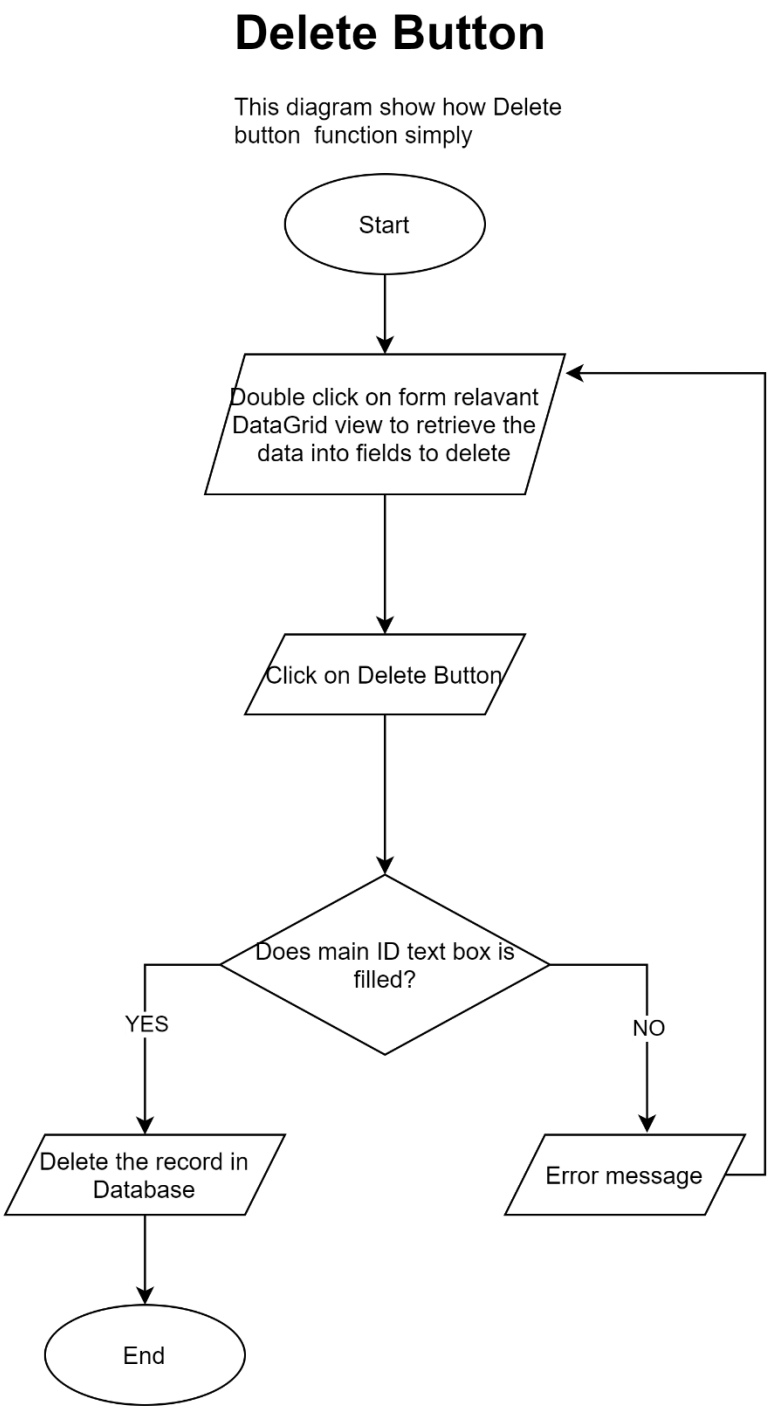


Figure 4. 18 Flow Chart for Delete Button Function

Flow Chart for Clear Button Function →

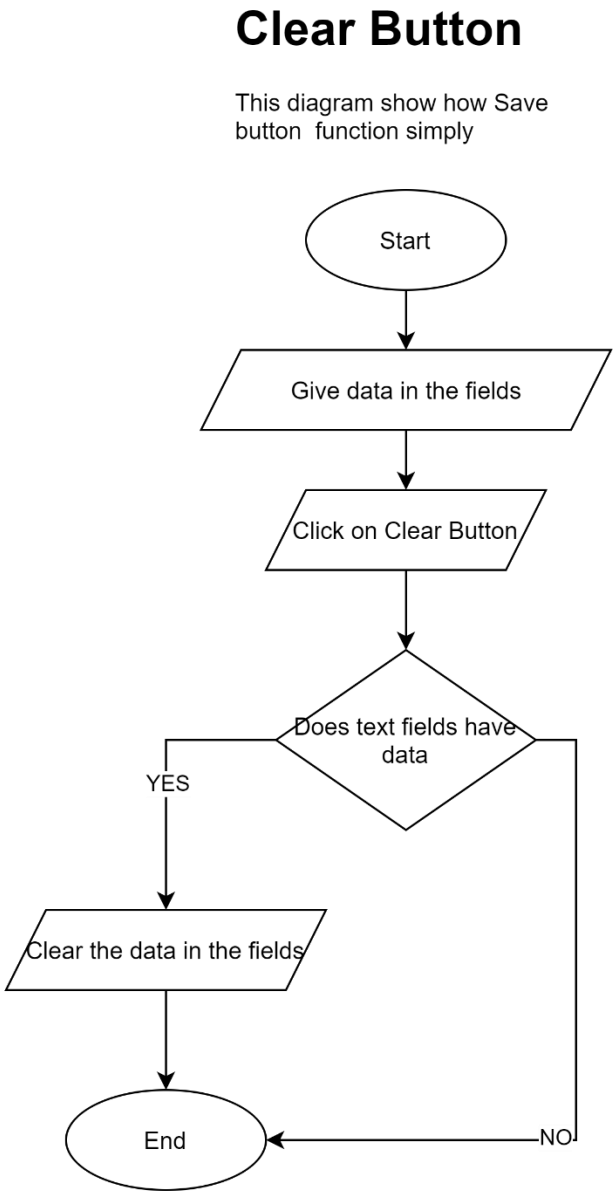
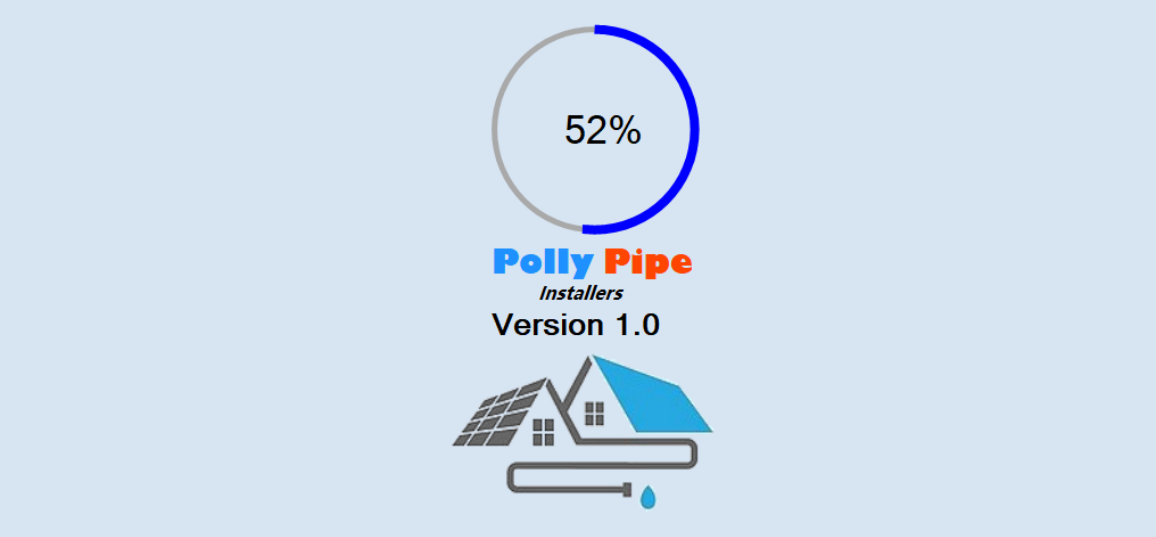


Figure 4. 19 Flow Chart for Clear Button Function

4.3.3 Source Code

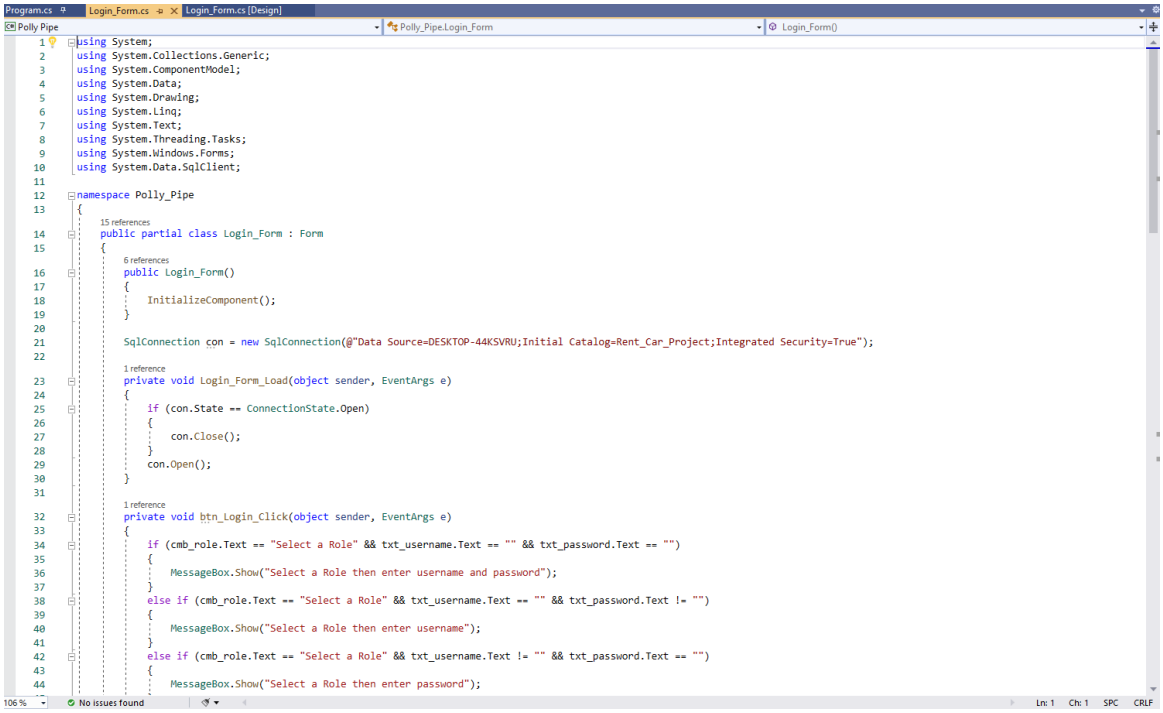
Splash Screen Form →



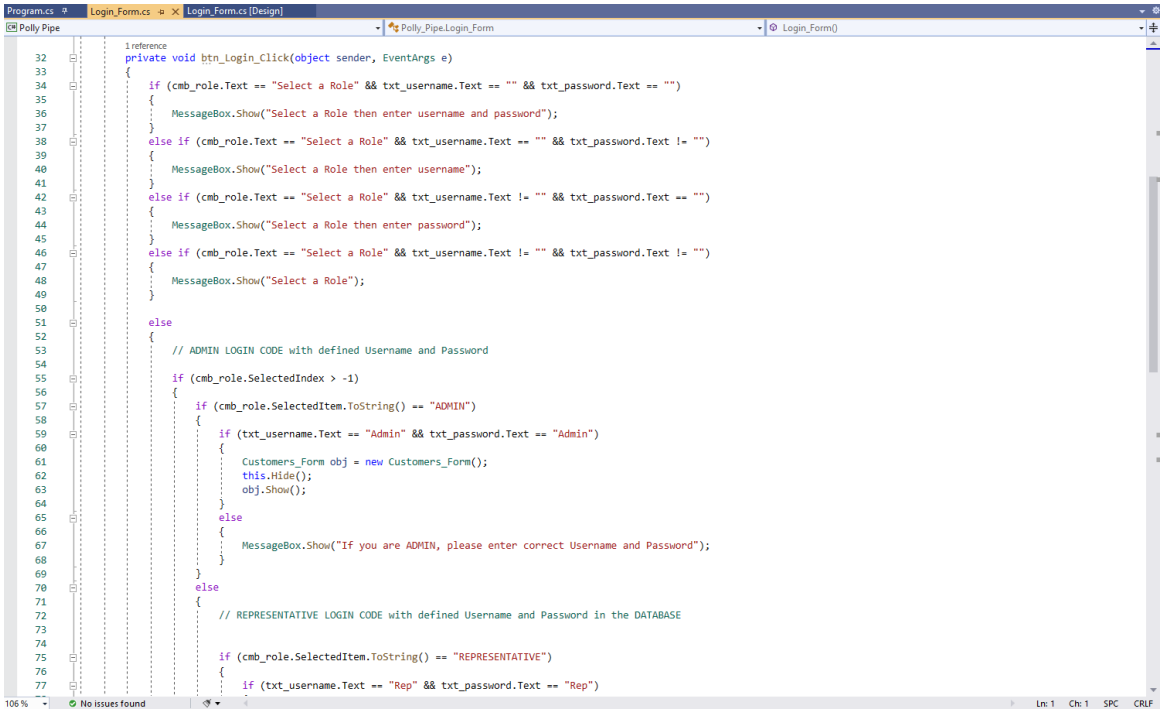

```
Program.cs | Splash_Screen_Form.cs | Splash_Screen_Form.cs [Design]
Polly Pipe | Polly_Pipe.Splash_Screen_Form | startpoint

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Polly_Pipe
{
    4 references
    public partial class Splash_Screen_Form : Form
    {
        int startpoint = 0;

        1 reference
        public Splash_Screen_Form()
        {
            InitializeComponent();
        }

        1 reference
        private void timer1_Tick(object sender, EventArgs e)
        {
            bunifuCircleProgressbar1.Value = startpoint;
            startpoint += 10;

            if (bunifuCircleProgressbar1.Value == 100)
            {
                timer1.Stop();

                Login_Form obj = new Login_Form();
                this.Hide();
                obj.Show();
            }
        }

        1 reference
        private void Splash_Screen_Form_Load(object sender, EventArgs e)
        {
            timer1.Start();
        }
    }
}
```

Login Form →

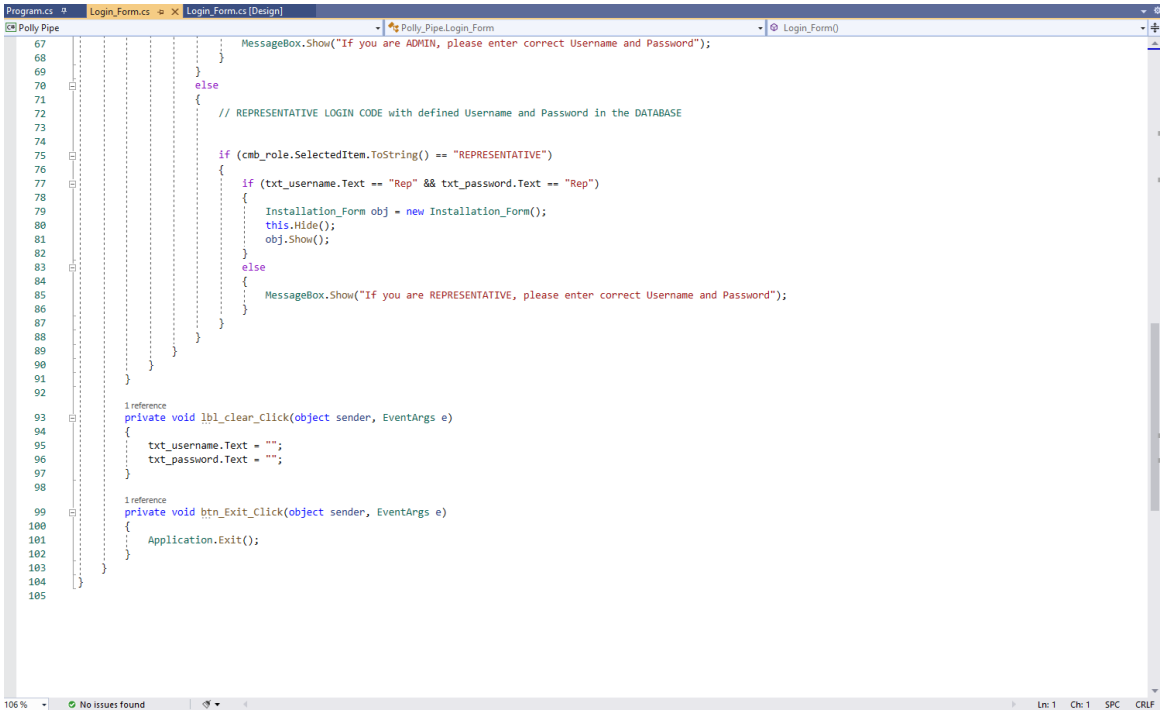

Polly Pipe
Installers
ADMIN
ADMIN
REPRESENTATIVE
Username
admin
Password
*****
Clear
Login
EXIT



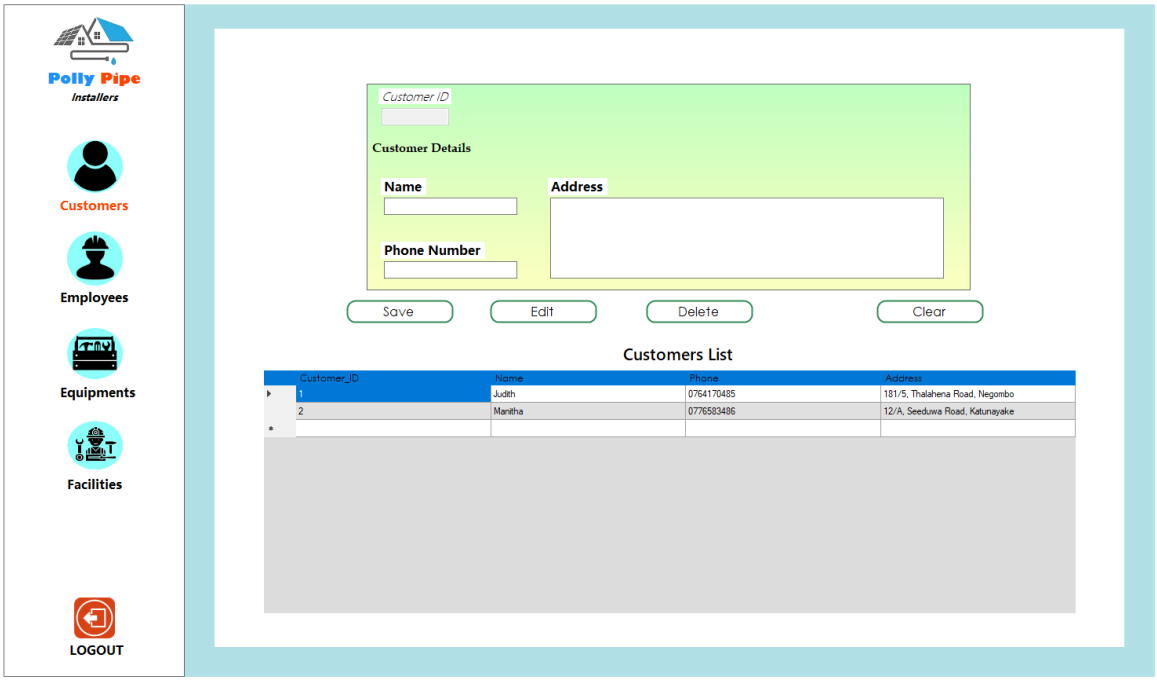
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    15 references
    public partial class Login_Form : Form
    {
        6 references
        public Login_Form()
        {
            InitializeComponent();
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Rent_Car_Project;Integrated Security=True");

        1 reference
        private void Login_Form_Load(object sender, EventArgs e)
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
            con.Open();
        }

        1 reference
        private void btn_Login_Click(object sender, EventArgs e)
        {
            if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter username and password");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text != "")
            {
                MessageBox.Show("Select a Role then enter username");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text != "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter password");
```



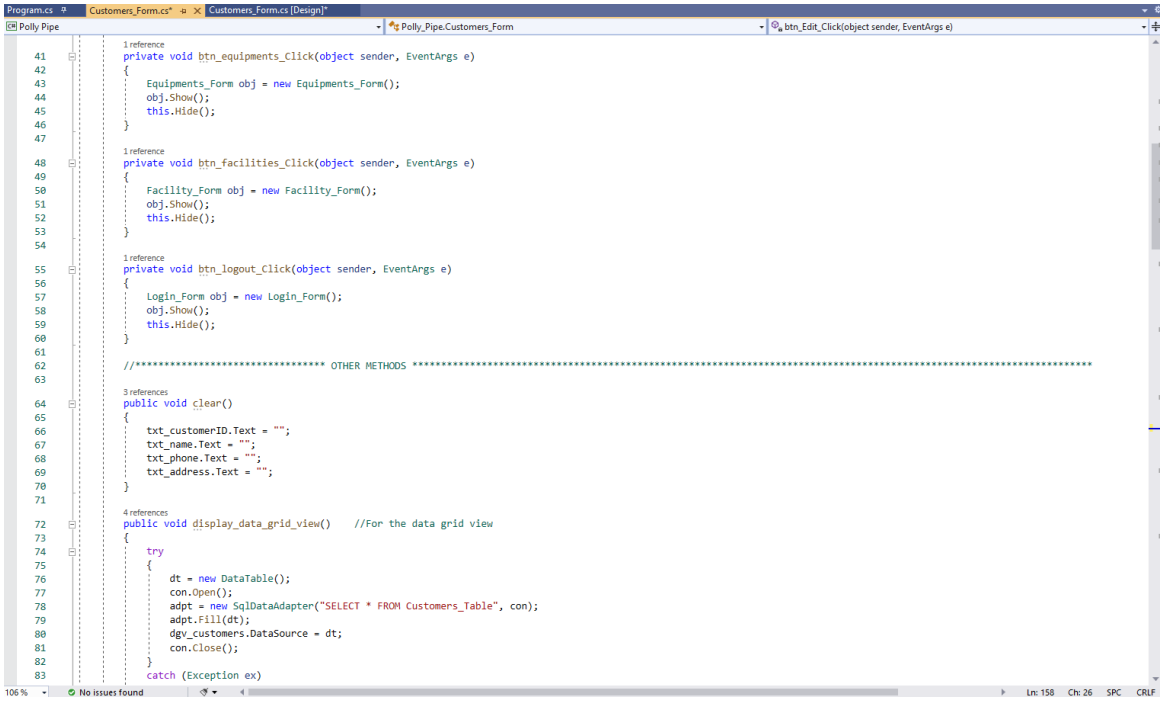
```
        1 reference
        private void btn_Login_Click(object sender, EventArgs e)
        {
            if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter username and password");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text == "" && txt_password.Text != "")
            {
                MessageBox.Show("Select a Role then enter username");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text != "" && txt_password.Text == "")
            {
                MessageBox.Show("Select a Role then enter password");
            }
            else if (cmb_role.Text == "Select a Role" && txt_username.Text != "" && txt_password.Text != "")
            {
                MessageBox.Show("Select a Role");
            }

            else
            {
                // ADMIN LOGIN CODE with defined Username and Password

                if (cmb_role.SelectedIndex > -1)
                {
                    if (cmb_role.SelectedItem.ToString() == "ADMIN")
                    {
                        if (txt_username.Text == "Admin" && txt_password.Text == "Admin")
                        {
                            Customers_Form obj = new Customers_Form();
                            this.Hide();
                            obj.Show();
                        }
                        else
                        {
                            MessageBox.Show("If you are ADMIN, please enter correct Username and Password");
                        }
                    }
                    else
                    {
                        // REPRESENTATIVE LOGIN CODE with defined Username and Password in the DATABASE


                        if (cmb_role.SelectedItem.ToString() == "REPRESENTATIVE")
                        {
                            if (txt_username.Text == "Rep" && txt_password.Text == "Rep")
```


Program.cs
Login_Form.cs
Login_Form.cs [Design]
Polly Pipe
Polly_Pipe.Login_Form
Login_Form()
MessageBox.Show("If you are ADMIN, please enter correct Username and Password");
}
}
else
{
// REPRESENTATIVE LOGIN CODE with defined Username and Password in the DATABASE
if (cmb_role.SelectedItem.ToString() == "REPRESENTATIVE")
{
if (txt_username.Text == "Rep" && txt_password.Text == "Rep")
{
Installation_Form obj = new Installation_Form();
this.Hide();
obj.Show();
}
else
{
MessageBox.Show("If you are REPRESENTATIVE, please enter correct Username and Password");
}
}
}
}
}
}
1 reference
private void lbl_clear_Click(object sender, EventArgs e)
{
txt_username.Text = "";
txt_password.Text = "";
}
1 reference
private void btn_Exit_Click(object sender, EventArgs e)
{
Application.Exit();
}
}
}
106 %
No issues found
Ln: 1
Ch: 1
SPC
CRLF

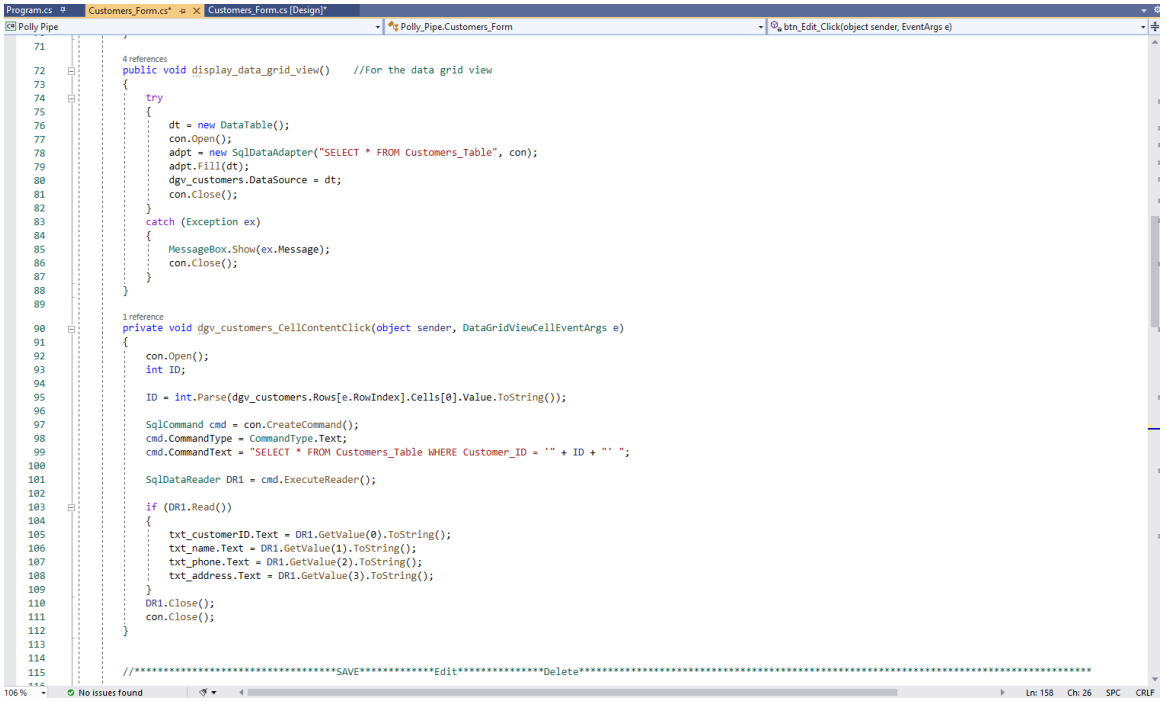
Customer Form (For Admin) →



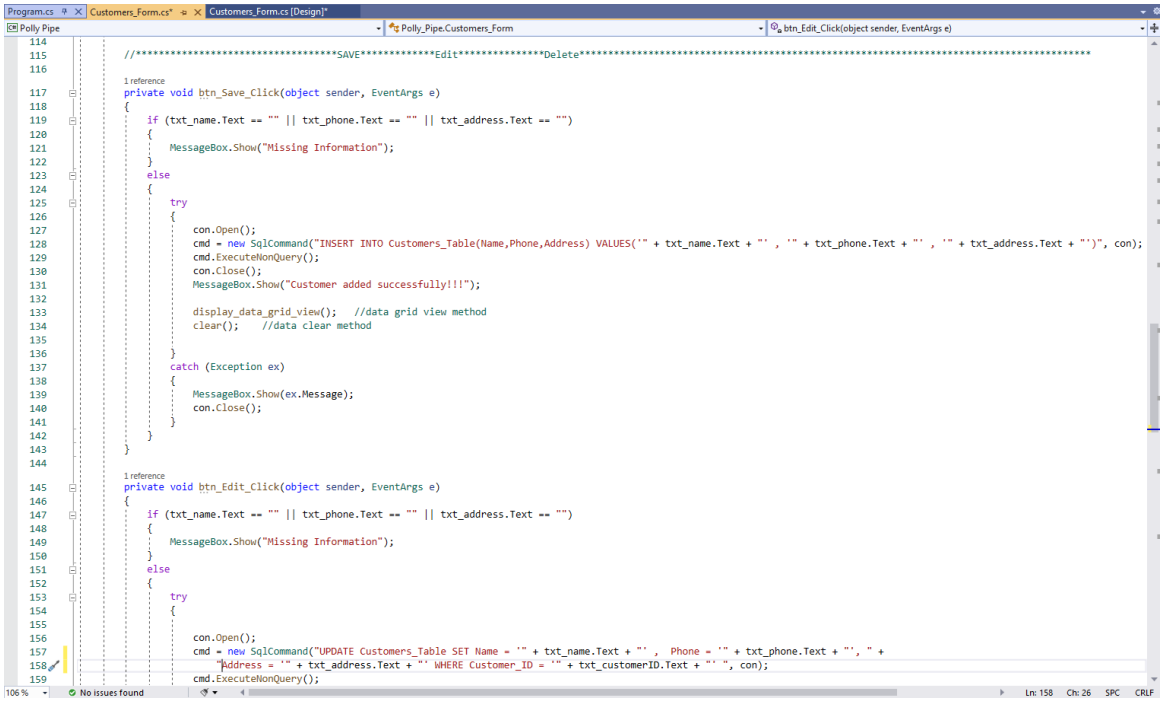

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    13 references
    public partial class Customers_Form : Form
    {
        5 references
        public Customers_Form()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Customers Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***************************************************************************************************************

        1 reference
        private void btn_employees_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
            this.Hide();
        }

        1 reference
        private void btn_equipments_Click(object sender, EventArgs e)
        {
            Equipments_Form obj = new Equipments_Form();
            obj.Show();
```



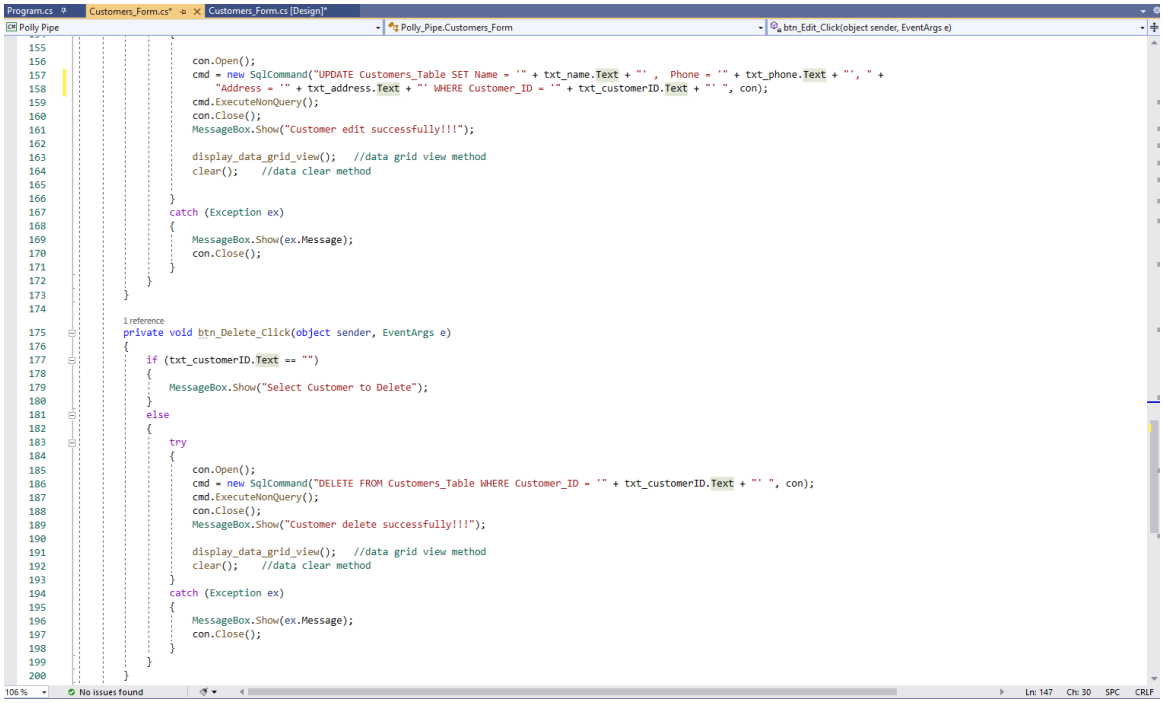
```
Program.cs | Customers_Form.cs* | Customers_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Customers_Form | btn_Edit_Click(object sender, EventArgs e)

41      1 reference
        private void btn_equipments_Click(object sender, EventArgs e)
42      {
43          Equipments_Form obj = new Equipments_Form();
44          obj.Show();
45          this.Hide();
46      }
47
        1 reference
48      private void btn_facilities_Click(object sender, EventArgs e)
49      {
50          Facility_Form obj = new Facility_Form();
51          obj.Show();
52          this.Hide();
53      }
54
        1 reference
55      private void btn_logout_Click(object sender, EventArgs e)
56      {
57          Login_Form obj = new Login_Form();
58          obj.Show();
59          this.Hide();
60      }
61
62      //********************************* OTHER METHODS ***********************************************************************************************************
63
        3 references
64      public void clear()
65      {
66          txt_customerID.Text = "";
67          txt_name.Text = "";
68          txt_phone.Text = "";
69          txt_address.Text = "";
70      }
71
        4 references
72      public void display_data_grid_view()    //For the data grid view
73      {
74          try
75          {
76              dt = new DataTable();
77              con.Open();
78              adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
79              adpt.Fill(dt);
80              dgv_customers.DataSource = dt;
81              con.Close();
82          }
83          catch (Exception ex)

106 % | No issues found | Ln: 158 Ch: 26 SPC CRLF
```



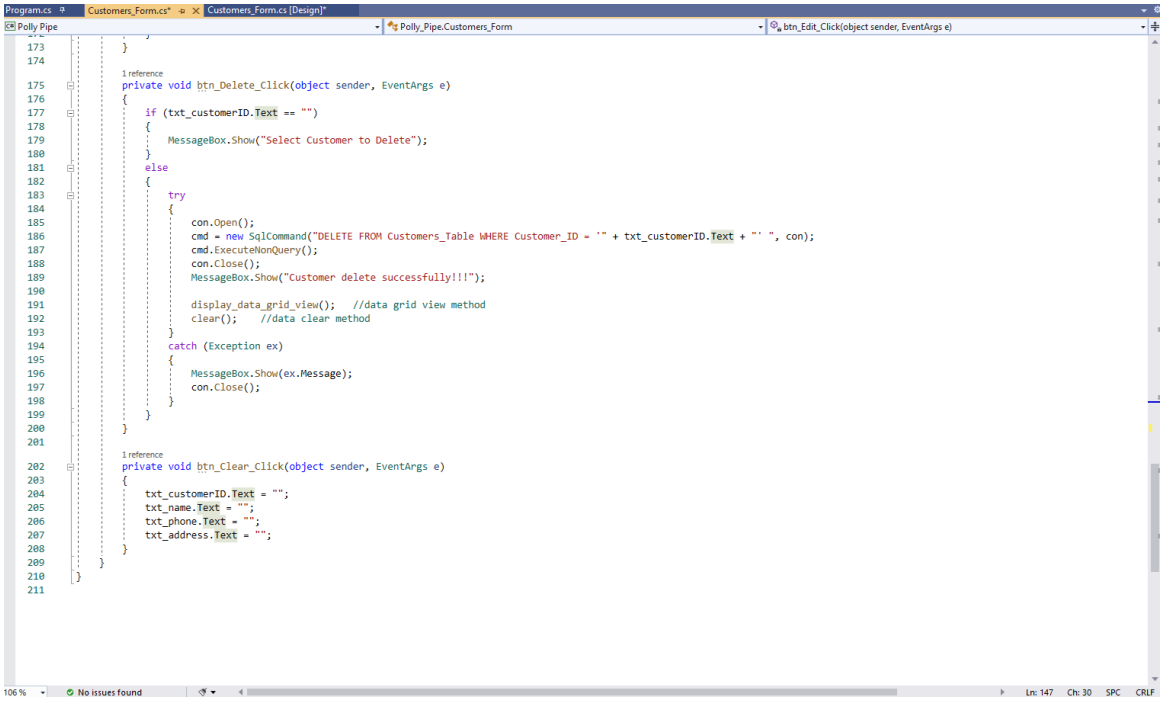
```
Program.cs | Customers_Form.cs* | Customers_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Customers_Form | btn_Edit_Click(object sender, EventArgs e)

71
        4 references
72      public void display_data_grid_view()    //For the data grid view
73      {
74          try
75          {
76              dt = new DataTable();
77              con.Open();
78              adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
79              adpt.Fill(dt);
80              dgv_customers.DataSource = dt;
81              con.Close();
82          }
83          catch (Exception ex)
84          {
85              MessageBox.Show(ex.Message);
86              con.Close();
87          }
88      }
89
        1 reference
90      private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
91      {
92          con.Open();
93          int ID;
94
95          ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());
96
97          SqlCommand cmd = con.CreateCommand();
98          cmd.CommandType = CommandType.Text;
99          cmd.CommandText = "SELECT * FROM Customers_Table WHERE Customer_ID = '" + ID + "' ";
100
101         SqlDataReader DR1 = cmd.ExecuteReader();
102
103         if (DR1.Read())
104         {
105             txt_customerID.Text = DR1.GetValue(0).ToString();
106             txt_name.Text = DR1.GetValue(1).ToString();
107             txt_phone.Text = DR1.GetValue(2).ToString();
108             txt_address.Text = DR1.GetValue(3).ToString();
109         }
110         DR1.Close();
111         con.Close();
112     }
113
114
115     //*************************************SAVE***********Edit**************Delete*****************************************************************************

106 % | No issues found | Ln: 158 Ch: 26 SPC CRLF
```



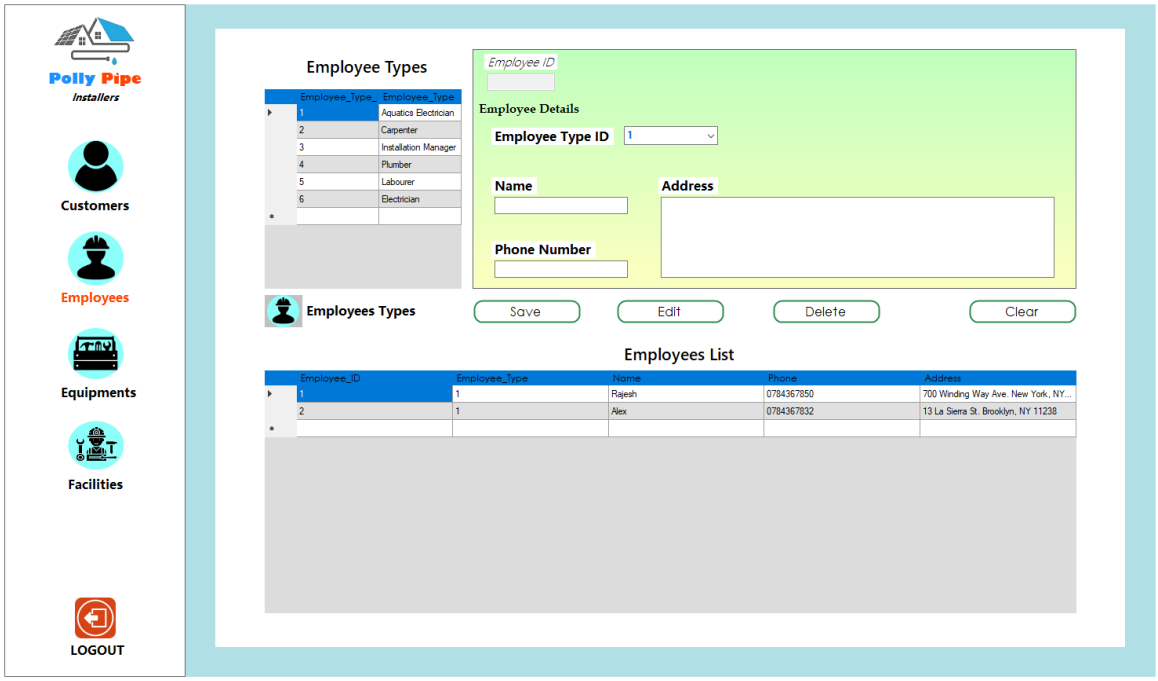
```
//*********************************SAVE***********Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Customers_Table(Name,Phone,Address) VALUES('" + txt_name.Text + "' , '" + txt_phone.Text + "' , '" + txt_address.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer added successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
            cmd = new SqlCommand("UPDATE Customers_Table SET Name = '" + txt_name.Text + "' ,  Phone = '" + txt_phone.Text + "', " +
                "Address = '" + txt_address.Text + "' WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
            cmd.ExecuteNonQuery();
```

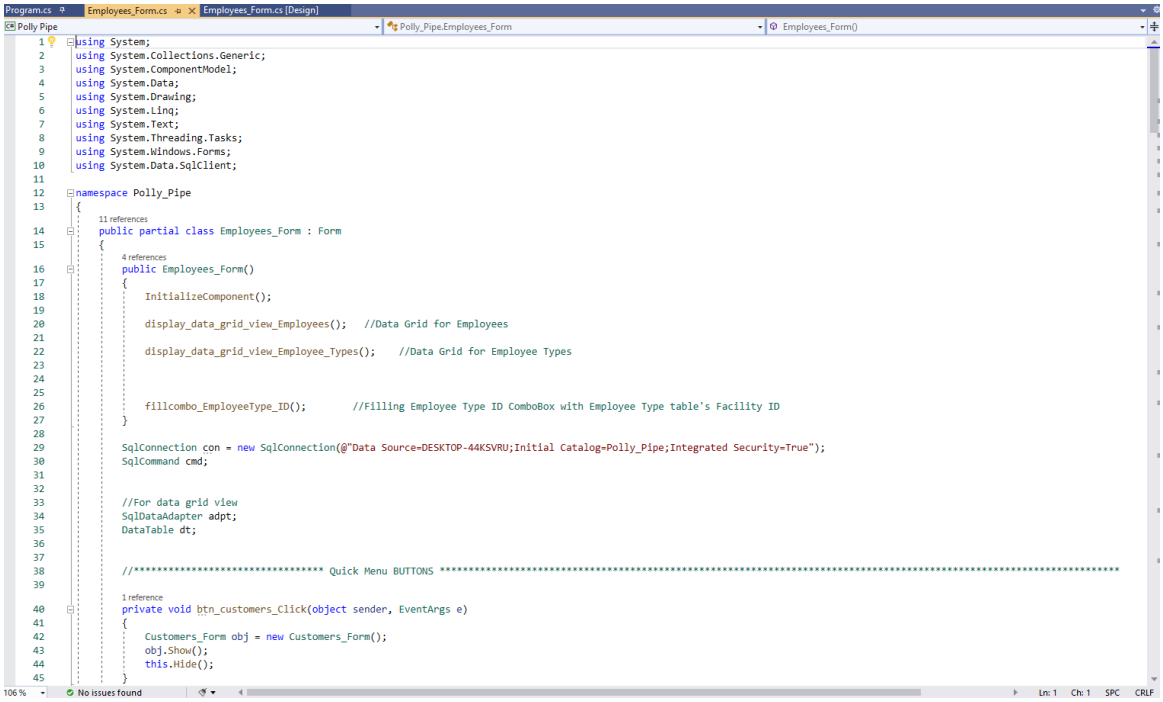


```
            con.Open();
            cmd = new SqlCommand("UPDATE Customers_Table SET Name = '" + txt_name.Text + "' ,  Phone = '" + txt_phone.Text + "', " +
                "Address = '" + txt_address.Text + "' WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer edit successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
    if (txt_customerID.Text == "")
    {
        MessageBox.Show("Select Customer to Delete");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("DELETE FROM Customers_Table WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer delete successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}
```

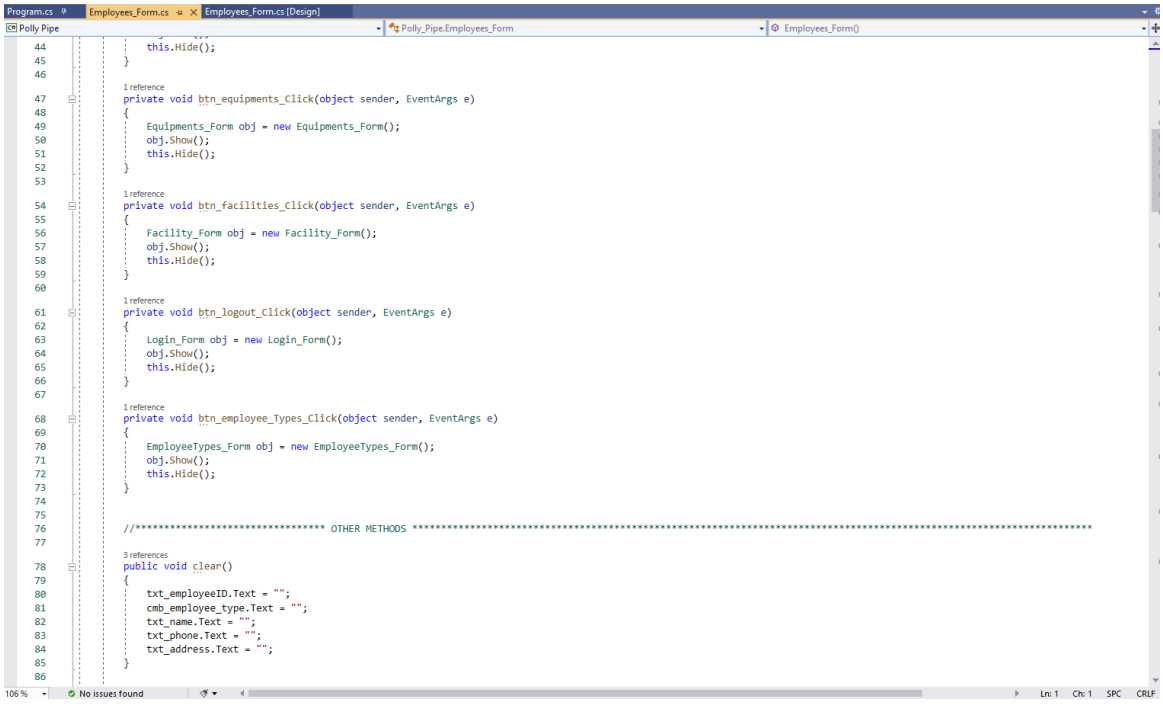

Program.cs
Customers_Form.cs*
Customers_Form.cs [Design]*
Polly Pipe
Polly_Pipe.Customers_Form
btn_Edit_Click(object sender, EventArgs e)
}
1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
if (txt_customerID.Text == "")
{
MessageBox.Show("Select Customer to Delete");
}
else
{
try
{
con.Open();
cmd = new SqlCommand("DELETE FROM Customers_Table WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
cmd.ExecuteNonQuery();
con.Close();
MessageBox.Show("Customer delete successfully!!!");
display_data_grid_view(); //data grid view method
clear(); //data clear method
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
con.Close();
}
}
}
1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
txt_customerID.Text = "";
txt_name.Text = "";
txt_phone.Text = "";
txt_address.Text = "";
}
}
}
106 %
No issues found
Ln: 147
Ch: 30
SPC
CRLF

Employee Form (For Admin) →

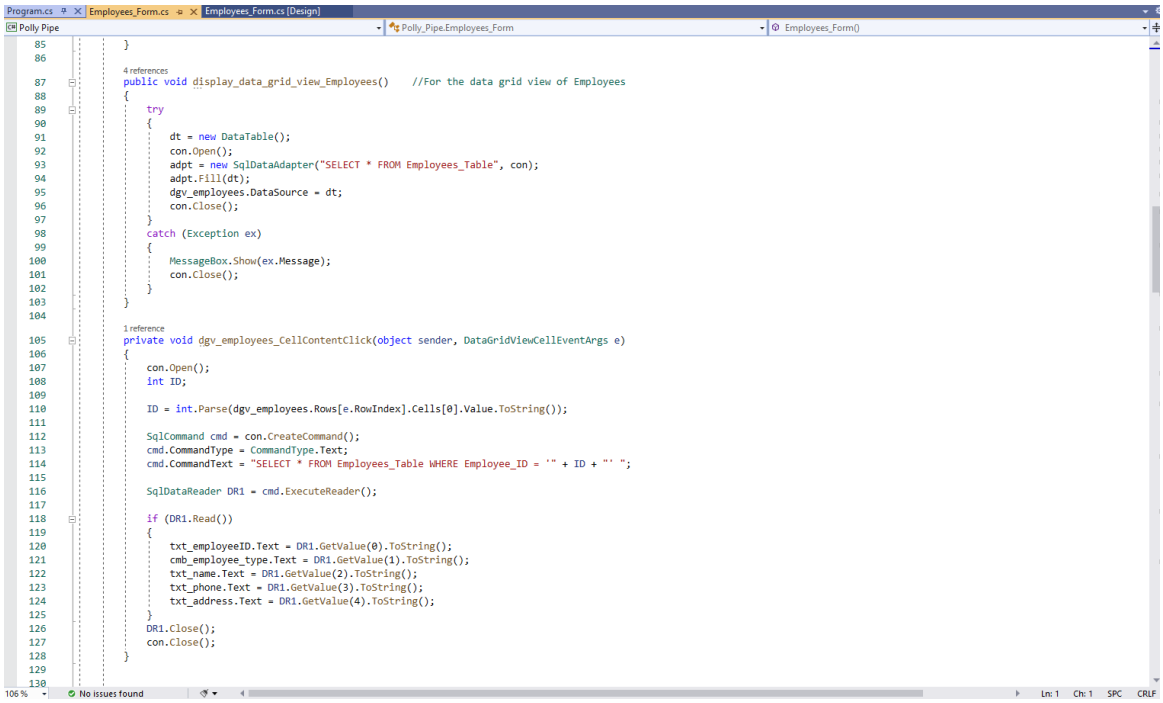




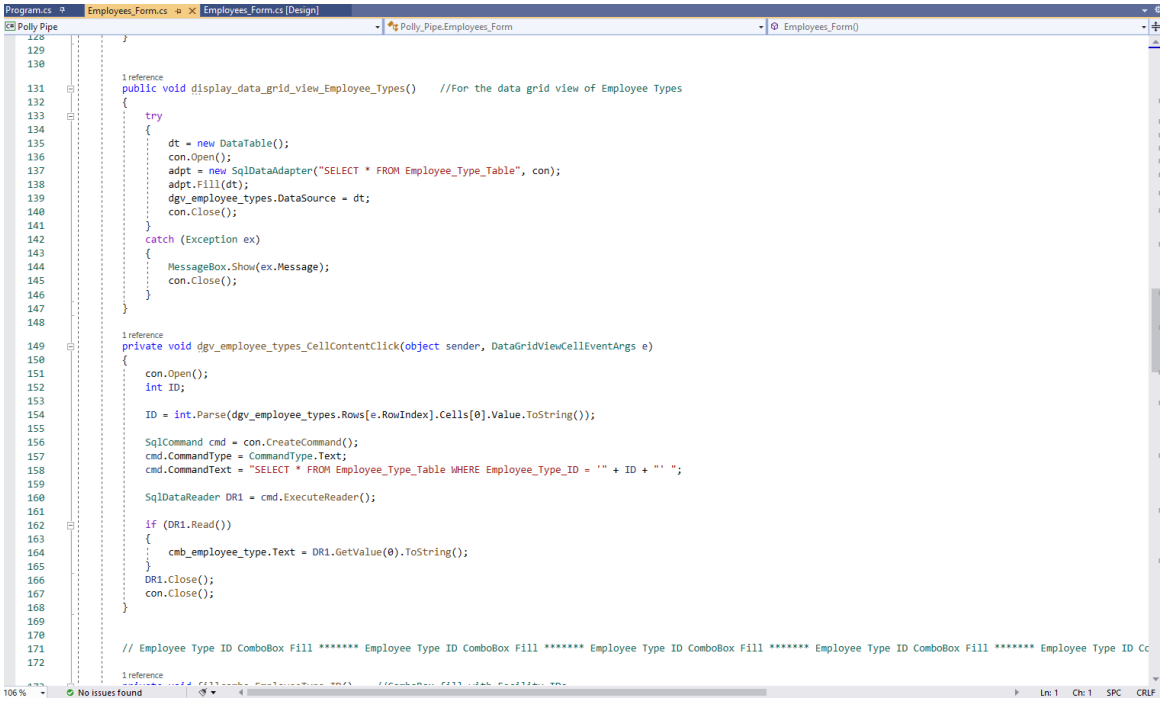
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    11 references
    public partial class Employees_Form : Form
    {
        4 references
        public Employees_Form()
        {
            InitializeComponent();

            display_data_grid_view_Employees();   //Data Grid for Employees

            display_data_grid_view_Employee_Types();    //Data Grid for Employee Types


            fillcombo_EmployeeType_ID();        //Filling Employee Type ID ComboBox with Employee Type table's Facility ID
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ***************************************************************************************************

        1 reference
        private void btn_customers_Click(object sender, EventArgs e)
        {
            Customers_Form obj = new Customers_Form();
            obj.Show();
            this.Hide();
        }
```



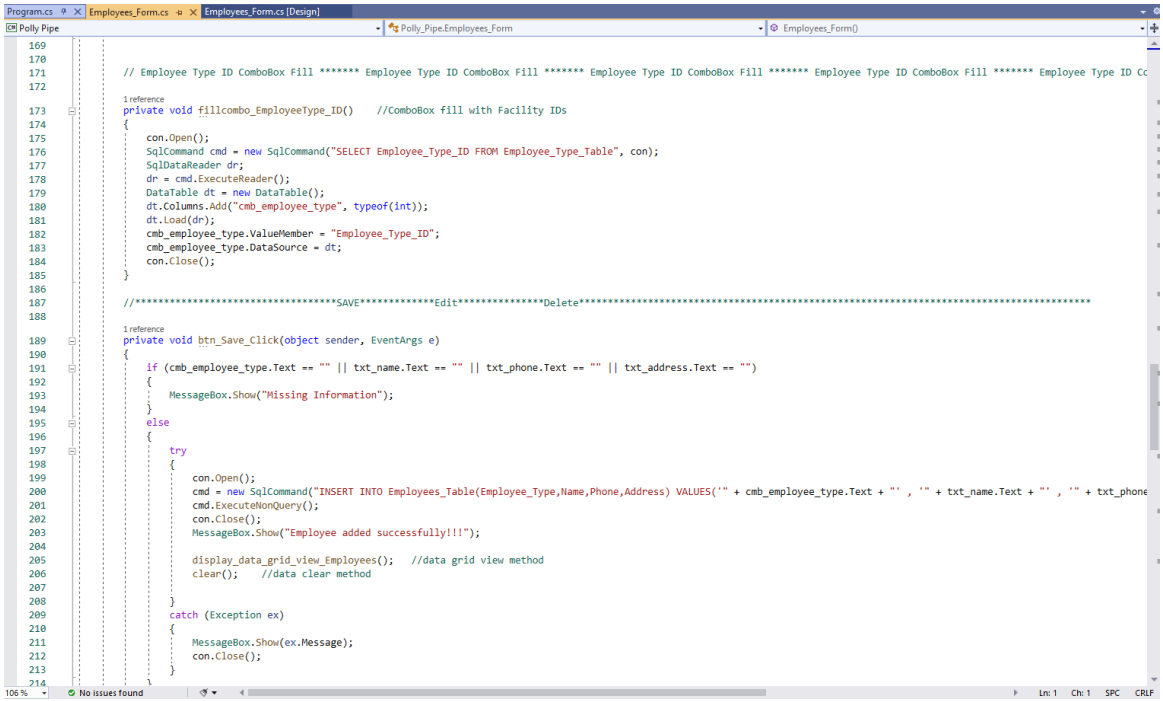
```
Program.cs | Employees_Form.cs | Employees_Form.cs [Design]
Polly Pipe | Polly_Pipe.Employees_Form | Employees_Form()

44          this.Hide();
45          }
46
47          1 reference
            private void btn_equipments_Click(object sender, EventArgs e)
48          {
49              Equipments_Form obj = new Equipments_Form();
50              obj.Show();
51              this.Hide();
52          }
53
54          1 reference
            private void btn_facilities_Click(object sender, EventArgs e)
55          {
56              Facility_Form obj = new Facility_Form();
57              obj.Show();
58              this.Hide();
59          }
60
61          1 reference
            private void btn_logout_Click(object sender, EventArgs e)
62          {
63              Login_Form obj = new Login_Form();
64              obj.Show();
65              this.Hide();
66          }
67
68          1 reference
            private void btn_employee_Types_Click(object sender, EventArgs e)
69          {
70              EmployeeTypes_Form obj = new EmployeeTypes_Form();
71              obj.Show();
72              this.Hide();
73          }
74
75
76          //********************************* OTHER METHODS ***********************************************************************************************************
77
78          3 references
            public void clear()
79          {
80              txt_employeeID.Text = "";
81              cmb_employee_type.Text = "";
82              txt_name.Text = "";
83              txt_phone.Text = "";
84              txt_address.Text = "";
85          }
86

106 %  No issues found                                                                                    Ln: 1  Ch: 1  SPC  CRLF
```



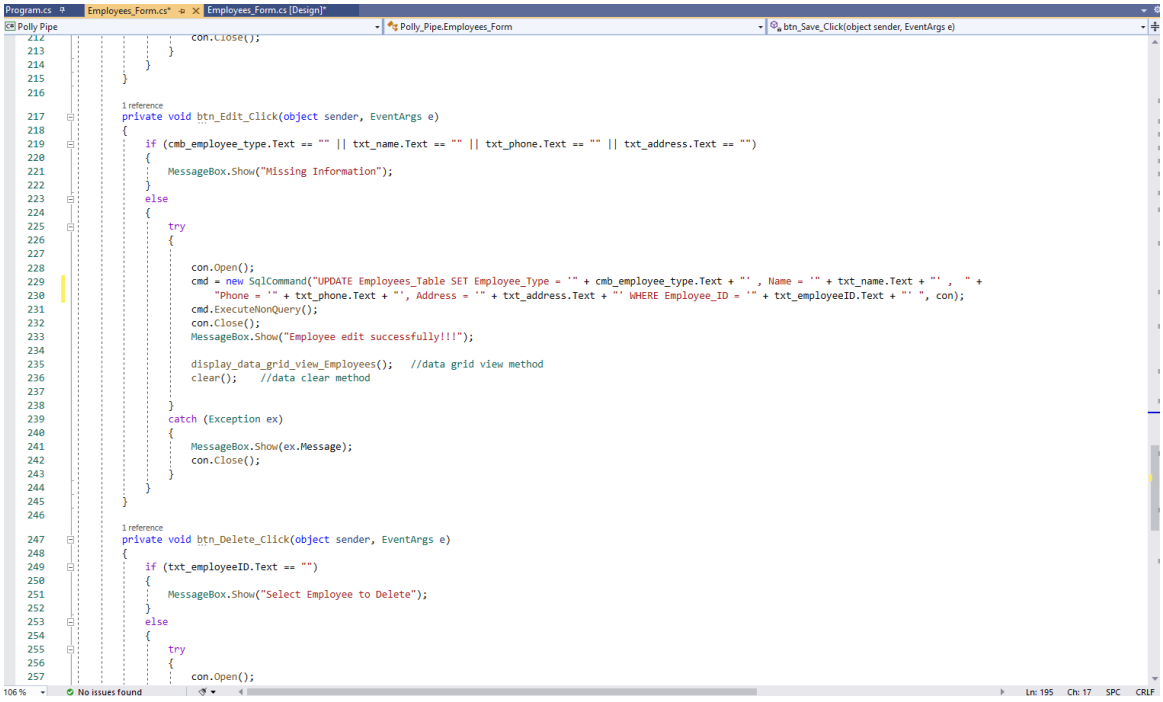
```
Program.cs | Employees_Form.cs | Employees_Form.cs [Design]
Polly Pipe | Polly_Pipe.Employees_Form | Employees_Form()

85          }
86
87          4 references
            public void display_data_grid_view_Employees()    //For the data grid view of Employees
88          {
89              try
90              {
91                  dt = new DataTable();
92                  con.Open();
93                  adpt = new SqlDataAdapter("SELECT * FROM Employees_Table", con);
94                  adpt.Fill(dt);
95                  dgv_employees.DataSource = dt;
96                  con.Close();
97              }
98              catch (Exception ex)
99              {
100                 MessageBox.Show(ex.Message);
101                 con.Close();
102             }
103         }
104
105         1 reference
            private void dgv_employees_CellContentClick(object sender, DataGridViewCellEventArgs e)
106         {
107             con.Open();
108             int ID;
109
110             ID = int.Parse(dgv_employees.Rows[e.RowIndex].Cells[0].Value.ToString());
111
112             SqlCommand cmd = con.CreateCommand();
113             cmd.CommandType = CommandType.Text;
114             cmd.CommandText = "SELECT * FROM Employees_Table WHERE Employee_ID = '" + ID + "' ";
115
116             SqlDataReader DR1 = cmd.ExecuteReader();
117
118             if (DR1.Read())
119             {
120                 txt_employeeID.Text = DR1.GetValue(0).ToString();
121                 cmb_employee_type.Text = DR1.GetValue(1).ToString();
122                 txt_name.Text = DR1.GetValue(2).ToString();
123                 txt_phone.Text = DR1.GetValue(3).ToString();
124                 txt_address.Text = DR1.GetValue(4).ToString();
125             }
126             DR1.Close();
127             con.Close();
128         }
129
130

106 %  No issues found                                                                                    Ln: 1  Ch: 1  SPC  CRLF
```



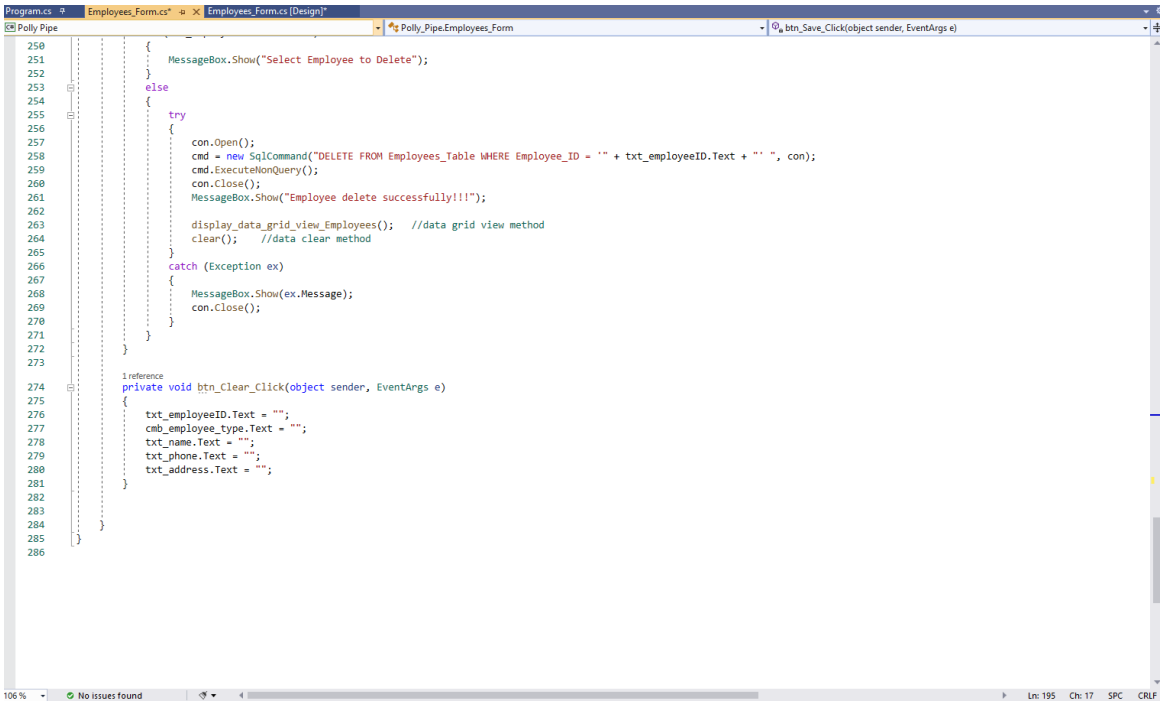
```
Program.cs | Employees_Form.cs | Employees_Form.cs [Design]
Polly Pipe | Polly_Pipe.Employees_Form | Employees_Form()

        }

        1 reference
        public void display_data_grid_view_Employee_Types()    //For the data grid view of Employee Types
        {
            try
            {
                dt = new DataTable();
                con.Open();
                adpt = new SqlDataAdapter("SELECT * FROM Employee_Type_Table", con);
                adpt.Fill(dt);
                dgv_employee_types.DataSource = dt;
                con.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }

        1 reference
        private void dgv_employee_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            con.Open();
            int ID;

            ID = int.Parse(dgv_employee_types.Rows[e.RowIndex].Cells[0].Value.ToString());

            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "SELECT * FROM Employee_Type_Table WHERE Employee_Type_ID = '" + ID + "' ";

            SqlDataReader DR1 = cmd.ExecuteReader();

            if (DR1.Read())
            {
                cmb_employee_type.Text = DR1.GetValue(0).ToString();
            }
            DR1.Close();
            con.Close();
        }


        // Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Co

        1 reference

106 %  No issues found                                                                    Ln: 1  Ch: 1  SPC  CRLF
```



```
Program.cs | Employees_Form.cs | Employees_Form.cs [Design]
Polly Pipe | Polly_Pipe.Employees_Form | Employees_Form()

        // Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Co

        1 reference
        private void fillcombo_EmployeeType_ID()    //ComboBox fill with Facility IDs
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT Employee_Type_ID FROM Employee_Type_Table", con);
            SqlDataReader dr;
            dr = cmd.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Columns.Add("cmb_employee_type", typeof(int));
            dt.Load(dr);
            cmb_employee_type.ValueMember = "Employee_Type_ID";
            cmb_employee_type.DataSource = dt;
            con.Close();
        }

        //**********************************SAVE***********Edit**************Delete*****************************************************************************************

        1 reference
        private void btn_Save_Click(object sender, EventArgs e)
        {
            if (cmb_employee_type.Text == "" || txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("INSERT INTO Employees_Table(Employee_Type,Name,Phone,Address) VALUES('" + cmb_employee_type.Text + "' , '" + txt_name.Text + "' , '" + txt_phone
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Employee added successfully!!!");

                    display_data_grid_view_Employees();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }

106 %  No issues found                                                                    Ln: 1  Ch: 1  SPC  CRLF
```

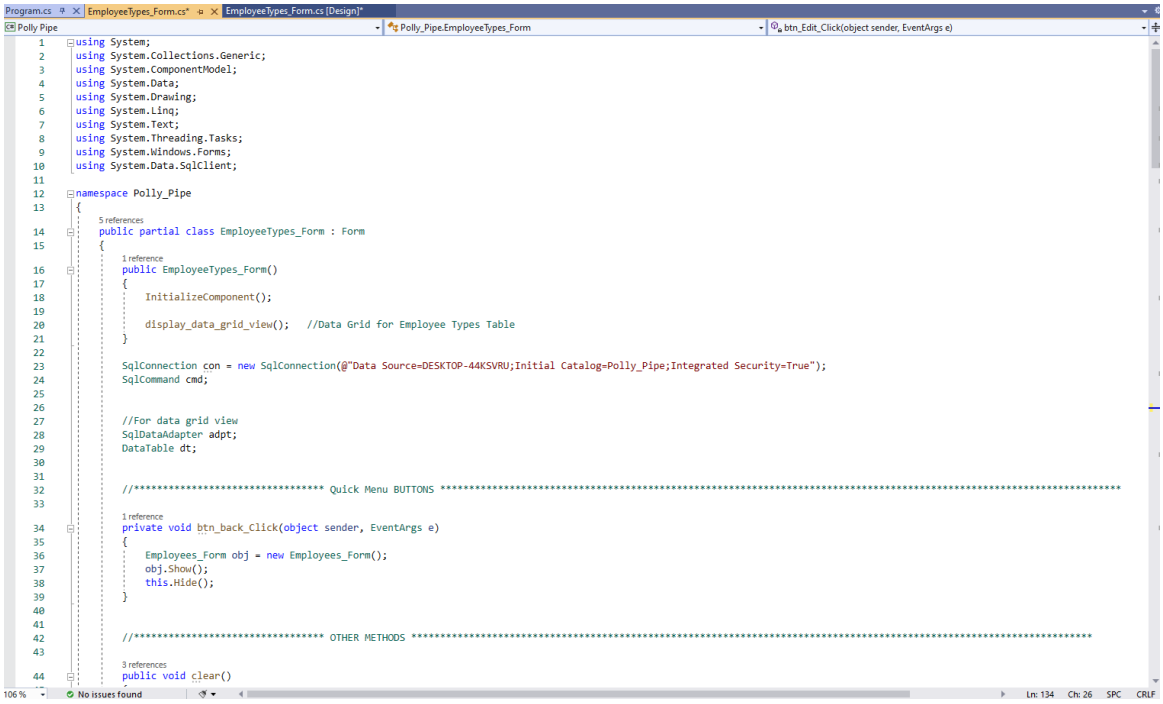


```
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Edit_Click(object sender, EventArgs e)
        {
            if (cmb_employee_type.Text == "" || txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {

                    con.Open();
                    cmd = new SqlCommand("UPDATE Employees_Table SET Employee_Type = '" + cmb_employee_type.Text + "' , Name = '" + txt_name.Text + "' ,  " +
                        "Phone = '" + txt_phone.Text + "', Address = '" + txt_address.Text + "' WHERE Employee_ID = '" + txt_employeeID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Employee edit successfully!!!");

                    display_data_grid_view_Employees();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (txt_employeeID.Text == "")
            {
                MessageBox.Show("Select Employee to Delete");
            }
            else
            {
                try
                {
                    con.Open();
```

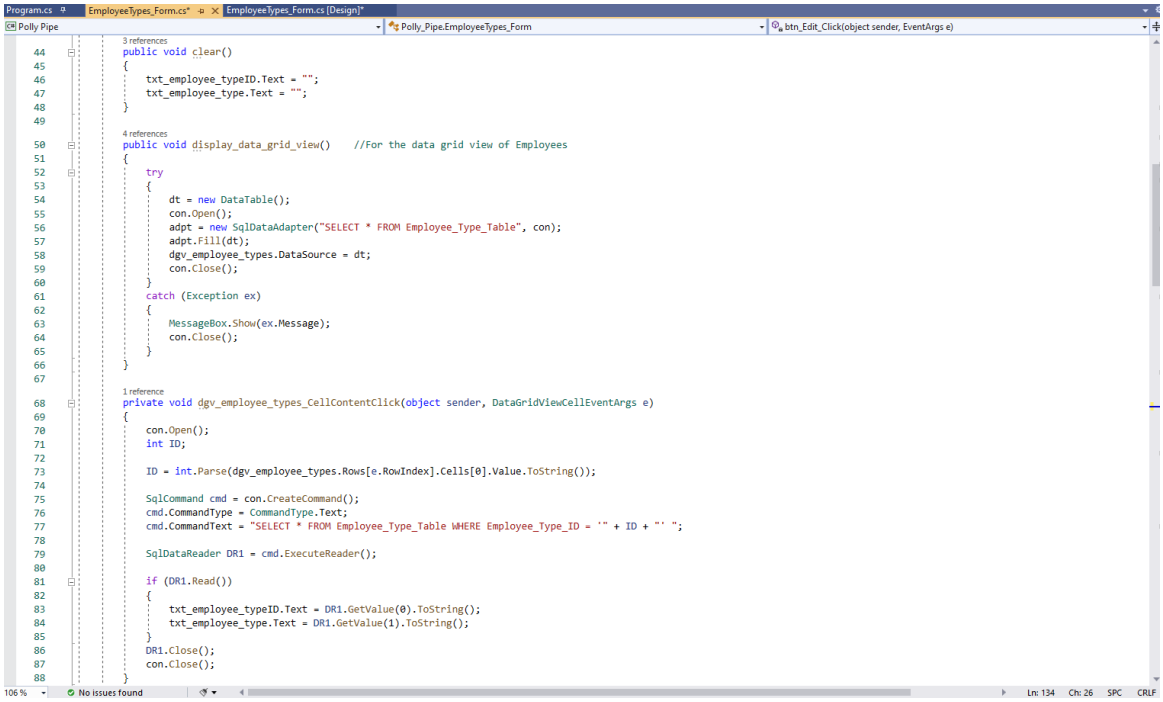


```
                {
                    MessageBox.Show("Select Employee to Delete");
                }
                else
                {
                    try
                    {
                        con.Open();
                        cmd = new SqlCommand("DELETE FROM Employees_Table WHERE Employee_ID = '" + txt_employeeID.Text + "' ", con);
                        cmd.ExecuteNonQuery();
                        con.Close();
                        MessageBox.Show("Employee delete successfully!!!");

                        display_data_grid_view_Employees();   //data grid view method
                        clear();    //data clear method
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                        con.Close();
                    }
                }
            }

            1 reference
            private void btn_Clear_Click(object sender, EventArgs e)
            {
                txt_employeeID.Text = "";
                cmb_employee_type.Text = "";
                txt_name.Text = "";
                txt_phone.Text = "";
                txt_address.Text = "";
            }


        }
    }
```

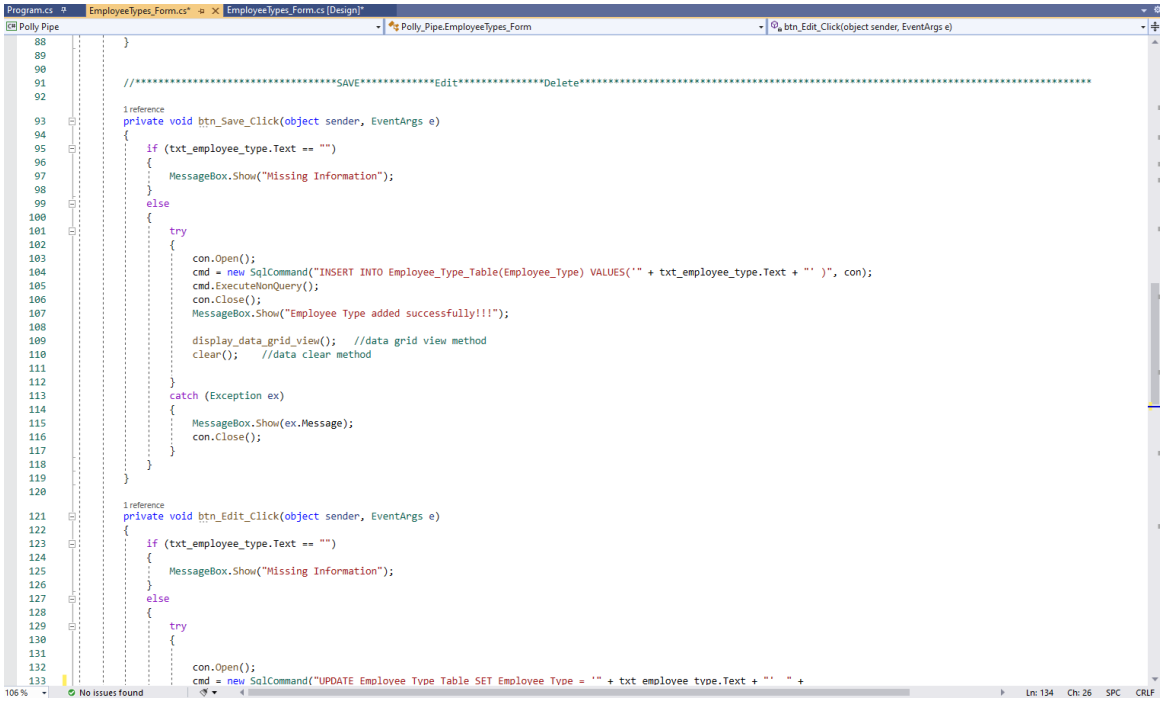
Employee Type Form (For Admin) →



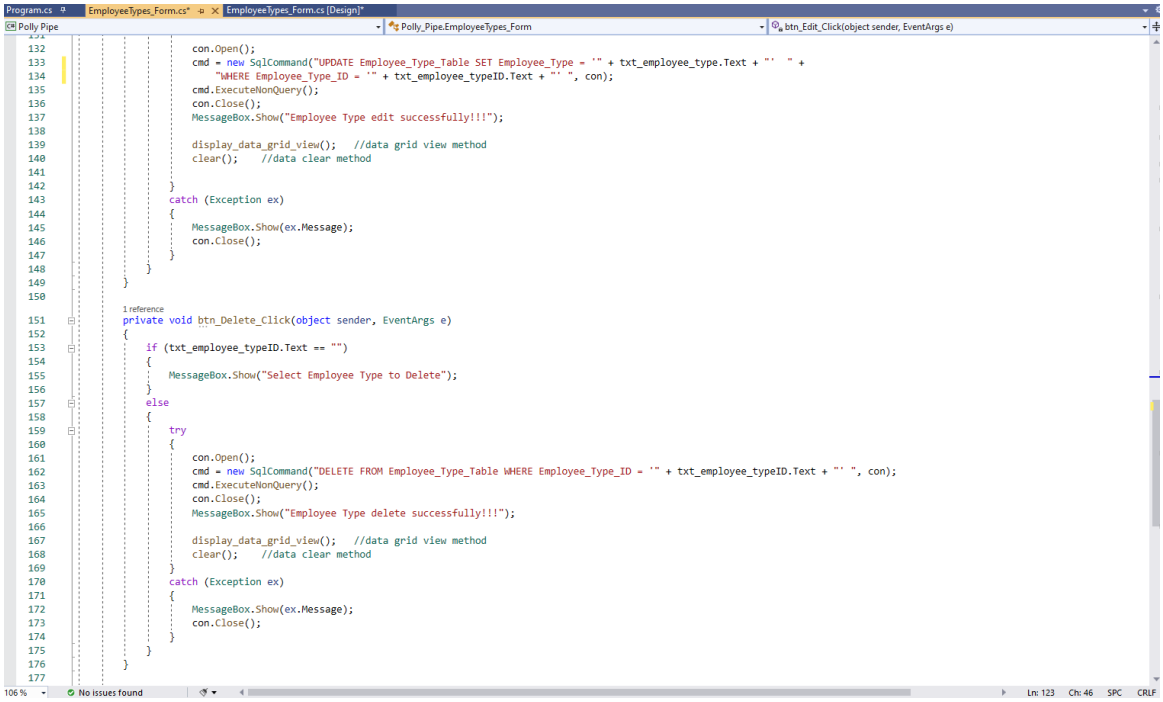
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    5 references
    public partial class EmployeeTypes_Form : Form
    {
        1 reference
        public EmployeeTypes_Form()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Employee Types Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***********************************************************************************************************

        1 reference
        private void btn_back_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
            this.Hide();
        }


        //********************************* OTHER METHODS ****************************************************************************************************************

        3 references
        public void clear()
```



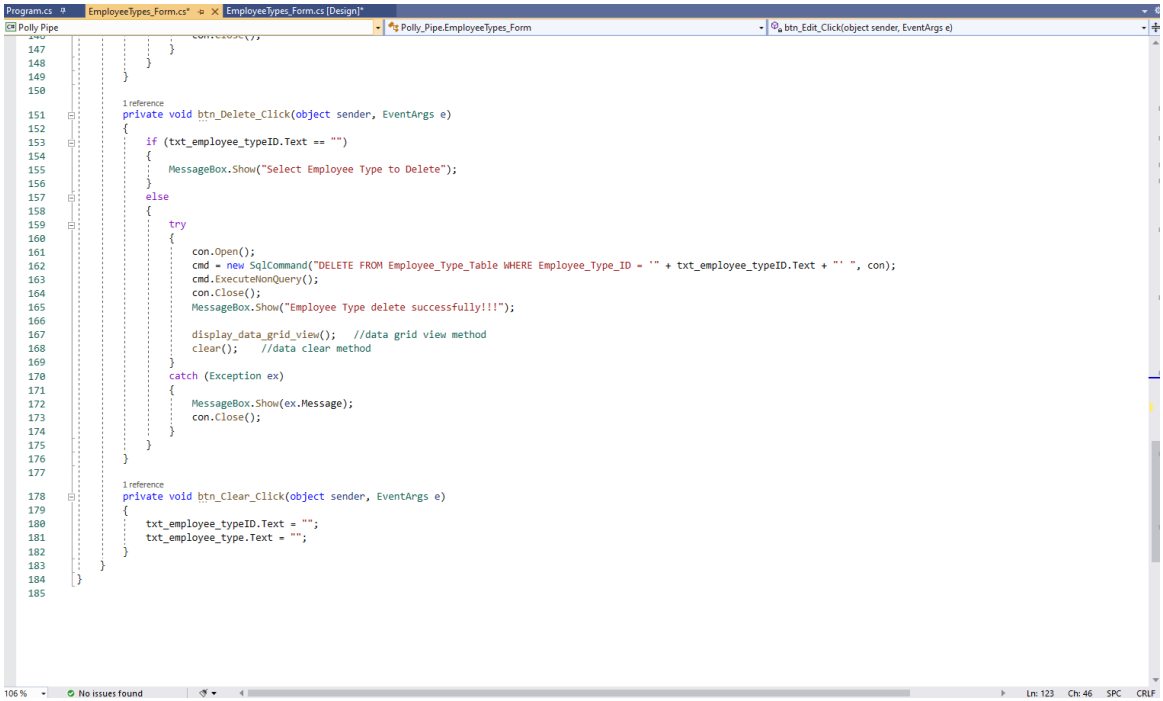
```
3 references
public void clear()
{
    txt_employee_typeID.Text = "";
    txt_employee_type.Text = "";
}

4 references
public void display_data_grid_view()    //For the data grid view of Employees
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Employee_Type_Table", con);
        adpt.Fill(dt);
        dgv_employee_types.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void dgv_employee_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_employee_types.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Employee_Type_Table WHERE Employee_Type_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_employee_typeID.Text = DR1.GetValue(0).ToString();
        txt_employee_type.Text = DR1.GetValue(1).ToString();
    }
    DR1.Close();
    con.Close();
}
```



```
}


//*********************************SAVE***********Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_employee_type.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Employee_Type_Table(Employee_Type) VALUES('" + txt_employee_type.Text + "' )", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Employee Type added successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }
}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_employee_type.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
            cmd = new SqlCommand("UPDATE Employee Type Table SET Employee Type = '" + txt employee type.Text + "'  " +
```

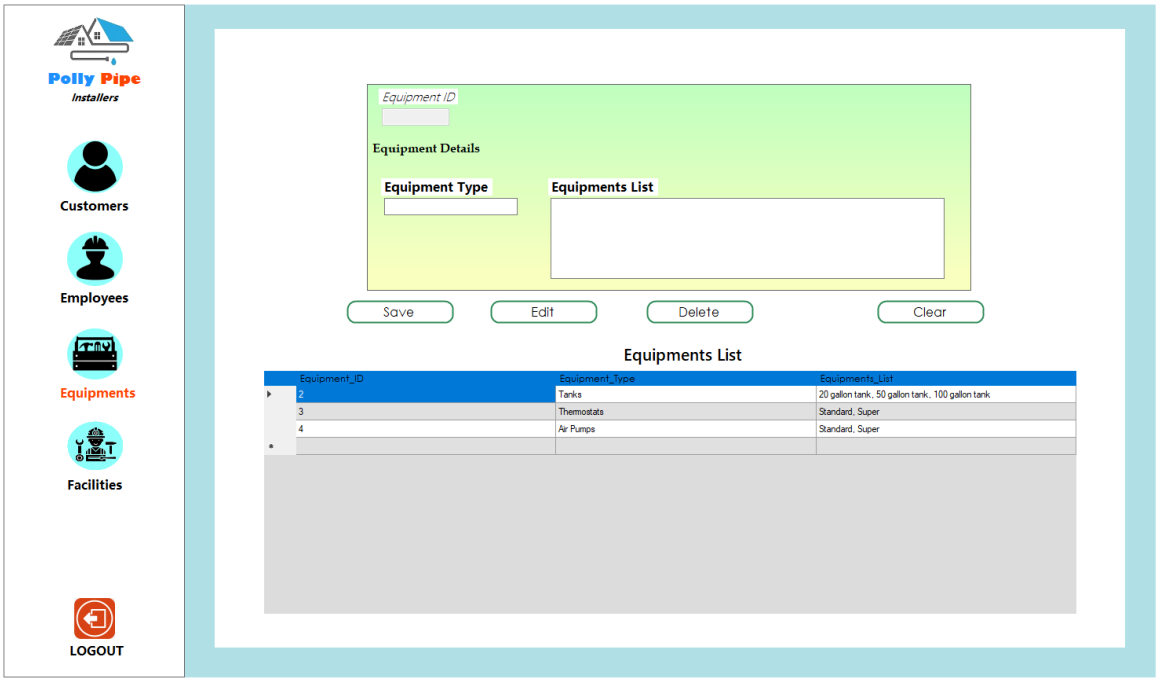


```
                con.Open();
                cmd = new SqlCommand("UPDATE Employee_Type_Table SET Employee_Type = '" + txt_employee_type.Text + "'  " +
                    "WHERE Employee_Type_ID = '" + txt_employee_typeID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Employee Type edit successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Delete_Click(object sender, EventArgs e)
    {
        if (txt_employee_typeID.Text == "")
        {
            MessageBox.Show("Select Employee Type to Delete");
        }
        else
        {
            try
            {
                con.Open();
                cmd = new SqlCommand("DELETE FROM Employee_Type_Table WHERE Employee_Type_ID = '" + txt_employee_typeID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Employee Type delete successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }
```

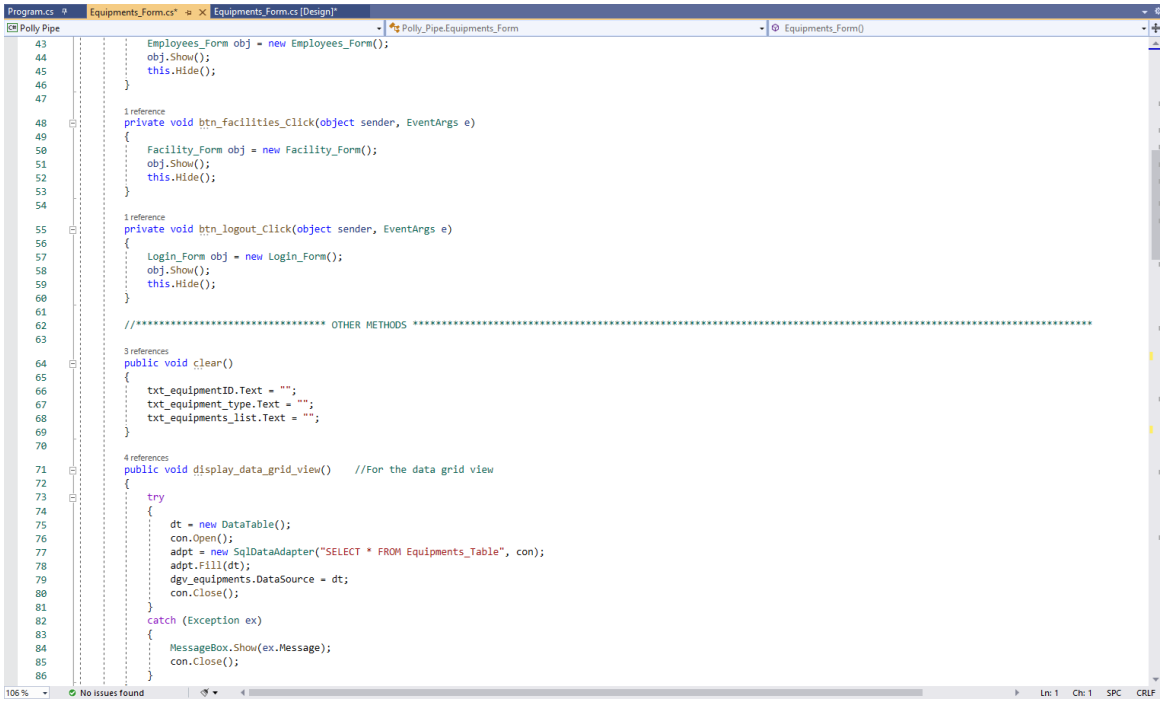


```
            }
        }
    }

    1 reference
    private void btn_Delete_Click(object sender, EventArgs e)
    {
        if (txt_employee_typeID.Text == "")
        {
            MessageBox.Show("Select Employee Type to Delete");
        }
        else
        {
            try
            {
                con.Open();
                cmd = new SqlCommand("DELETE FROM Employee_Type_Table WHERE Employee_Type_ID = '" + txt_employee_typeID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Employee Type delete successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Clear_Click(object sender, EventArgs e)
    {
        txt_employee_typeID.Text = "";
        txt_employee_type.Text = "";
    }
}
```

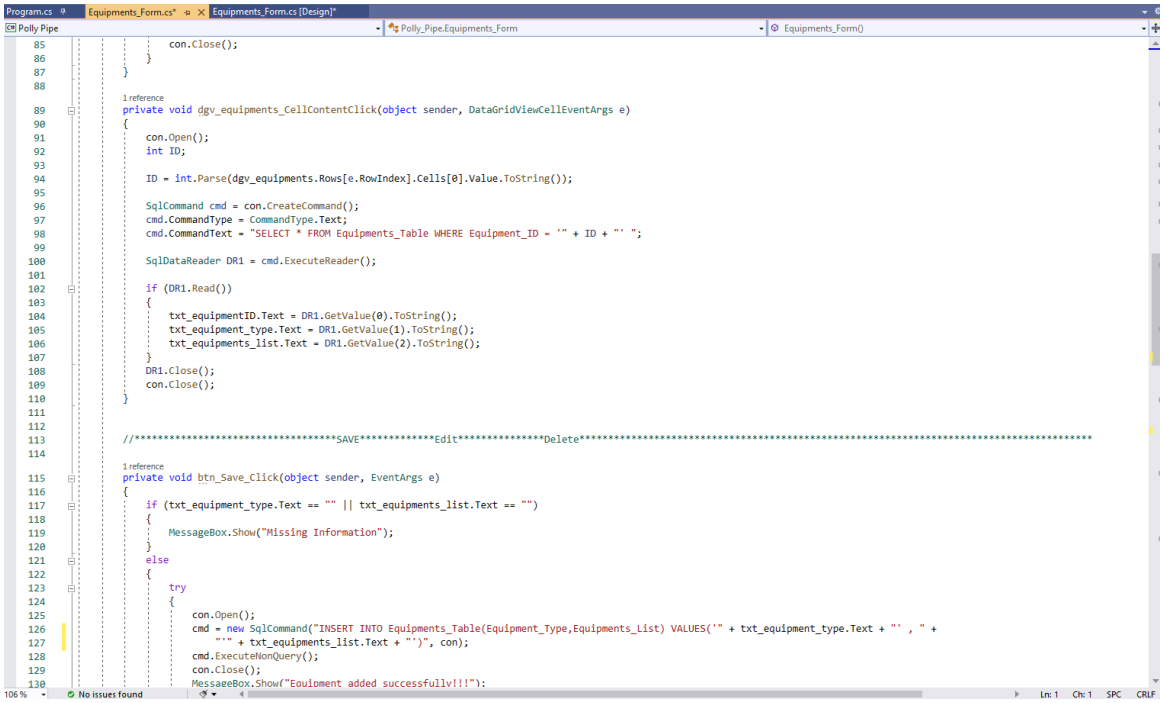
Equipment Form (For Admin) →



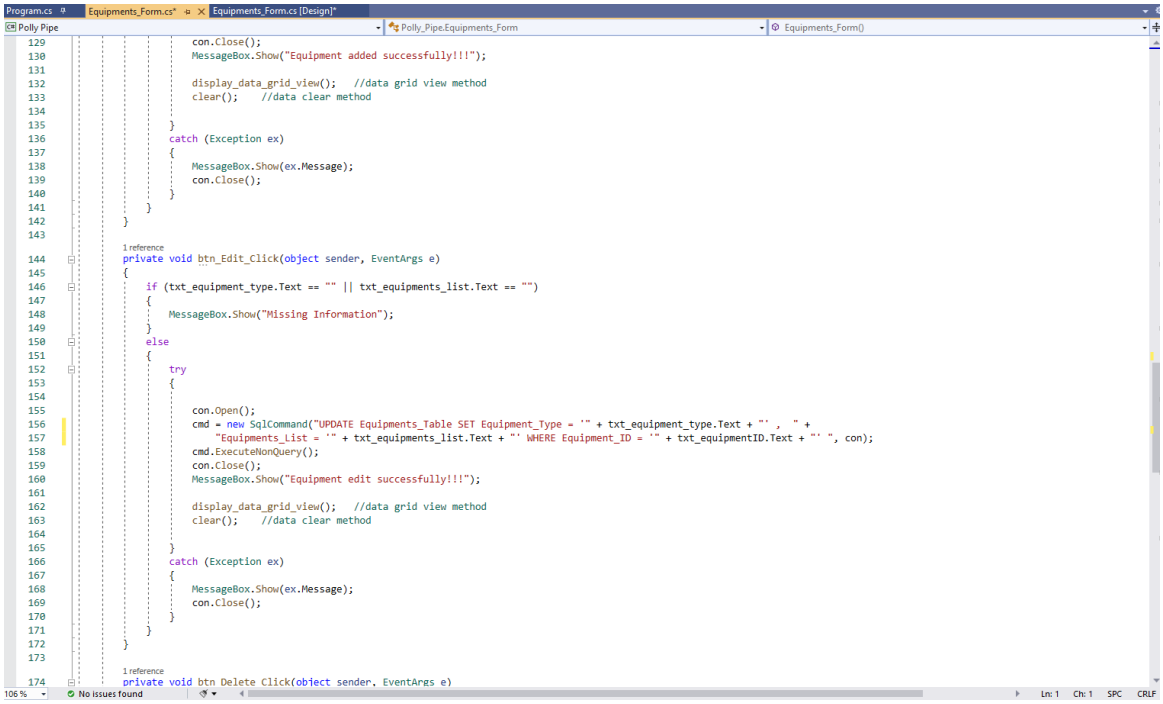

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    public partial class Equipments_Form : Form
    {
        public Equipments_Form()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Equipments Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************

        private void btn_customers_Click(object sender, EventArgs e)
        {
            Customers_Form obj = new Customers_Form();
            obj.Show();
            this.Hide();
        }

        private void btn_employees_Click(object sender, EventArgs e)
        {
            Employees_Form obj = new Employees_Form();
            obj.Show();
```



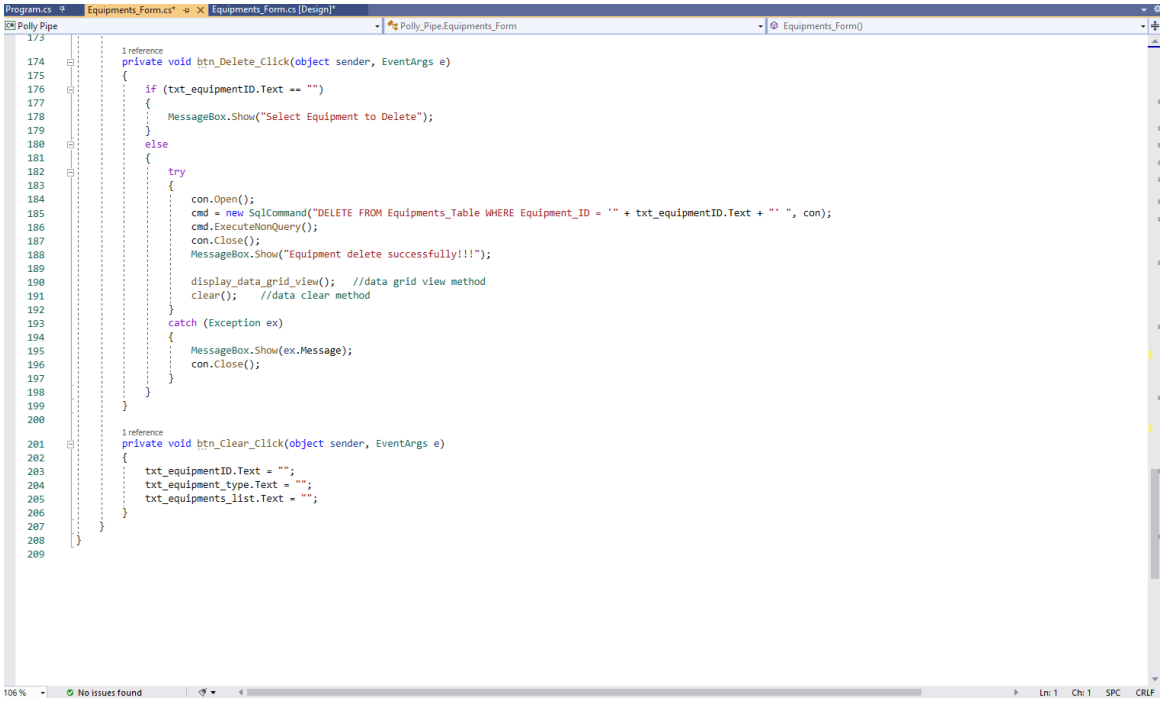
```
Employees_Form obj = new Employees_Form();
obj.Show();
this.Hide();
}

1 reference
private void btn_facilities_Click(object sender, EventArgs e)
{
    Facility_Form obj = new Facility_Form();
    obj.Show();
    this.Hide();
}

1 reference
private void btn_logout_Click(object sender, EventArgs e)
{
    Login_Form obj = new Login_Form();
    obj.Show();
    this.Hide();
}

//********************************* OTHER METHODS ***********************************************************************************************************************

3 references
public void clear()
{
    txt_equipmentID.Text = "";
    txt_equipment_type.Text = "";
    txt_equipments_list.Text = "";
}

4 references
public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Equipments_Table", con);
        adpt.Fill(dt);
        dgv_equipments.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
```



```
        con.Close();
    }
}

1 reference
private void dgv_equipments_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_equipments.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Equipments_Table WHERE Equipment_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_equipmentID.Text = DR1.GetValue(0).ToString();
        txt_equipment_type.Text = DR1.GetValue(1).ToString();
        txt_equipments_list.Text = DR1.GetValue(2).ToString();
    }
    DR1.Close();
    con.Close();
}


//*********************************SAVE************Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_equipment_type.Text == "" || txt_equipments_list.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Equipments_Table(Equipment_Type,Equipments_List) VALUES('" + txt_equipment_type.Text + "' , " +
                "'" + txt_equipments_list.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Equipment added successfully!!!");
```

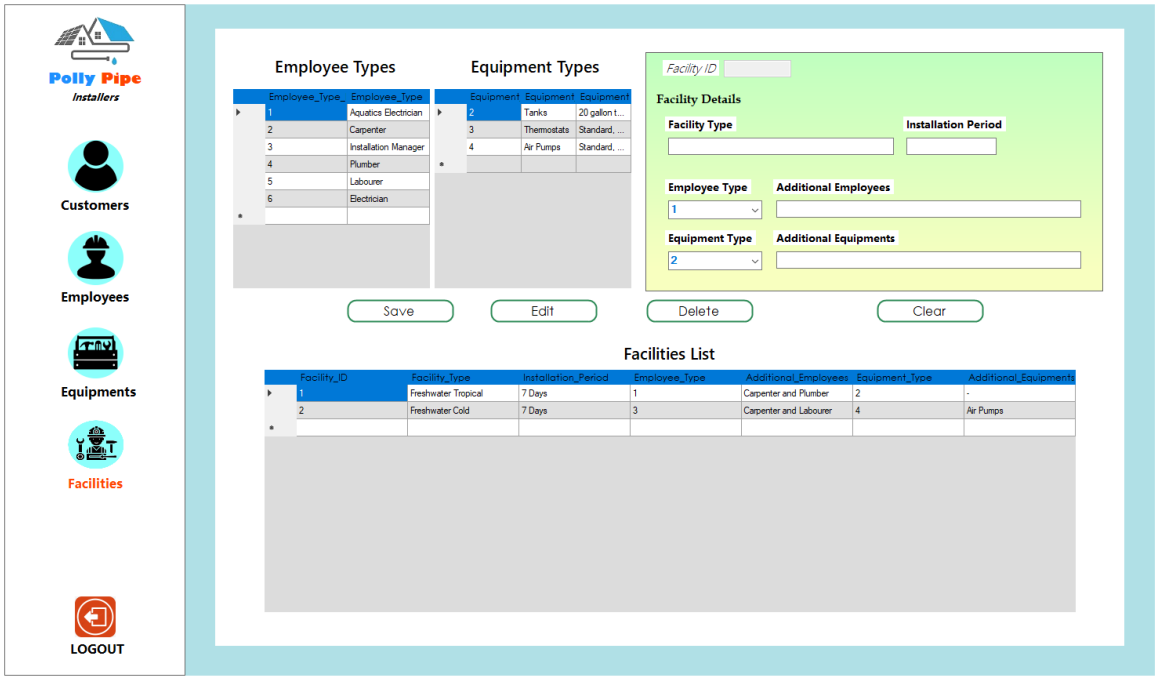


```
Program.cs  Equipments_Form.cs*  Equipments_Form.cs [Design]*
Polly Pipe  Polly_Pipe.Equipments_Form  Equipments_Form()

                    con.Close();
                    MessageBox.Show("Equipment added successfully!!!");

                    display_data_grid_view();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Edit_Click(object sender, EventArgs e)
        {
            if (txt_equipment_type.Text == "" || txt_equipments_list.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {

                    con.Open();
                    cmd = new SqlCommand("UPDATE Equipments_Table SET Equipment_Type = '" + txt_equipment_type.Text + "' ,  " +
                        "Equipments_List = '" + txt_equipments_list.Text + "' WHERE Equipment_ID = '" + txt_equipmentID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Equipment edit successfully!!!");

                    display_data_grid_view();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)

106 %  No issues found  Ln: 1  Ch: 1  SPC  CRLF
```

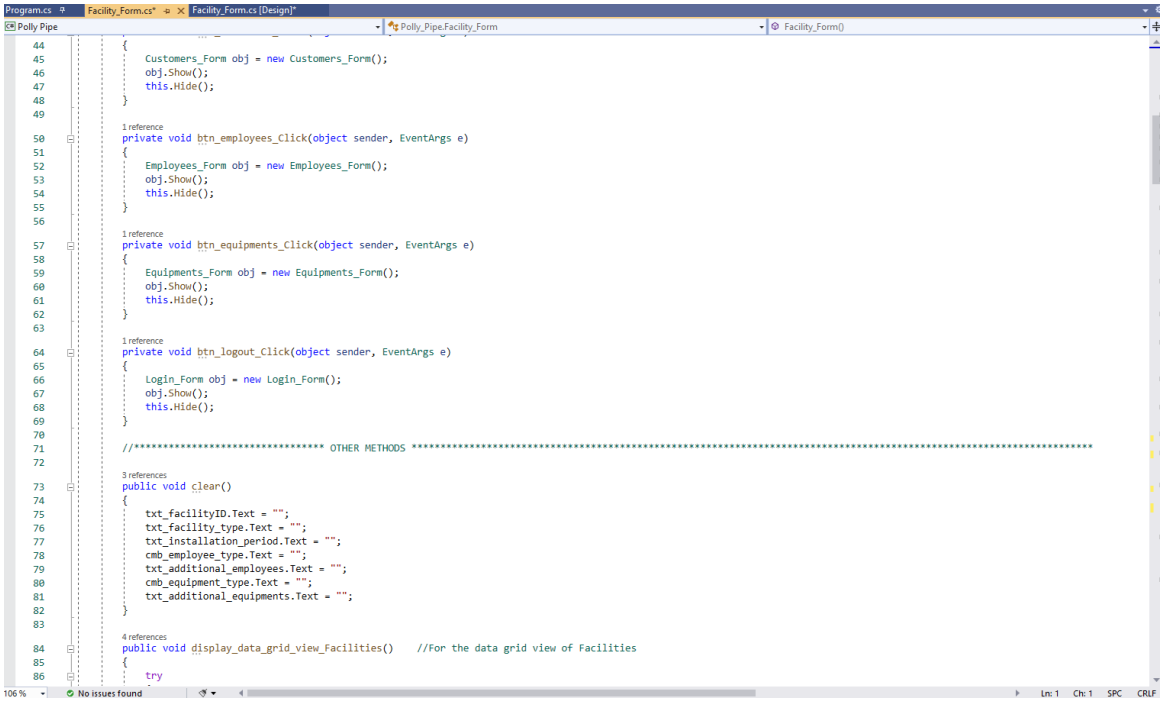


```
Program.cs  Equipments_Form.cs*  Equipments_Form.cs [Design]*
Polly Pipe  Polly_Pipe.Equipments_Form  Equipments_Form()

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (txt_equipmentID.Text == "")
            {
                MessageBox.Show("Select Equipment to Delete");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("DELETE FROM Equipments_Table WHERE Equipment_ID = '" + txt_equipmentID.Text + "' ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Equipment delete successfully!!!");

                    display_data_grid_view();   //data grid view method
                    clear();    //data clear method
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Clear_Click(object sender, EventArgs e)
        {
            txt_equipmentID.Text = "";
            txt_equipment_type.Text = "";
            txt_equipments_list.Text = "";
        }
    }
}

106 %  No issues found  Ln: 1  Ch: 1  SPC  CRLF
```

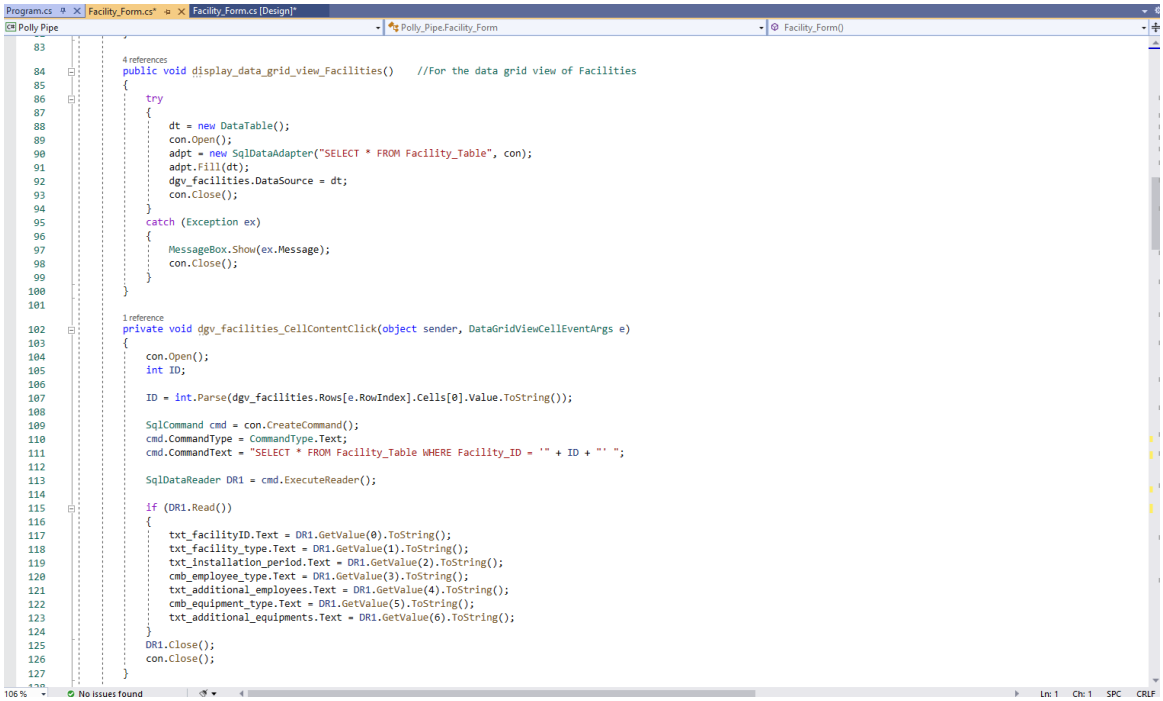
Facility Form (For Admin) →



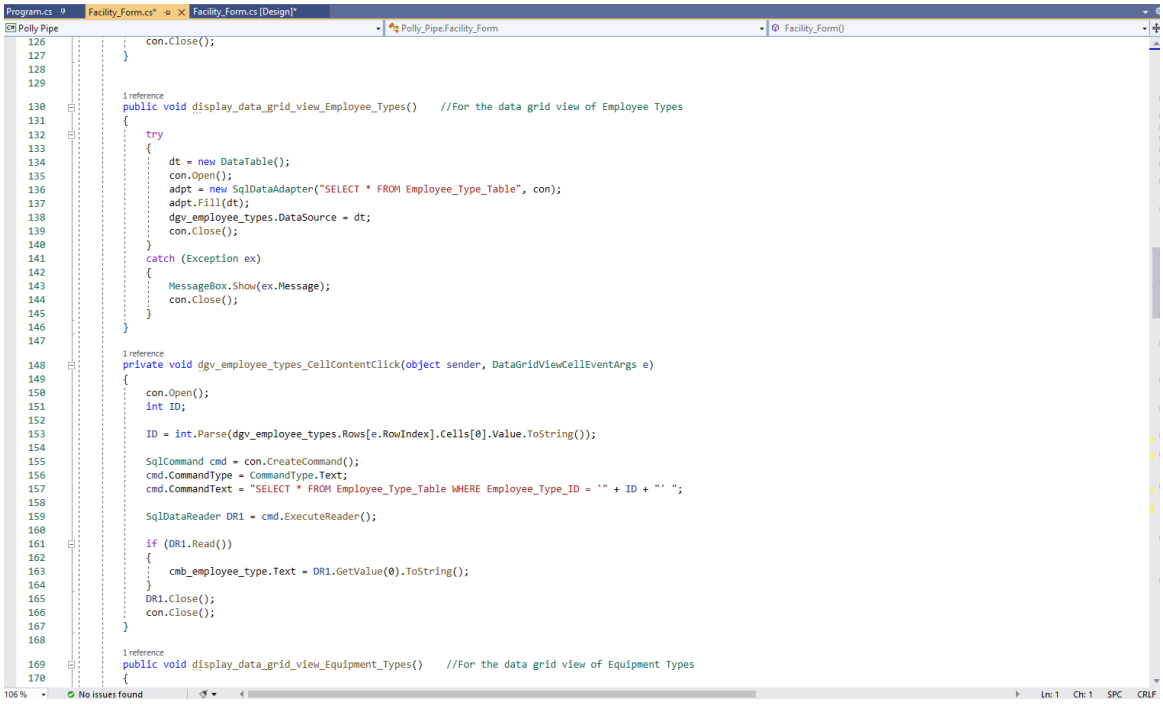

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    public partial class Facility_Form : Form
    {
        public Facility_Form()
        {
            InitializeComponent();

            display_data_grid_view_Facilities();   //Data Grid for Facilities

            display_data_grid_view_Employee_Types();    //Data Grid for Employee Types

            display_data_grid_view_Equipment_Types();   //Data Grid for Equipment Types


            fillcombo_EmployeeType_ID();        //Filling Employee Type ID ComboBox with Employee Type table's Facility ID

            fillcombo_EquipmentType_ID();       //Filling Equipment ID ID ComboBox with Equipment table's Facility ID
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //******************************** Quick Menu BUTTONS ********************************************************************************************************************

        private void btn_customers_Click(object sender, EventArgs e)
        {
            Customers_Form obj = new Customers_Form();
```



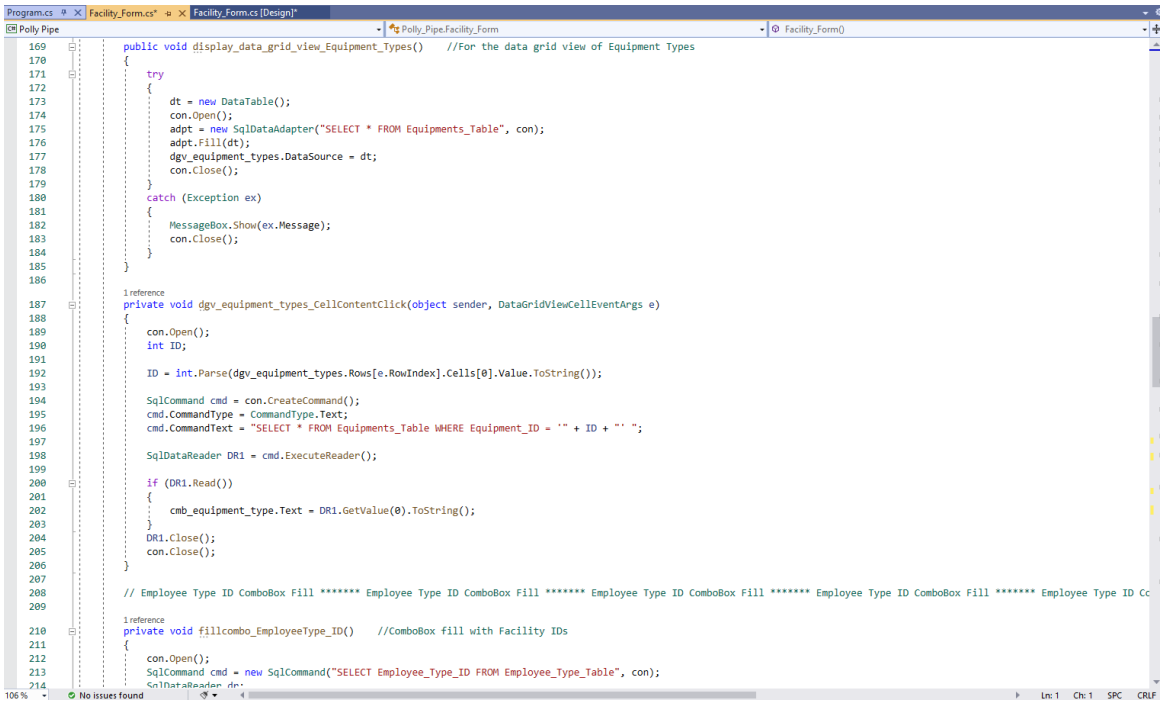
```
44          {
45              Customers_Form obj = new Customers_Form();
46              obj.Show();
47              this.Hide();
48          }
49
50          1 reference
            private void btn_employees_Click(object sender, EventArgs e)
51          {
52              Employees_Form obj = new Employees_Form();
53              obj.Show();
54              this.Hide();
55          }
56
57          1 reference
            private void btn_equipments_Click(object sender, EventArgs e)
58          {
59              Equipments_Form obj = new Equipments_Form();
60              obj.Show();
61              this.Hide();
62          }
63
64          1 reference
            private void btn_logout_Click(object sender, EventArgs e)
65          {
66              Login_Form obj = new Login_Form();
67              obj.Show();
68              this.Hide();
69          }
70
71          //******************************** OTHER METHODS ***********************************************************************************************************
72
73          3 references
            public void clear()
74          {
75              txt_facilityID.Text = "";
76              txt_facility_type.Text = "";
77              txt_installation_period.Text = "";
78              cmb_employee_type.Text = "";
79              txt_additional_employees.Text = "";
80              cmb_equipment_type.Text = "";
81              txt_additional_equipments.Text = "";
82          }
83
84          4 references
            public void display_data_grid_view_Facilities()    //For the data grid view of Facilities
85          {
86              try
```



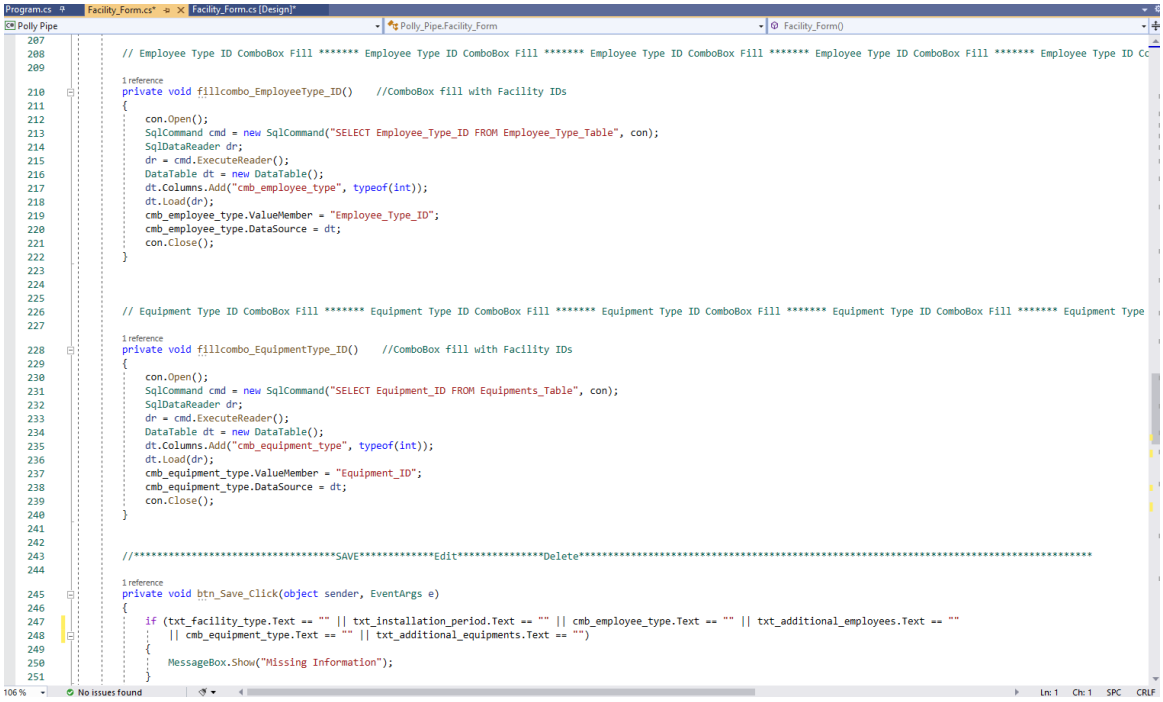
```
83
84          4 references
            public void display_data_grid_view_Facilities()    //For the data grid view of Facilities
85          {
86              try
87              {
88                  dt = new DataTable();
89                  con.Open();
90                  adpt = new SqlDataAdapter("SELECT * FROM Facility_Table", con);
91                  adpt.Fill(dt);
92                  dgv_facilities.DataSource = dt;
93                  con.Close();
94              }
95              catch (Exception ex)
96              {
97                  MessageBox.Show(ex.Message);
98                  con.Close();
99              }
100         }
101
102         1 reference
            private void dgv_facilities_CellContentClick(object sender, DataGridViewCellEventArgs e)
103         {
104             con.Open();
105             int ID;
106
107             ID = int.Parse(dgv_facilities.Rows[e.RowIndex].Cells[0].Value.ToString());
108
109             SqlCommand cmd = con.CreateCommand();
110             cmd.CommandType = CommandType.Text;
111             cmd.CommandText = "SELECT * FROM Facility_Table WHERE Facility_ID = '" + ID + "' ";
112
113             SqlDataReader DR1 = cmd.ExecuteReader();
114
115             if (DR1.Read())
116             {
117                 txt_facilityID.Text = DR1.GetValue(0).ToString();
118                 txt_facility_type.Text = DR1.GetValue(1).ToString();
119                 txt_installation_period.Text = DR1.GetValue(2).ToString();
120                 cmb_employee_type.Text = DR1.GetValue(3).ToString();
121                 txt_additional_employees.Text = DR1.GetValue(4).ToString();
122                 cmb_equipment_type.Text = DR1.GetValue(5).ToString();
123                 txt_additional_equipments.Text = DR1.GetValue(6).ToString();
124             }
125             DR1.Close();
126             con.Close();
127         }
```



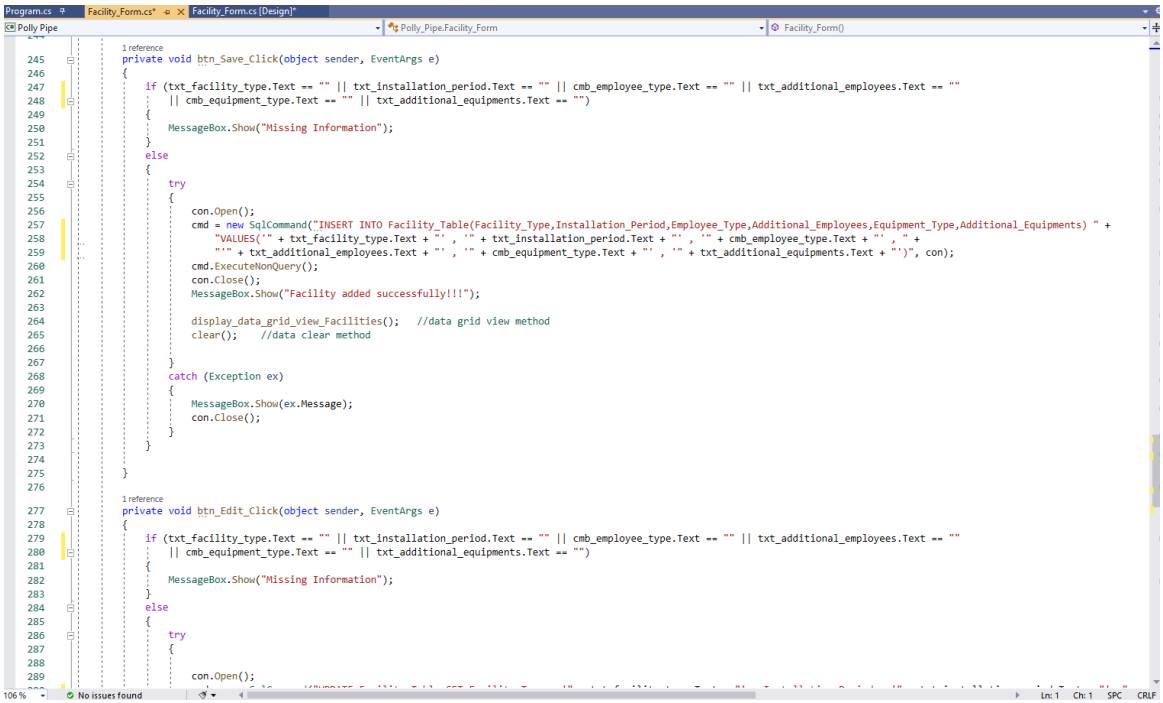
```
126             con.Close();
127         }
128
129
130         public void display_data_grid_view_Employee_Types()    //For the data grid view of Employee Types
131         {
132             try
133             {
134                 dt = new DataTable();
135                 con.Open();
136                 adpt = new SqlDataAdapter("SELECT * FROM Employee_Type_Table", con);
137                 adpt.Fill(dt);
138                 dgv_employee_types.DataSource = dt;
139                 con.Close();
140             }
141             catch (Exception ex)
142             {
143                 MessageBox.Show(ex.Message);
144                 con.Close();
145             }
146         }
147
148         private void dgv_employee_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
149         {
150             con.Open();
151             int ID;
152
153             ID = int.Parse(dgv_employee_types.Rows[e.RowIndex].Cells[0].Value.ToString());
154
155             SqlCommand cmd = con.CreateCommand();
156             cmd.CommandType = CommandType.Text;
157             cmd.CommandText = "SELECT * FROM Employee_Type_Table WHERE Employee_Type_ID = '" + ID + "' ";
158
159             SqlDataReader DR1 = cmd.ExecuteReader();
160
161             if (DR1.Read())
162             {
163                 cmb_employee_type.Text = DR1.GetValue(0).ToString();
164             }
165             DR1.Close();
166             con.Close();
167         }
168
169         public void display_data_grid_view_Equipment_Types()    //For the data grid view of Equipment Types
170         {
```



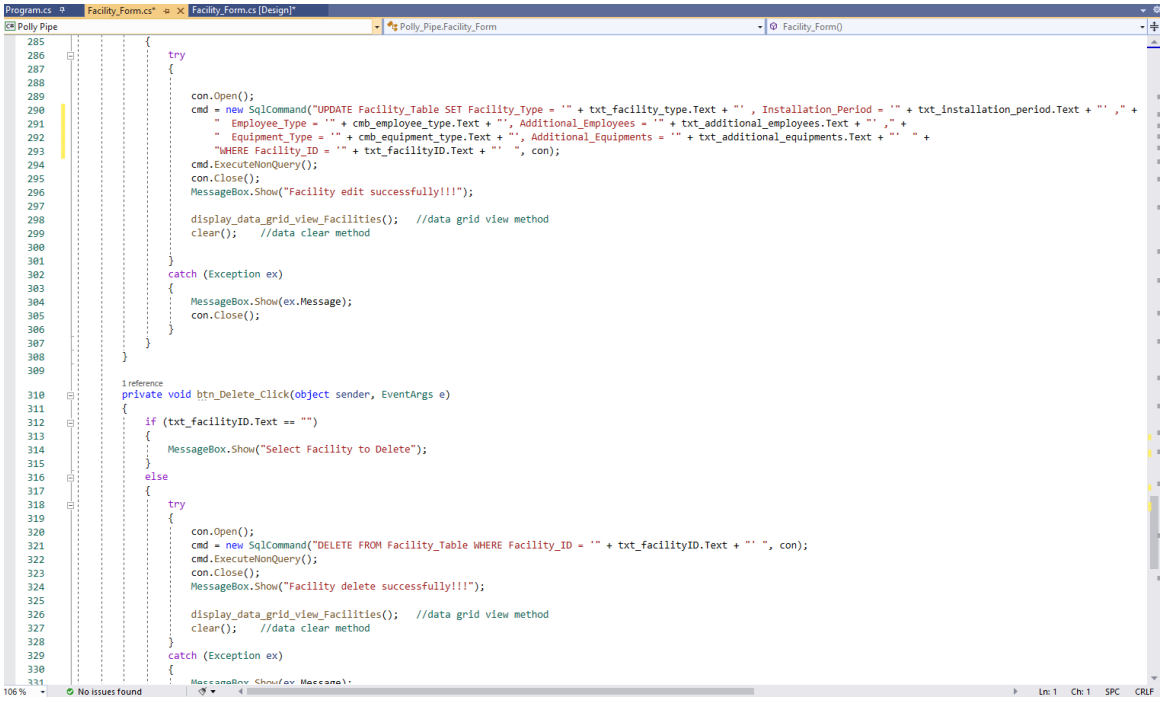
```
169         public void display_data_grid_view_Equipment_Types()    //For the data grid view of Equipment Types
170         {
171             try
172             {
173                 dt = new DataTable();
174                 con.Open();
175                 adpt = new SqlDataAdapter("SELECT * FROM Equipments_Table", con);
176                 adpt.Fill(dt);
177                 dgv_equipment_types.DataSource = dt;
178                 con.Close();
179             }
180             catch (Exception ex)
181             {
182                 MessageBox.Show(ex.Message);
183                 con.Close();
184             }
185         }
186
187         private void dgv_equipment_types_CellContentClick(object sender, DataGridViewCellEventArgs e)
188         {
189             con.Open();
190             int ID;
191
192             ID = int.Parse(dgv_equipment_types.Rows[e.RowIndex].Cells[0].Value.ToString());
193
194             SqlCommand cmd = con.CreateCommand();
195             cmd.CommandType = CommandType.Text;
196             cmd.CommandText = "SELECT * FROM Equipments_Table WHERE Equipment_ID = '" + ID + "' ";
197
198             SqlDataReader DR1 = cmd.ExecuteReader();
199
200             if (DR1.Read())
201             {
202                 cmb_equipment_type.Text = DR1.GetValue(0).ToString();
203             }
204             DR1.Close();
205             con.Close();
206         }
207
208         // Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Cc
209
210         private void fillcombo_EmployeeType_ID()    //ComboBox fill with Facility IDs
211         {
212             con.Open();
213             SqlCommand cmd = new SqlCommand("SELECT Employee_Type_ID FROM Employee_Type_Table", con);
```



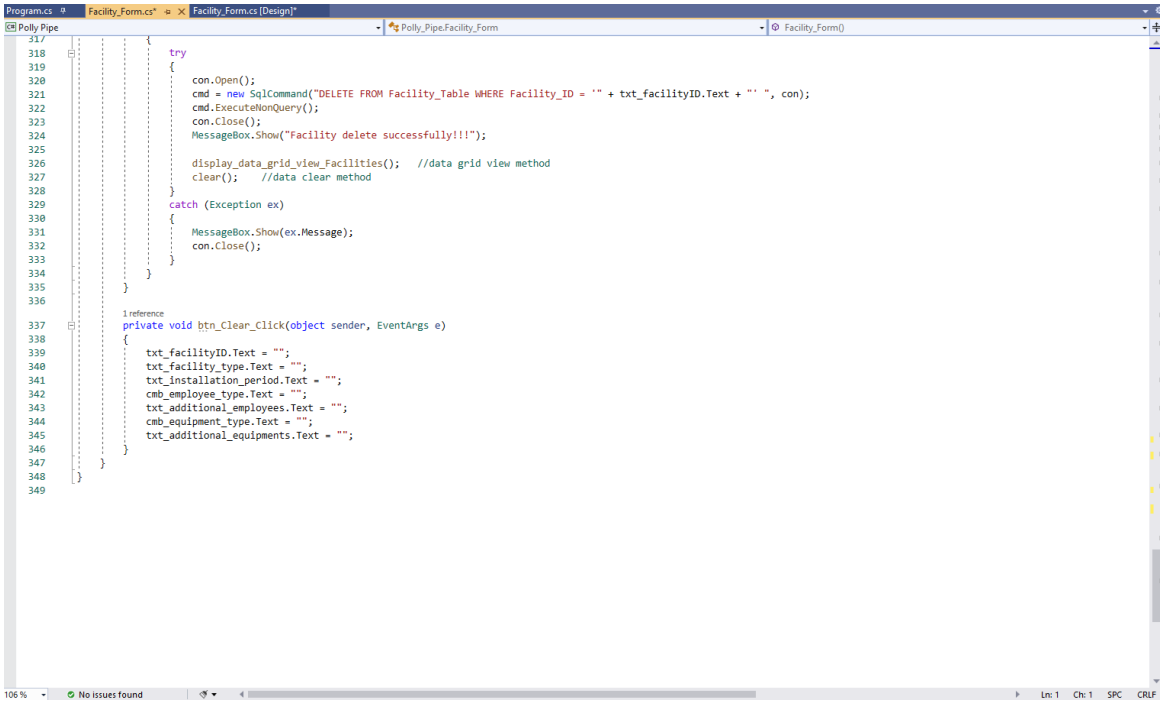
```
// Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID ComboBox Fill ******* Employee Type ID Co

1 reference
private void fillcombo_EmployeeType_ID()    //ComboBox fill with Facility IDs
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Employee_Type_ID FROM Employee_Type_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_employee_type", typeof(int));
    dt.Load(dr);
    cmb_employee_type.ValueMember = "Employee_Type_ID";
    cmb_employee_type.DataSource = dt;
    con.Close();
}



// Equipment Type ID ComboBox Fill ******* Equipment Type ID ComboBox Fill ******* Equipment Type ID ComboBox Fill ******* Equipment Type ID ComboBox Fill ******* Equipment Type

1 reference
private void fillcombo_EquipmentType_ID()    //ComboBox fill with Facility IDs
{
    con.Open();
    SqlCommand cmd = new SqlCommand("SELECT Equipment_ID FROM Equipments_Table", con);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Columns.Add("cmb_equipment_type", typeof(int));
    dt.Load(dr);
    cmb_equipment_type.ValueMember = "Equipment_ID";
    cmb_equipment_type.DataSource = dt;
    con.Close();
}


//*********************************SAVE***********Edit**************Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_facility_type.Text == "" || txt_installation_period.Text == "" || cmb_employee_type.Text == "" || txt_additional_employees.Text == ""
        || cmb_equipment_type.Text == "" || txt_additional_equipments.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
```



```
1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_facility_type.Text == "" || txt_installation_period.Text == "" || cmb_employee_type.Text == "" || txt_additional_employees.Text == ""
        || cmb_equipment_type.Text == "" || txt_additional_equipments.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Facility_Table(Facility_Type,Installation_Period,Employee_Type,Additional_Employees,Equipment_Type,Additional_Equipments) " +
                "VALUES('" + txt_facility_type.Text + "' , '" + txt_installation_period.Text + "' , '" + cmb_employee_type.Text + "' , " +
                "'" + txt_additional_employees.Text + "' , '" + cmb_equipment_type.Text + "' , '" + txt_additional_equipments.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Facility added successfully!!!");

            display_data_grid_view_Facilities();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            con.Close();
        }
    }

}

1 reference
private void btn_Edit_Click(object sender, EventArgs e)
{
    if (txt_facility_type.Text == "" || txt_installation_period.Text == "" || cmb_employee_type.Text == "" || txt_additional_employees.Text == ""
        || cmb_equipment_type.Text == "" || txt_additional_equipments.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {

            con.Open();
```

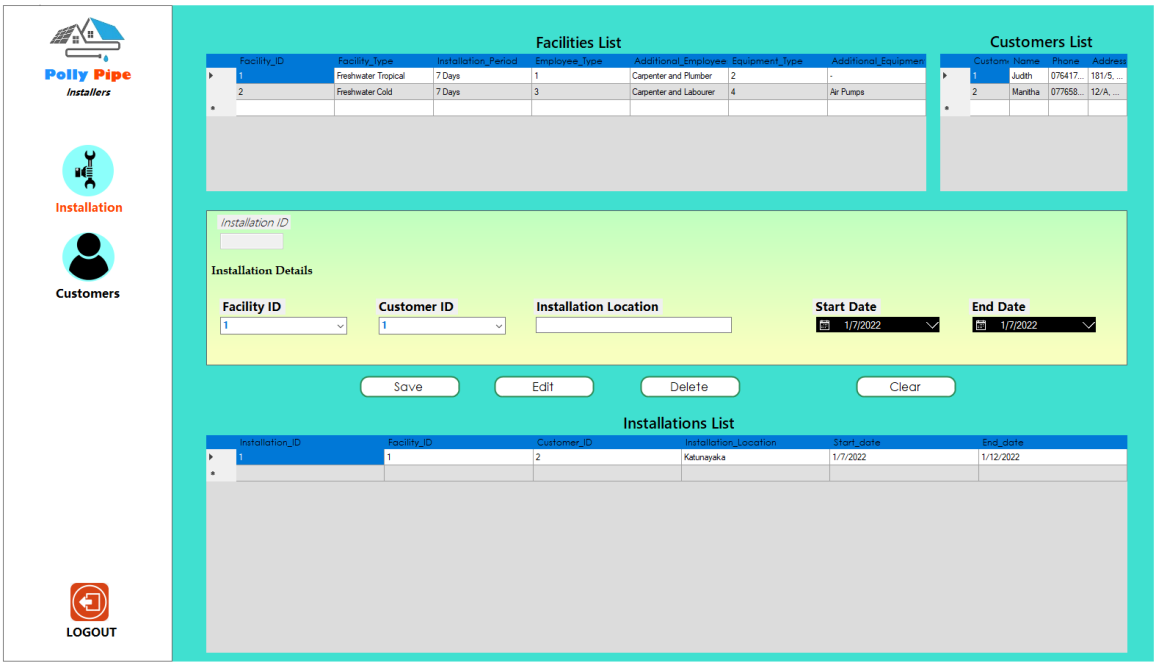


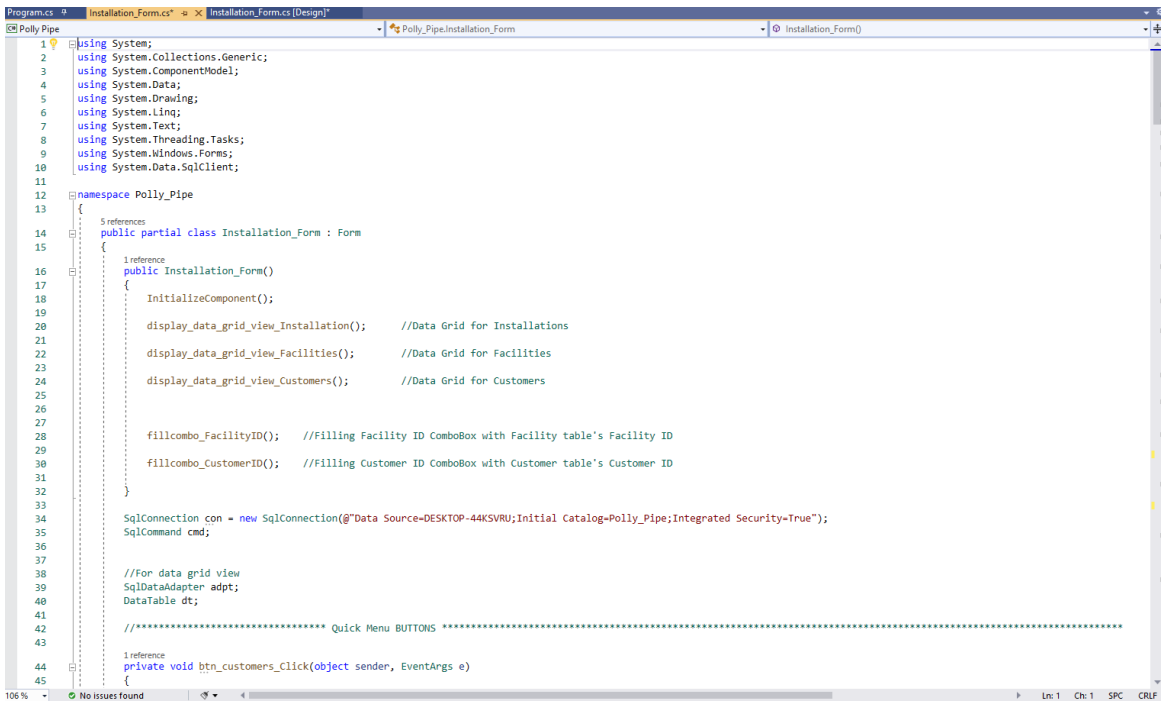
```
Program.cs | Facility_Form.cs* | Facility_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Facility_Form | Facility_Form()

285             {
286                 try
287                 {
288
289                     con.Open();
290                     cmd = new SqlCommand("UPDATE Facility_Table SET Facility_Type = '" + txt_facility_type.Text + "' , Installation_Period = '" + txt_installation_period.Text + "' ," +
291                         "  Employee_Type = '" + cmb_employee_type.Text + "', Additional_Employees = '" + txt_additional_employees.Text + "' ," +
292                         "  Equipment_Type = '" + cmb_equipment_type.Text + "', Additional_Equipments = '" + txt_additional_equipments.Text + "'  " +
293                         "WHERE Facility_ID = '" + txt_facilityID.Text + "'  ", con);
294                     cmd.ExecuteNonQuery();
295                     con.Close();
296                     MessageBox.Show("Facility edit successfully!!!");
297
298                     display_data_grid_view_Facilities();   //data grid view method
299                     clear();    //data clear method
300
301                 }
302                 catch (Exception ex)
303                 {
304                     MessageBox.Show(ex.Message);
305                     con.Close();
306                 }
307             }
308         }
309
310         1 reference
311         private void btn_Delete_Click(object sender, EventArgs e)
312         {
313             if (txt_facilityID.Text == "")
314             {
315                 MessageBox.Show("Select Facility to Delete");
316             }
317             else
318             {
319                 try
320                 {
321                     con.Open();
322                     cmd = new SqlCommand("DELETE FROM Facility_Table WHERE Facility_ID = '" + txt_facilityID.Text + "' ", con);
323                     cmd.ExecuteNonQuery();
324                     con.Close();
325                     MessageBox.Show("Facility delete successfully!!!");
326
327                     display_data_grid_view_Facilities();   //data grid view method
328                     clear();    //data clear method
329                 }
330                 catch (Exception ex)
331                 {
                    MessageBox.Show(ex.Message);

106 %  No issues found                                                    Ln: 1  Ch: 1  SPC  CRLF
```



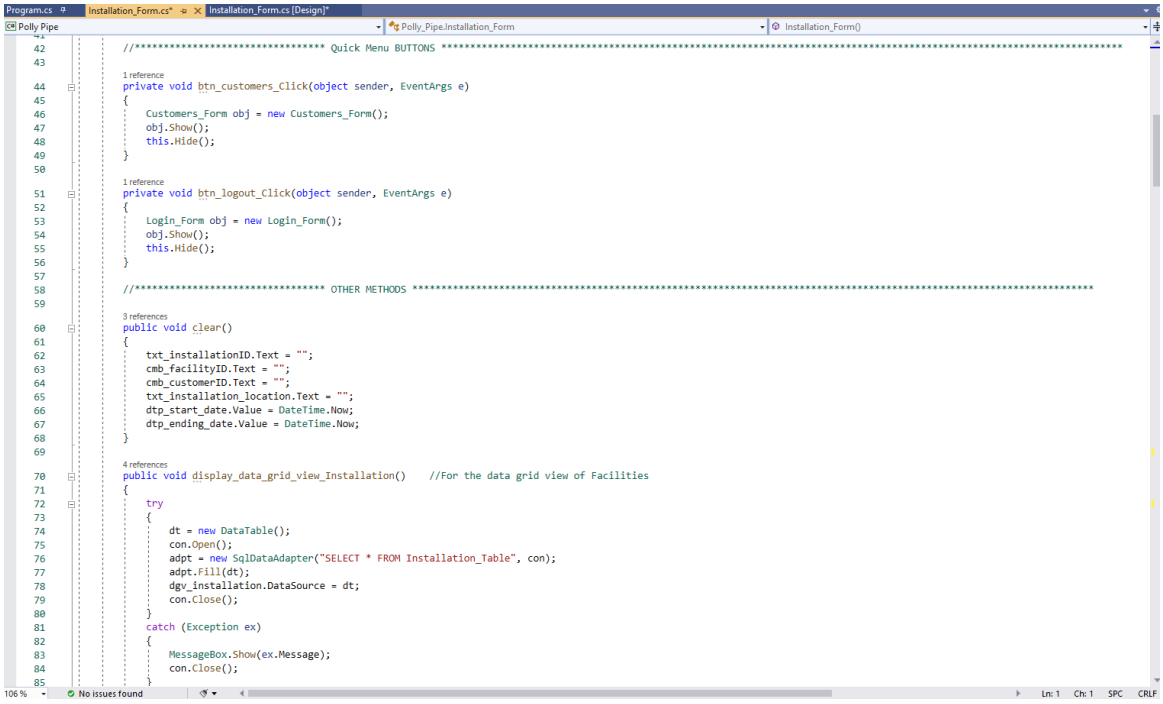
```
Program.cs | Facility_Form.cs* | Facility_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Facility_Form | Facility_Form()

317             {
318                 try
319                 {
320                     con.Open();
321                     cmd = new SqlCommand("DELETE FROM Facility_Table WHERE Facility_ID = '" + txt_facilityID.Text + "' ", con);
322                     cmd.ExecuteNonQuery();
323                     con.Close();
324                     MessageBox.Show("Facility delete successfully!!!");
325
326                     display_data_grid_view_Facilities();   //data grid view method
327                     clear();    //data clear method
328                 }
329                 catch (Exception ex)
330                 {
331                     MessageBox.Show(ex.Message);
332                     con.Close();
333                 }
334             }
335         }
336
337         1 reference
338         private void btn_Clear_Click(object sender, EventArgs e)
339         {
340             txt_facilityID.Text = "";
341             txt_facility_type.Text = "";
342             txt_installation_period.Text = "";
343             cmb_employee_type.Text = "";
344             txt_additional_employees.Text = "";
345             cmb_equipment_type.Text = "";
346             txt_additional_equipments.Text = "";
347         }
348     }
349 }

106 %  No issues found                                                    Ln: 1  Ch: 1  SPC  CRLF
```

Installation Form (For Sales Representative) →

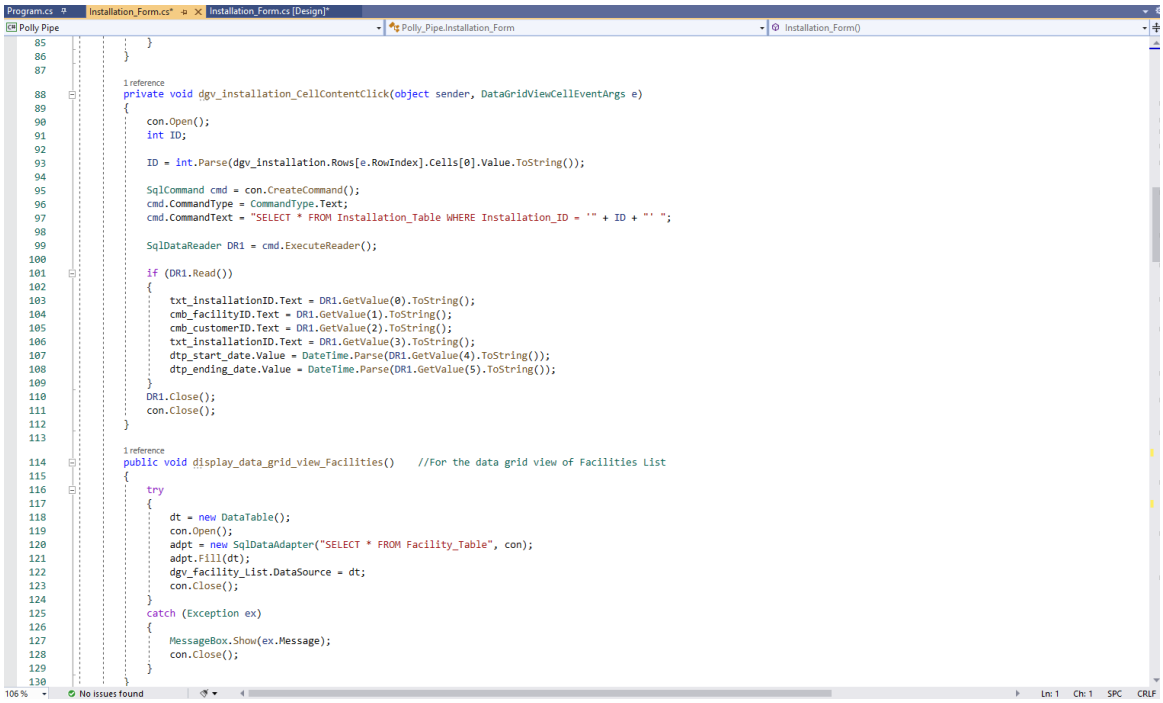




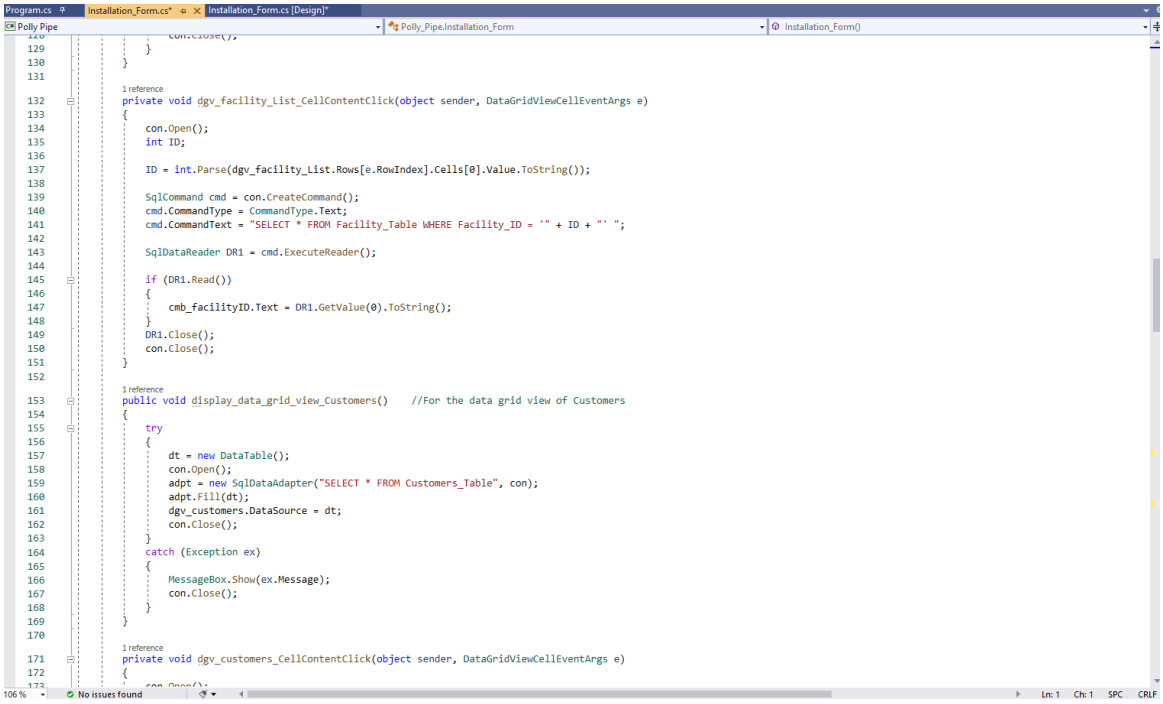
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    5 references
    public partial class Installation_Form : Form
    {
        1 reference
        public Installation_Form()
        {
            InitializeComponent();

            display_data_grid_view_Installation();      //Data Grid for Installations

            display_data_grid_view_Facilities();        //Data Grid for Facilities

            display_data_grid_view_Customers();         //Data Grid for Customers


            fillcombo_FacilityID();    //Filling Facility ID ComboBox with Facility table's Facility ID

            fillcombo_CustomerID();    //Filling Customer ID ComboBox with Customer table's Customer ID

        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;

        //******************************** Quick Menu BUTTONS ********************************************************************************************************

        1 reference
        private void btn_customers_Click(object sender, EventArgs e)
        {
```



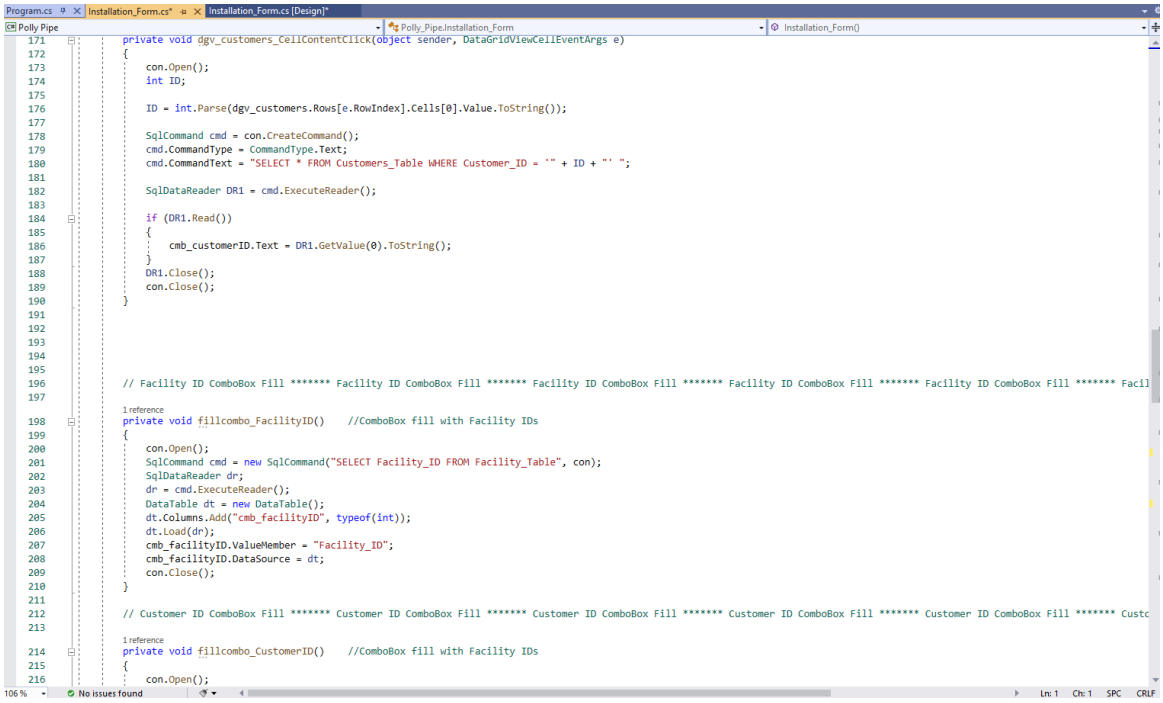
```
//********************************* Quick Menu BUTTONS ***********************************************************************************************************

1 reference
private void btn_customers_Click(object sender, EventArgs e)
{
    Customers_Form obj = new Customers_Form();
    obj.Show();
    this.Hide();
}

1 reference
private void btn_logout_Click(object sender, EventArgs e)
{
    Login_Form obj = new Login_Form();
    obj.Show();
    this.Hide();
}

//********************************* OTHER METHODS ****************************************************************************************************************

3 references
public void clear()
{
    txt_installationID.Text = "";
    cmb_facilityID.Text = "";
    cmb_customerID.Text = "";
    txt_installation_location.Text = "";
    dtp_start_date.Value = DateTime.Now;
    dtp_ending_date.Value = DateTime.Now;
}

4 references
public void display_data_grid_view_Installation()    //For the data grid view of Facilities
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Installation_Table", con);
        adpt.Fill(dt);
        dgv_installation.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
```



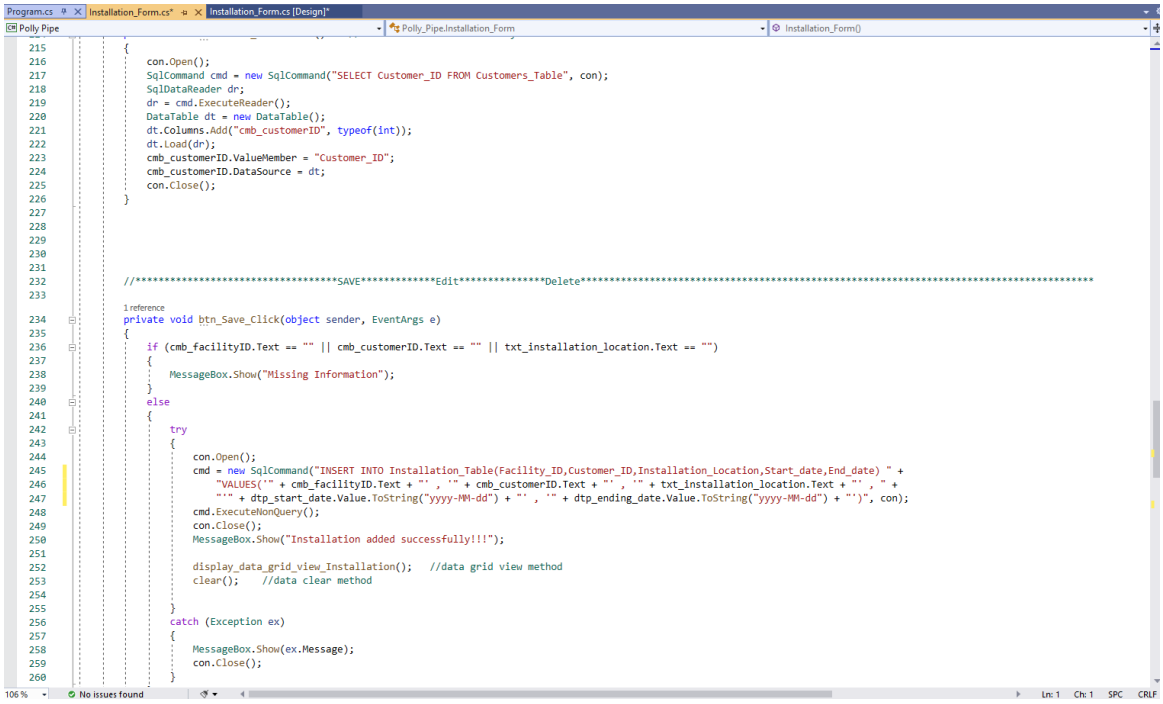
```
    }
}

1 reference
private void dgv_installation_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_installation.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Installation_Table WHERE Installation_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_installationID.Text = DR1.GetValue(0).ToString();
        cmb_facilityID.Text = DR1.GetValue(1).ToString();
        cmb_customerID.Text = DR1.GetValue(2).ToString();
        txt_installationID.Text = DR1.GetValue(3).ToString();
        dtp_start_date.Value = DateTime.Parse(DR1.GetValue(4).ToString());
        dtp_ending_date.Value = DateTime.Parse(DR1.GetValue(5).ToString());
    }
    DR1.Close();
    con.Close();
}

1 reference
public void display_data_grid_view_Facilities()    //For the data grid view of Facilities List
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Facility_Table", con);
        adpt.Fill(dt);
        dgv_facility_List.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
```



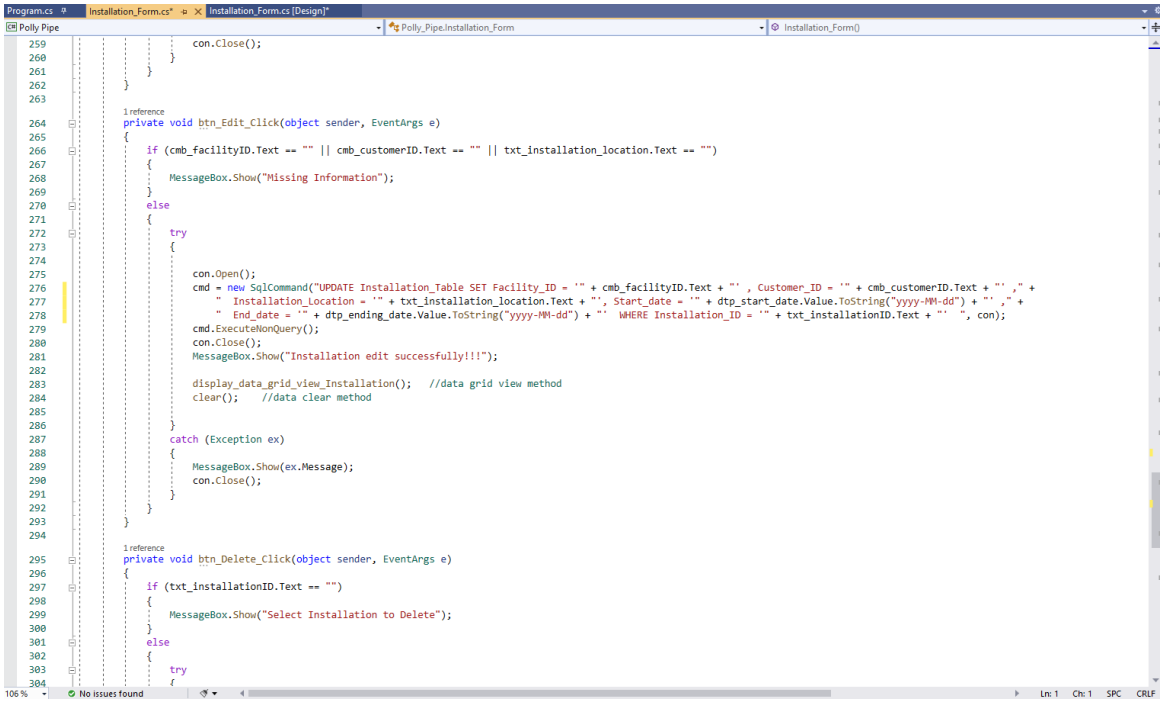
```
Program.cs | Installation_Form.cs* | Installation_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Installation_Form | Installation_Form()
129         }
130     }
131
        1 reference
132     private void dgv_facility_List_CellContentClick(object sender, DataGridViewCellEventArgs e)
133     {
134         con.Open();
135         int ID;
136
137         ID = int.Parse(dgv_facility_List.Rows[e.RowIndex].Cells[0].Value.ToString());
138
139         SqlCommand cmd = con.CreateCommand();
140         cmd.CommandType = CommandType.Text;
141         cmd.CommandText = "SELECT * FROM Facility_Table WHERE Facility_ID = '" + ID + "' ";
142
143         SqlDataReader DR1 = cmd.ExecuteReader();
144
145         if (DR1.Read())
146         {
147             cmb_facilityID.Text = DR1.GetValue(0).ToString();
148         }
149         DR1.Close();
150         con.Close();
151     }
152
        1 reference
153     public void display_data_grid_view_Customers()    //For the data grid view of Customers
154     {
155         try
156         {
157             dt = new DataTable();
158             con.Open();
159             adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
160             adpt.Fill(dt);
161             dgv_customers.DataSource = dt;
162             con.Close();
163         }
164         catch (Exception ex)
165         {
166             MessageBox.Show(ex.Message);
167             con.Close();
168         }
169     }
170
        1 reference
171     private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
172     {
173         con.Open();
106 %  No issues found   Ln: 1  Ch: 1  SPC  CRLF
```



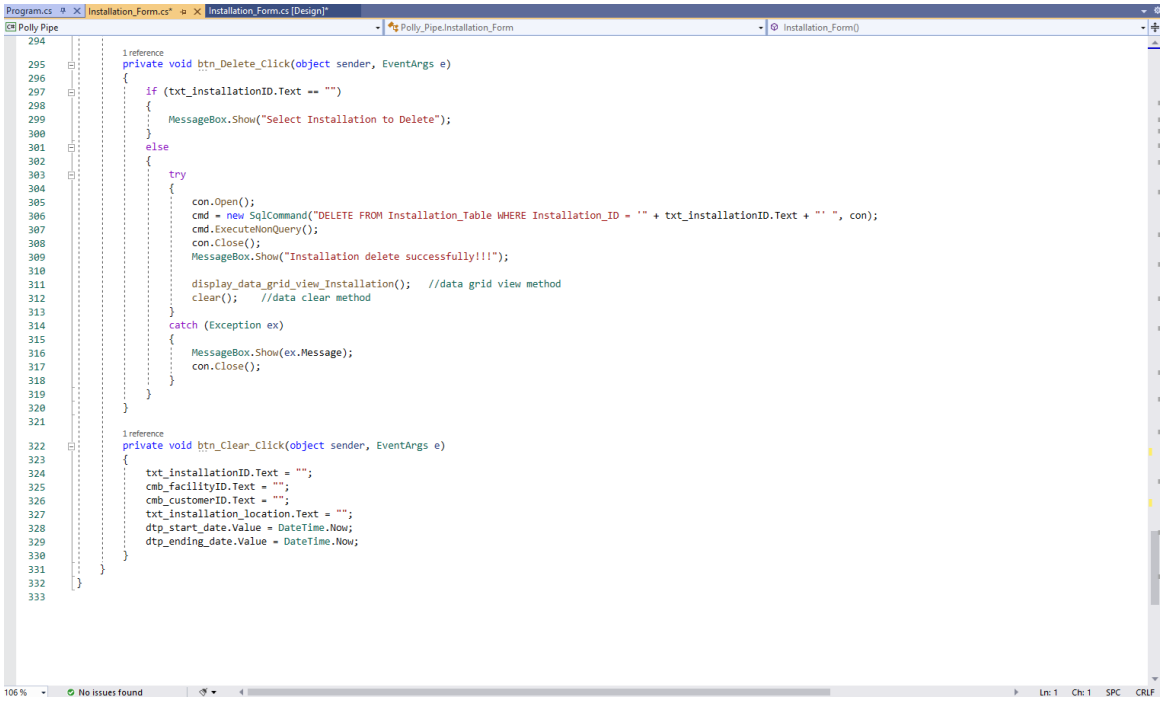
```
Program.cs | Installation_Form.cs* | Installation_Form.cs [Design]*
Polly Pipe | Polly_Pipe.Installation_Form | Installation_Form()
171     private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
172     {
173         con.Open();
174         int ID;
175
176         ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());
177
178         SqlCommand cmd = con.CreateCommand();
179         cmd.CommandType = CommandType.Text;
180         cmd.CommandText = "SELECT * FROM Customers_Table WHERE Customer_ID = '" + ID + "' ";
181
182         SqlDataReader DR1 = cmd.ExecuteReader();
183
184         if (DR1.Read())
185         {
186             cmb_customerID.Text = DR1.GetValue(0).ToString();
187         }
188         DR1.Close();
189         con.Close();
190     }
191
192
193
194
195
196     // Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facility ID ComboBox Fill ******* Facil
197
        1 reference
198     private void fillcombo_FacilityID()    //ComboBox fill with Facility IDs
199     {
200         con.Open();
201         SqlCommand cmd = new SqlCommand("SELECT Facility_ID FROM Facility_Table", con);
202         SqlDataReader dr;
203         dr = cmd.ExecuteReader();
204         DataTable dt = new DataTable();
205         dt.Columns.Add("cmb_facilityID", typeof(int));
206         dt.Load(dr);
207         cmb_facilityID.ValueMember = "Facility_ID";
208         cmb_facilityID.DataSource = dt;
209         con.Close();
210     }
211
212     // Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Customer ID ComboBox Fill ******* Custc
213
        1 reference
214     private void fillcombo_CustomerID()    //ComboBox fill with Facility IDs
215     {
216         con.Open();
106 %  No issues found   Ln: 1  Ch: 1  SPC  CRLF
```



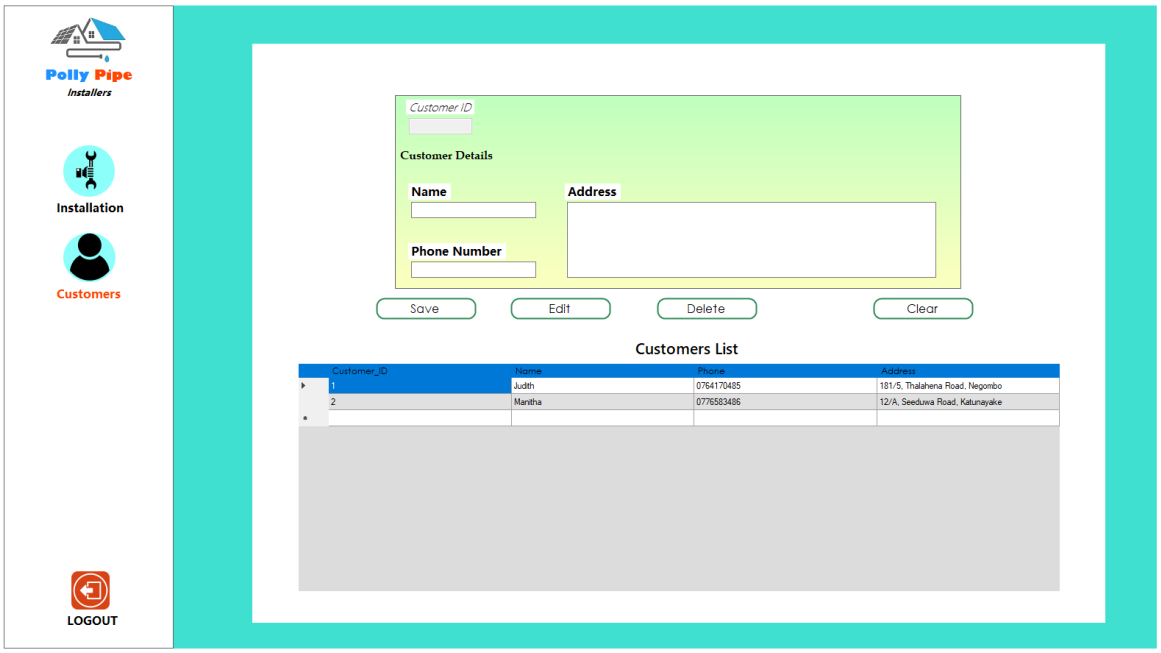
```
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT Customer_ID FROM Customers_Table", con);
            SqlDataReader dr;
            dr = cmd.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Columns.Add("cmb_customerID", typeof(int));
            dt.Load(dr);
            cmb_customerID.ValueMember = "Customer_ID";
            cmb_customerID.DataSource = dt;
            con.Close();
        }




        //**********************************SAVE************Edit**************Delete*****************************************************************************************

        1 reference
        private void btn_Save_Click(object sender, EventArgs e)
        {
            if (cmb_facilityID.Text == "" || cmb_customerID.Text == "" || txt_installation_location.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {
                    con.Open();
                    cmd = new SqlCommand("INSERT INTO Installation_Table(Facility_ID,Customer_ID,Installation_Location,Start_date,End_date) " +
                        "VALUES('" + cmb_facilityID.Text + "' , '" + cmb_customerID.Text + "' , '" + txt_installation_location.Text + "' , " +
                        "'" + dtp_start_date.Value.ToString("yyyy-MM-dd") + "' , '" + dtp_ending_date.Value.ToString("yyyy-MM-dd") + "')", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Installation added successfully!!!");

                    display_data_grid_view_Installation();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
```



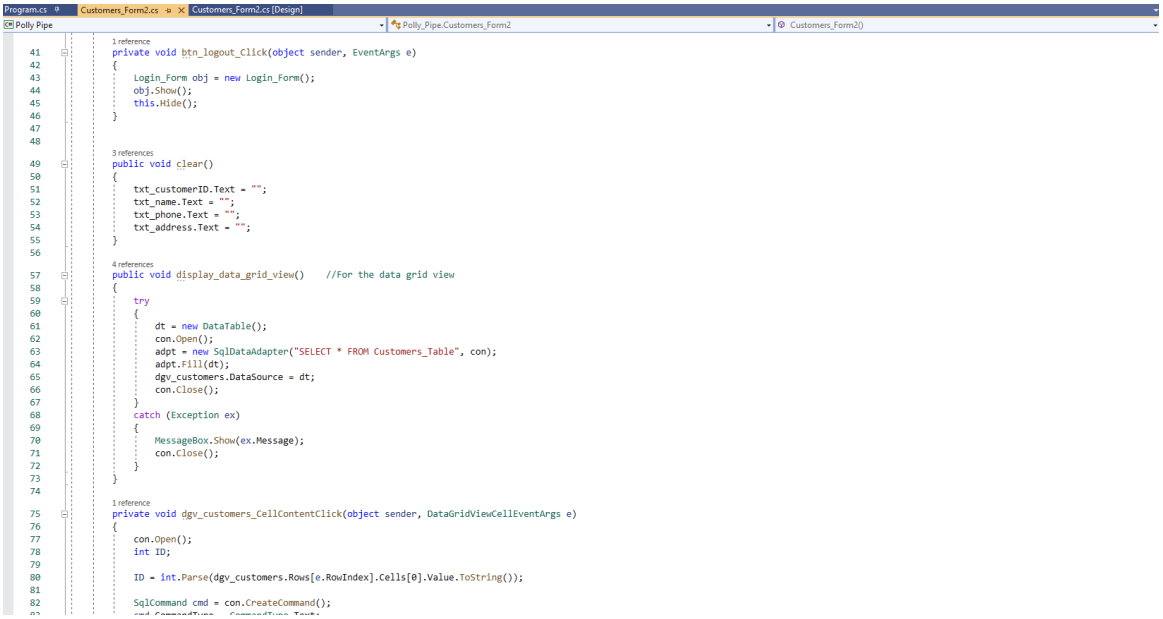
```
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Edit_Click(object sender, EventArgs e)
        {
            if (cmb_facilityID.Text == "" || cmb_customerID.Text == "" || txt_installation_location.Text == "")
            {
                MessageBox.Show("Missing Information");
            }
            else
            {
                try
                {

                    con.Open();
                    cmd = new SqlCommand("UPDATE Installation_Table SET Facility_ID = '" + cmb_facilityID.Text + "' , Customer_ID = '" + cmb_customerID.Text + "' ," +
                        "  Installation_Location = '" + txt_installation_location.Text + "', Start_date = '" + dtp_start_date.Value.ToString("yyyy-MM-dd") + "' ," +
                        "  End_date = '" + dtp_ending_date.Value.ToString("yyyy-MM-dd") + "'  WHERE Installation_ID = '" + txt_installationID.Text + "'  ", con);
                    cmd.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show("Installation edit successfully!!!");

                    display_data_grid_view_Installation();   //data grid view method
                    clear();    //data clear method

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                    con.Close();
                }
            }
        }

        1 reference
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            if (txt_installationID.Text == "")
            {
                MessageBox.Show("Select Installation to Delete");
            }
            else
            {
                try
```


Program.cs
Installation_Form.cs*
Installation_Form.cs [Design]*
Polly Pipe
Polly_Pipe.Installation_Form
Installation_Form()
1 reference
private void btn_Delete_Click(object sender, EventArgs e)
{
if (txt_installationID.Text == "")
{
MessageBox.Show("Select Installation to Delete");
}
else
{
try
{
con.Open();
cmd = new SqlCommand("DELETE FROM Installation_Table WHERE Installation_ID = '" + txt_installationID.Text + "' ", con);
cmd.ExecuteNonQuery();
con.Close();
MessageBox.Show("Installation delete successfully!!!");
display_data_grid_view_Installation(); //data grid view method
clear(); //data clear method
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
con.Close();
}
}
}
1 reference
private void btn_Clear_Click(object sender, EventArgs e)
{
txt_installationID.Text = "";
cmb_facilityID.Text = "";
cmb_customerID.Text = "";
txt_installation_location.Text = "";
dtp_start_date.Value = DateTime.Now;
dtp_ending_date.Value = DateTime.Now;
}
}
}
106 %
No issues found
Ln: 1
Ch: 1
SPC
CRLF

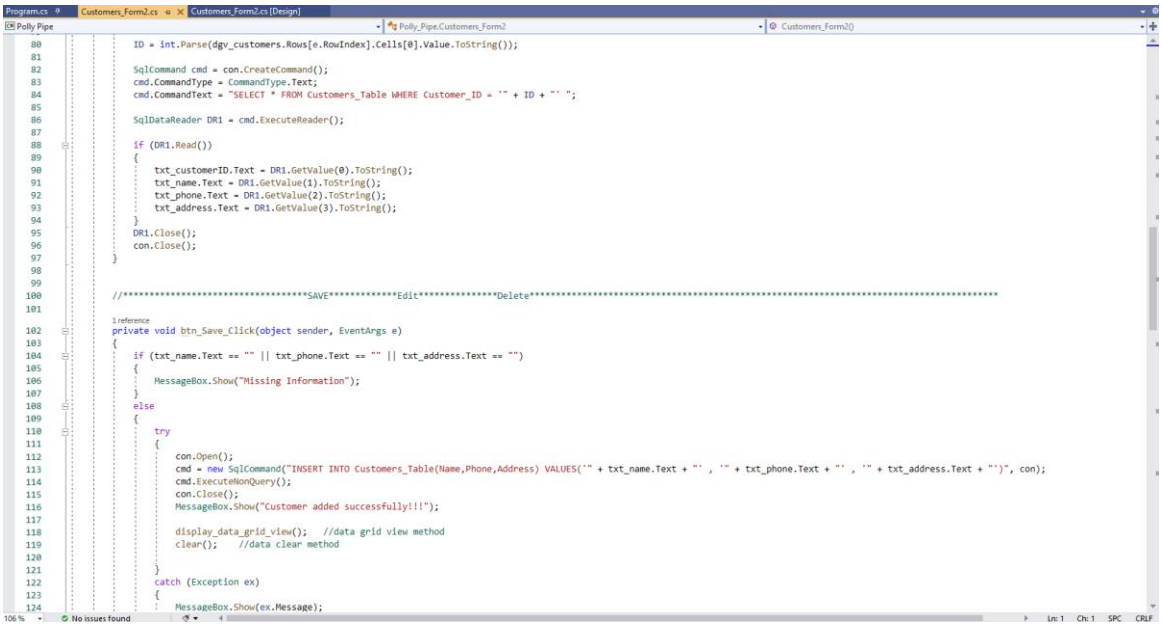
Customer Form (For Sales Representative) →



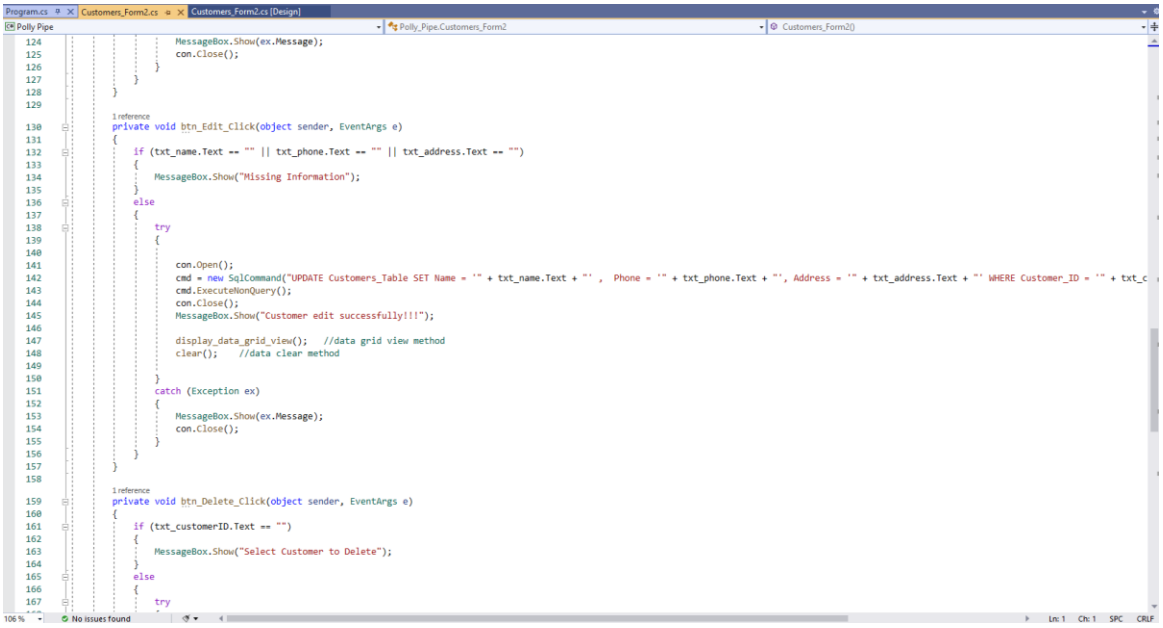

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Polly_Pipe
{
    public partial class Customers_Form2 : Form
    {
        public Customers_Form2()
        {
            InitializeComponent();

            display_data_grid_view();   //Data Grid for Customers Table
        }

        SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-44KSVRU;Initial Catalog=Polly_Pipe;Integrated Security=True");
        SqlCommand cmd;


        //For data grid view
        SqlDataAdapter adpt;
        DataTable dt;


        //********************************* Quick Menu BUTTONS ***********************************************************************************************************

        private void btn_installation_Click(object sender, EventArgs e)
        {
            Login_Form obj = new Login_Form();
            obj.Show();
            this.Hide();
        }

        private void btn_logout_Click(object sender, EventArgs e)
        {
```



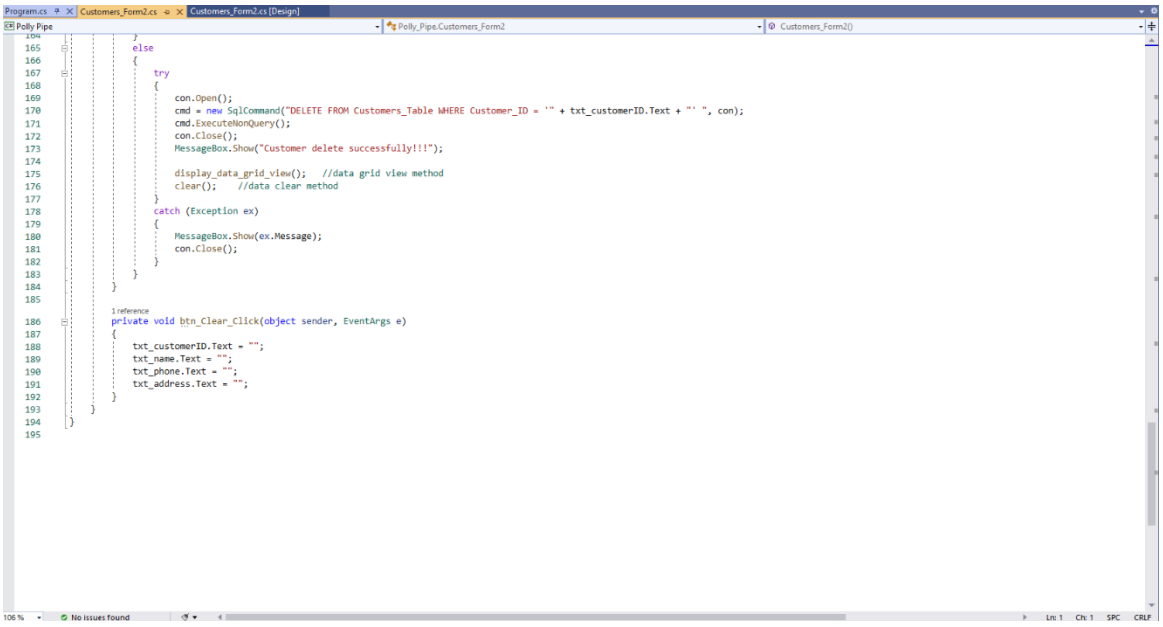
```
Program.cs | Customers_Form2.cs | Customers_Form2.cs [Design]
Polly Pipe | Polly_Pipe.Customers_Form2 | Customers_Form2()

1 reference
private void btn_logout_Click(object sender, EventArgs e)
{
    Login_Form obj = new Login_Form();
    obj.Show();
    this.Hide();
}


3 references
public void clear()
{
    txt_customerID.Text = "";
    txt_name.Text = "";
    txt_phone.Text = "";
    txt_address.Text = "";
}

4 references
public void display_data_grid_view()    //For the data grid view
{
    try
    {
        dt = new DataTable();
        con.Open();
        adpt = new SqlDataAdapter("SELECT * FROM Customers_Table", con);
        adpt.Fill(dt);
        dgv_customers.DataSource = dt;
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
        con.Close();
    }
}

1 reference
private void dgv_customers_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    con.Open();
    int ID;

    ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
```



```
Program.cs | Customers_Form2.cs | Customers_Form2.cs [Design]
Polly Pipe | Polly_Pipe.Customers_Form2 | Customers_Form2()

    ID = int.Parse(dgv_customers.Rows[e.RowIndex].Cells[0].Value.ToString());

    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "SELECT * FROM Customers_Table WHERE Customer_ID = '" + ID + "' ";

    SqlDataReader DR1 = cmd.ExecuteReader();

    if (DR1.Read())
    {
        txt_customerID.Text = DR1.GetValue(0).ToString();
        txt_name.Text = DR1.GetValue(1).ToString();
        txt_phone.Text = DR1.GetValue(2).ToString();
        txt_address.Text = DR1.GetValue(3).ToString();
    }
    DR1.Close();
    con.Close();
}


//*********************************SAVE***********Edit**********Delete*****************************************************************************************

1 reference
private void btn_Save_Click(object sender, EventArgs e)
{
    if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
    {
        MessageBox.Show("Missing Information");
    }
    else
    {
        try
        {
            con.Open();
            cmd = new SqlCommand("INSERT INTO Customers_Table(Name,Phone,Address) VALUES('" + txt_name.Text + "' , '" + txt_phone.Text + "' , '" + txt_address.Text + "')", con);
            cmd.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Customer added successfully!!!");

            display_data_grid_view();   //data grid view method
            clear();    //data clear method

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
106 %  No issues found                                                                                    Ln: 1  Ch: 1  SPC  CRLF
```



```
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Edit_Click(object sender, EventArgs e)
    {
        if (txt_name.Text == "" || txt_phone.Text == "" || txt_address.Text == "")
        {
            MessageBox.Show("Missing Information");
        }
        else
        {
            try
            {

                con.Open();
                cmd = new SqlCommand("UPDATE Customers_Table SET Name = '" + txt_name.Text + "' ,  Phone = '" + txt_phone.Text + "', Address = '" + txt_address.Text + "' WHERE Customer_ID = '" + txt_c
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Customer edit successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Delete_Click(object sender, EventArgs e)
    {
        if (txt_customerID.Text == "")
        {
            MessageBox.Show("Select Customer to Delete");
        }
        else
        {
            try
```



```
        }
        else
        {
            try
            {
                con.Open();
                cmd = new SqlCommand("DELETE FROM Customers_Table WHERE Customer_ID = '" + txt_customerID.Text + "' ", con);
                cmd.ExecuteNonQuery();
                con.Close();
                MessageBox.Show("Customer delete successfully!!!");

                display_data_grid_view();   //data grid view method
                clear();    //data clear method
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                con.Close();
            }
        }
    }

    1 reference
    private void btn_Clear_Click(object sender, EventArgs e)
    {
        txt_customerID.Text = "";
        txt_name.Text = "";
        txt_phone.Text = "";
        txt_address.Text = "";
    }
}
```

4.3.4 System Requirements

Software Requirements :-

- Microsoft Visual Studio 2019
- Microsoft SQL Server Management Studio 2018
- Windows operating system

Hardware Requirements :-

- Any basic computers which having Windows Operating System.

4.3.5 Future Enhancements

Managing software quality has become an essential component of project management at all levels. Delivering projects with high-quality software will be more cost-effective and efficient. Finding ways to adopt effective testing strategies as soon as possible will aid in the detection and resolution of issues.

- Make a cloud-based database :- Moving the database to the cloud, as cloud services develop and become more economical, can provide flexible, affordable, and scalable database management. A reliable and efficient database connection is required for Windows applications. A functional, flexible, and secure database serves as the foundation for developing interesting application.
- Ability to edit main ID fields :- And also, should develop the system to giving capability of editing Identity fields because sometimes user would be needed to change the ID number of an element.
- Create a dashboard :- Creating a dashboard for the system is needed. The dashboard displays all data from across the organization in one place. A unique dashboard provides important insights into the entire business, allowing users to see and analyze data.
- Create a user sign up form :- Should create a new user sign up form is helping to register new admin or a user for system. Rather than depending on system developer for security purpose, this helps to enhance the security of the system further.
- Insert a search bar :- Should create search bar for each form of the system helps to find recorded data in the database.
- Should give ability to printout the records.
- Should give ability to export the records.

Conclusion

This entire assignment is based on an implementation of a Water Equipment installation system design for selected company (Polly Pipe). The purpose of this assignment is to improve database developing and management skills.

This report includes basic database software handling, creating designs, understanding data flows, understanding different database designing methods, working with IDEs, Basic system creation using Microsoft SQL and Visual Studio C# windows forms application, database normalization by finding and resolve anomalies, using different SQL query types when creating a database for a system, make a documentation for a system with describing future enhancements for the system.

References

Lucidchart. 2022. Entity-Relationship Diagram Symbols and Notation | Lucidchart.

[ONLINE] Available at: https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning#:~:text=Cardinality%20and%20ordinality,instance%20in%20the%20related%20entity..

[Accessed 2 February 2022].

FileMaker Pro 18 Advanced Help. 2022. FileMaker Pro 18 Advanced Help.

[ONLINE] Available at: https://fmhelp.filemaker.com/help/18/fmp/en/index.html#page/FMP_Help%2Fmany-to-many-relationships.html%23.

[Accessed 2 February 2022].

FileMaker Pro 18 Advanced Help. 2022. FileMaker Pro 18 Advanced Help.

[ONLINE] Available at: https://fmhelp.filemaker.com/help/18/fmp/en/index.html#page/FMP_Help/one-to-one-relationships.html.

[Accessed 2 February 2022].

FileMaker Pro 18 Advanced Help. 2022. FileMaker Pro 18 Advanced Help.

[ONLINE] Available at: https://fmhelp.filemaker.com/help/18/fmp/en/index.html#page/FMP_Help%2Fone-to-many-relationships.html%23.

[Accessed 2 February 2022].

FileMaker Pro 18 Advanced Help. 2022. FileMaker Pro 18 Advanced Help.

[ONLINE] Available at: https://fmhelp.filemaker.com/help/18/fmp/en/index.html#page/FMP_Help/many-to-many-relationships.html.

[Accessed 2 February 2022].

GeeksforGeeks. 2022. Weak Entity Set in ER diagrams - GeeksforGeeks.

[ONLINE] Available at: https://www.geeksforgeeks.org/weak-entity-set-in-er-diagrams/.

[Accessed 2 February 2022].

Visual Studio. 2022. Visual Studio IDE with .NET - Develop Any App Using C#, F#, VB.

[ONLINE] Available at: https://visualstudio.microsoft.com/vs/features/net-development/.

[Accessed 2 February 2022].

Comparing Logical and Physical ERD. 2022. Comparing Logical and Physical ERD.

[ONLINE] Available at: https://www.visual-paradigm.com/tutorials/compare-logical-physical-erd.jsp.

[Accessed 2 February 2022].

Hackr.io. 2022. Normalization in DBMS: 1NF, 2NF, 3NF and BCNF with Examples.

[ONLINE] Available at: https://hackr.io/blog/dbms-normalization.

[Accessed 2 February 2022].

www.javatpoint.com. 2022. DBMS Normalization: 1NF, 2NF, 3NF and BCNF with Examples - javatpoint.

[ONLINE] Available at: https://www.javatpoint.com/dbms-normalization#:~:text=Normalization%20is%20the%20process%20of,Insertion%2C%20Update%20and%20Deletion%20Anomalies..

[Accessed 2 February 2022].

GeeksforGeeks. 2022. Anomalies in Relational Model - GeeksforGeeks.

[ONLINE] Available at: https://www.geeksforgeeks.org/anomalies-in-relational-model/.

[Accessed 2 February 2022].

Boyce-Codd Normal Form (BCNF) of Database Normalization | Studytonight. 2022. Boyce-Codd Normal Form (BCNF) of Database Normalization | Studytonight.

[ONLINE] Available at: https://www.studytonight.com/dbms/boyce-codd-normal-form.php.

[Accessed 2 February 2022].

First Normal Form (1NF) of Database Normalization | Studytonight. 2022. First Normal Form (1NF) of Database Normalization | Studytonight.

[ONLINE] Available at: https://www.studytonight.com/dbms/first-normal-form.php.

[Accessed 8 February 2022].

SQL - Overview. 2022. SQL - Overview.

[ONLINE] Available at: https://www.tutorialspoint.com/sql/sql-overview.htm.

[Accessed 8 February 2022].

Alter Table Statement in SQL Server . 2022. Alter Table Statement in SQL Server .

[ONLINE] Available at: https://www.c-sharpcorner.com/blogs/alter-table-statement-in-sql-server1.

[Accessed 8 February 2022].

Difference between TRUNCATE and DELETE in SQL . 2022. Difference between TRUNCATE and DELETE in SQL .

[ONLINE] Available at: https://www.c-sharpcorner.com/blogs/difference-between-truncate-delete-and-drop-in-sql-server1.

[Accessed 8 February 2022].

DROP IF EXISTS In SQL Server 2016. 2022. DROP IF EXISTS In SQL Server 2016.

[ONLINE] Available at: https://www.c-sharpcorner.com/article/drop-if-exists-in-sql-server-2016/.

[Accessed 8 February 2022].

Types Of SQL Statements With Examples . 2022. Types Of SQL Statements With Examples .

[ONLINE] Available at: https://www.c-sharpcorner.com/blogs/types-of-sql-statements-with-example.

[Accessed 8 February 2022].

Type of SQL Statements. 2022. Type of SQL Statements.

[ONLINE] Available at: https://way2tutorial.com/sql/type-of-sql-statements.php.

[Accessed 8 February 2022].

www.javatpoint.com. 2022. SQL Commands: DDL, DML, DCL, TCL, DQL - javatpoint.

[ONLINE] Available at: https://www.javatpoint.com/dbms-sql-command.

[Accessed 8 February 2022].

study.com. 2022. No page title.

[ONLINE] Available at: https://study.com/academy/lesson/data-definition-language-ddl-definition-example.html.

[Accessed 8 February 2022].

Techopedia. 2022. What is Data Manipulation Language (DML)? - Definition from Techopedia.

[ONLINE] Available at: https://www.techopedia.com/definition/1179/data-manipulation-language-dml.

[Accessed 8 February 2022].

SQL GROUP BY Statement. 2022. SQL GROUP BY Statement.

[ONLINE] Available at: https://www.w3schools.com/sql/sql_groupby.asp.

[Accessed 8 February 2022].

Group By clause in SQL | Studytonight. 2022. Group By clause in SQL | Studytonight.

[ONLINE] Available at: https://www.studytonight.com/dbms/groupby-clause.php.

[Accessed 8 February 2022].

Pressbooks. 2022. Chapter 16 SQL Data Manipulation Language – Database Design – 2nd Edition .

[ONLINE] Available at: https://opentextbc.ca/dbdesign01/chapter/chapter-sql-dml/.

[Accessed 8 February 2022].

Type of SQL Statements. 2022. Type of SQL Statements.

[ONLINE] Available at: https://way2tutorial.com/sql/type-of-sql-statements.php.

[Accessed 11 February 2022].

www.javatpoint.com. 2022. SQL Commands: DDL, DML, DCL, TCL, DQL - javatpoint.

[ONLINE] Available at: https://www.javatpoint.com/dbms-sql-command.

[Accessed 11 February 2022].

Difference between TRUNCATE and DELETE in SQL . 2022. Difference between TRUNCATE and DELETE in SQL .

[ONLINE] Available at: https://www.c-sharpcorner.com/blogs/difference-between-truncate-delete-and-drop-in-sql-server1.

[Accessed 11 February 2022].

SELECT Statement in SQL Server. 2022. SELECT Statement in SQL Server.

[ONLINE] Available at: https://www.c-sharpcorner.com/UploadFile/219d4d/select-statement-in-sql-server/.

[Accessed 11 February 2022].

Types Of SQL Statements With Examples . 2022. Types Of SQL Statements With Examples .

[ONLINE] Available at: https://www.c-sharpcorner.com/blogs/types-of-sql-statements-with-example.

[Accessed 11 February 2022].

SQL HAVING Clause. 2022. SQL HAVING Clause.

[ONLINE] Available at: https://www.w3schools.com/sql/sql_having.asp.

[Accessed 11 February 2022].

SQL IN Operator. 2022. SQL IN Operator.

[ONLINE] Available at: https://www.w3schools.com/sql/sql_in.asp.

[Accessed 11 February 2022].

SQL BETWEEN Operator. 2022. SQL BETWEEN Operator.

[ONLINE] Available at: https://www.w3schools.com/sql/sql_between.asp.

[Accessed 11 February 2022].

Online Product Research Survey Template | QuickTapSurvey. 2022. Online Product Research Survey Template | QuickTapSurvey.

[ONLINE] Available at: https://www.quicktapsurvey.com/research/templates/new-product-research-survey-online.

[Accessed 11 February 2022].

Online Product Research Survey Template | QuickTapSurvey. 2022. Online Product Research Survey Template | QuickTapSurvey.

[ONLINE] Available at: https://www.quicktapsurvey.com/research/templates/new-product-research-survey-online.

[Accessed 28 February 2022].

Allie Decker. 2022. 11 Feedback Form Templates and Examples.

[ONLINE] Available at: https://blog.hubspot.com/website/feedback-form-template.

[Accessed 28 February 2022].

Boast. 2022. 5 Customer Feedback Form Templates To Improve Your Surveys - Boast.

[ONLINE] Available at: https://boast.io/5-customer-feedback-form-templates-to-improve-your-surveys/.

[Accessed 28 February 2022].

FormAssembly. 2022. Feedback Form Best Practices: 7 Tips | FormAssembly.

[ONLINE] Available at: https://www.formassembly.com/blog/feedback-form/.

[Accessed 28 February 2022].

Help Scout. 2022. Customer Feedback: Why It's Important + 7 Ways to Collect It.

[ONLINE] Available at: https://www.helpscout.com/blog/customer-feedback/.

[Accessed 28 February 2022].

Help Scout. 2022. Customer Feedback: Why It's Important + 7 Ways to Collect It.

[ONLINE] Available at: https://www.helpscout.com/blog/customer-feedback/.

[Accessed 28 February 2022].

Helpie WP. 2022. Best Examples of User Documentation - Helpie WP.

[ONLINE] Available at: https://helpiewp.com/user-documentation/.

[Accessed 28 February 2022].

AltexSoft. 2022. Technical Documentation in Software Development: Types and Tools | AltexSoft.

[ONLINE] Available at: https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/.

[Accessed 28 February 2022].

AltexSoft. 2022. User Manuals and Other Documentation: Types, Tools, and Best Practices | AltexSoft.

[ONLINE] Available at: https://www.altexsoft.com/blog/user-manuals-documentation/.

[Accessed 28 February 2022].

Planio. 2022. 5 Steps to Create Technical Documentation That's (Actually) Helpful | Planio.

[ONLINE] Available at: https://plan.io/blog/technical-documentation/.

[Accessed 28 February 2022].

AltexSoft Inc. 2022. Software Documentation Types and Best Practices | by AltexSoft Inc | Prototypr.

[ONLINE] Available at: https://blog.prototypr.io/software-documentation-types-and-best-practices-1726ca595c7f.

[Accessed 28 February 2022].