**ITDA310 – Project Documentation**

**Cover page**

|  |  |
| --- | --- |
| **Name** | Margo |
| **Surname** | Henning |
| **Student Number** | PT2015-0747 |
| **Submission date** | 15 - 19 May 2017 |
| **Project title** | ITDA310 Project |
| **System** | The Comic Book Store Management System |
| **Lecturer** | Kranthi Shikari |

**Table of contents**

[1. Introduction 3](#_Toc483201972)

[2. Description of the distributed database management system 4](#_Toc483201973)

[3. Application of Relational integrity rules 5](#_Toc483201974)

[4. Application of Database Security 7](#_Toc483201975)

[5. Identification of entities and entity relationship modelling 8](#_Toc483201976)

[6. Identification of relationship types for entities 9](#_Toc483201977)

[7. Determination of attribute domains 10](#_Toc483201978)

[8. Determination of candidate, primary and alternative key attributes 11](#_Toc483201979)

[9. Timestamps 13](#_Toc483201980)

[10. Views, Triggers, Procedures 14](#_Toc483201981)

[11. Developing the database 29](#_Toc483201982)

[12. Developing Application and integrating with database 33](#_Toc483201983)

[13. Connecting to the database 35](#_Toc483201984)

[14. User Manual 37](#_Toc483201985)

[15. Conclusion 49](#_Toc483201986)

[16. References 50](#_Toc483201987)

# Introduction

In this project, a system was created which includes a database as well as a C# application.

This document includes database and application documentation as well as a user manual for the system.

# Description of the distributed database management system

A distributed database management system (DDBMS) refers to a centralized application that manages a distributed database as if it was stored on the same machine. The DDBMS is responsible to synchronize all the data on the database periodically, when multiple users should be allowed to access the same data it is necessary that actions performed on the data at one location that it should be automatically reflected in data stored at another location. (Rouse,2005)

The DBMS used in this project is known as MySQL server and more specifically, the MySQL Workbench was used to interact with the database through a graphic user interface.

MySQL DBMS are known as open source software which allows for integration with multiple development languages like C, C++, C#, Java, PHP and many more. (MySQL DBMS, 2017)

A DDBMS assures transparent management of distributed, fragmented and replicated data as well as improvement of reliability and availability through distributed transactions. (Distributed Database Management System, 2017)

# Application of Relational integrity rules

There are two main relational integrity rules known as entity integrity and referential integrity.

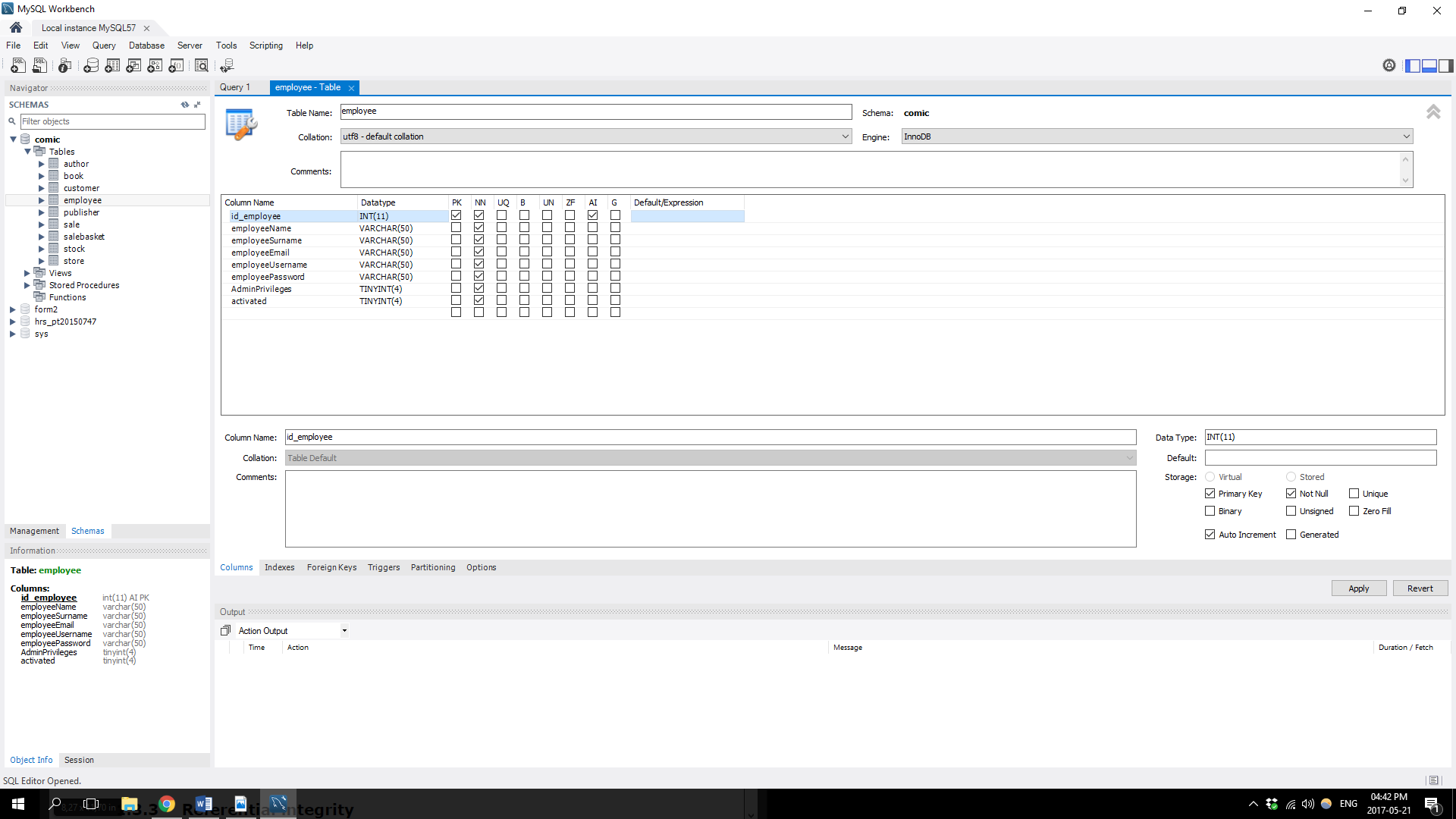
The entity integrity rule refers to a rule that is applied to primary keys of base relations.

This rule states that in a base relation it is not allowed for an attribute of a primary key to be null. A primary key serve the purpose of a minimal identifier to identify records of tables uniquely. If a null value is allowed for any part of a primary key we are saying that not all of the attributes that makes the primary key are necessary to distinguish between records of a table which goes against the purpose of a primary key.

Applying this rule in a MySQL system involves checking the attribute as ‘NOT NULL’ automatically once you indicate that a specific attribute is a primary key.

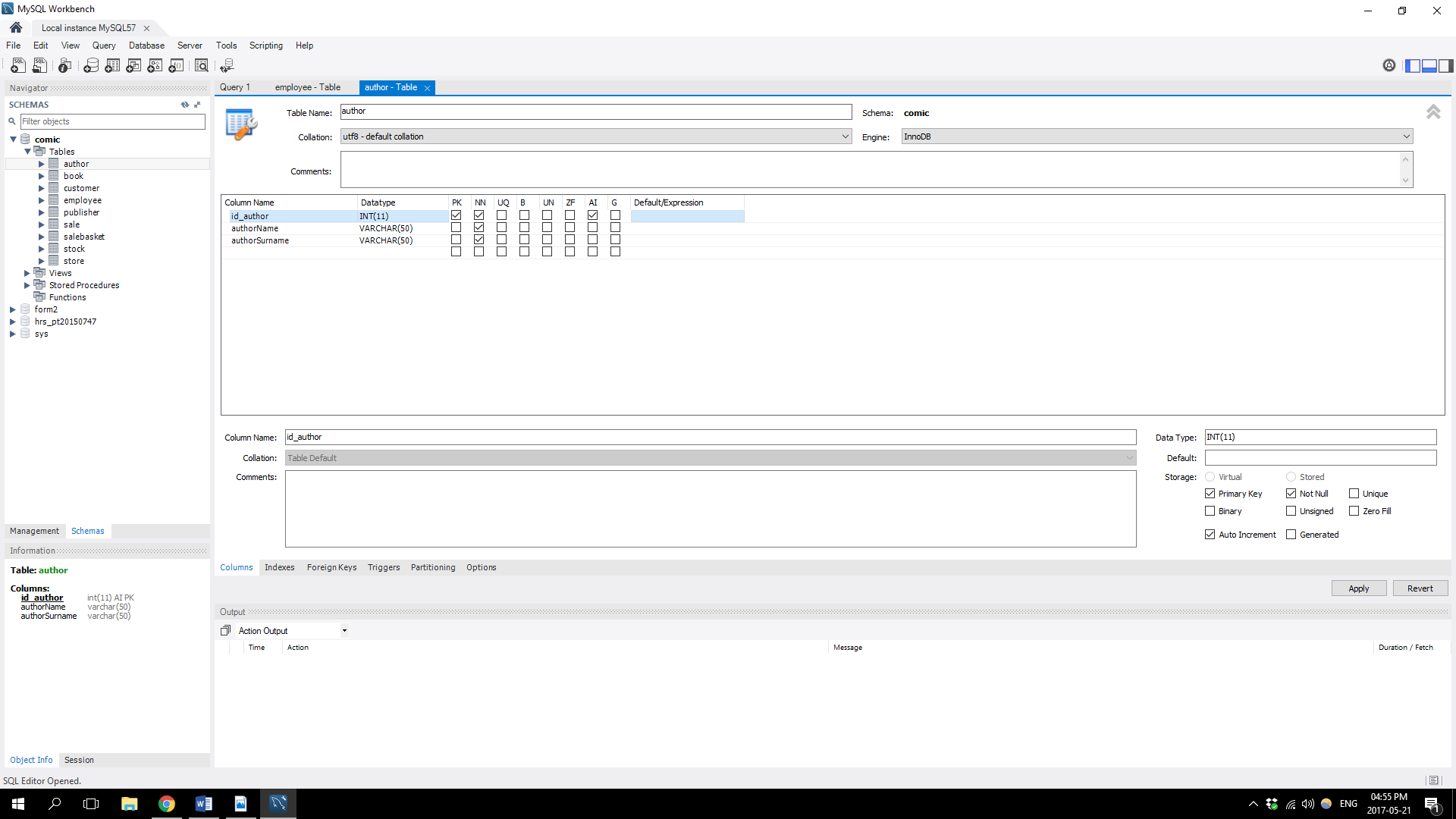
The entity integrity rule was applied to all tables in the project’s database.

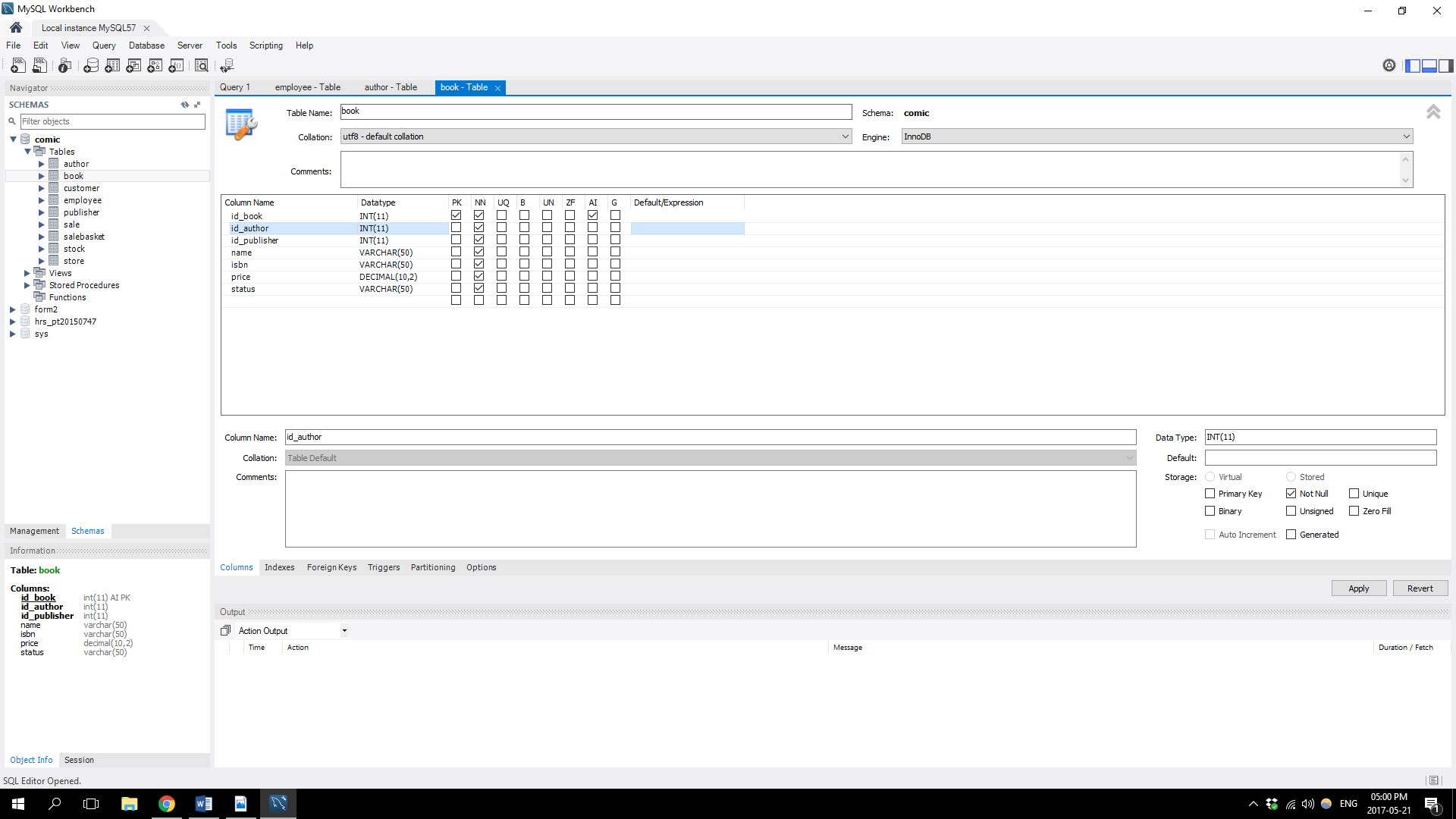
The following screenshot demonstrates how the entity integrity rule is applied in the Employee Table:



The referential integrity rule refers to a rule which is applied to foreign keys. When a foreign key exists in a relation, the foreign key value should match a candidate key value of some table in its original relation, or the foreign key value should be completely null.

The following screenshots demonstrates how the referential integrity rule is applied.

The first screenshot shows the Author table where id\_author is the primary key.

The second screenshot shows how id\_author the primary key of the Author table is used as a foreign key in the Book table:

# Application of Database Security

* Database Recovery

Database recovery is necessary for the restoring of the database to its correct state in the case of failure.

* Concurrency Control p47
* Data Integrity
* User Rights and Privileges

# Identification of entities and entity relationship modelling

The following is a list of all entities that was identified for this project’s database:

- author

- book

- customer

- employee

- publisher

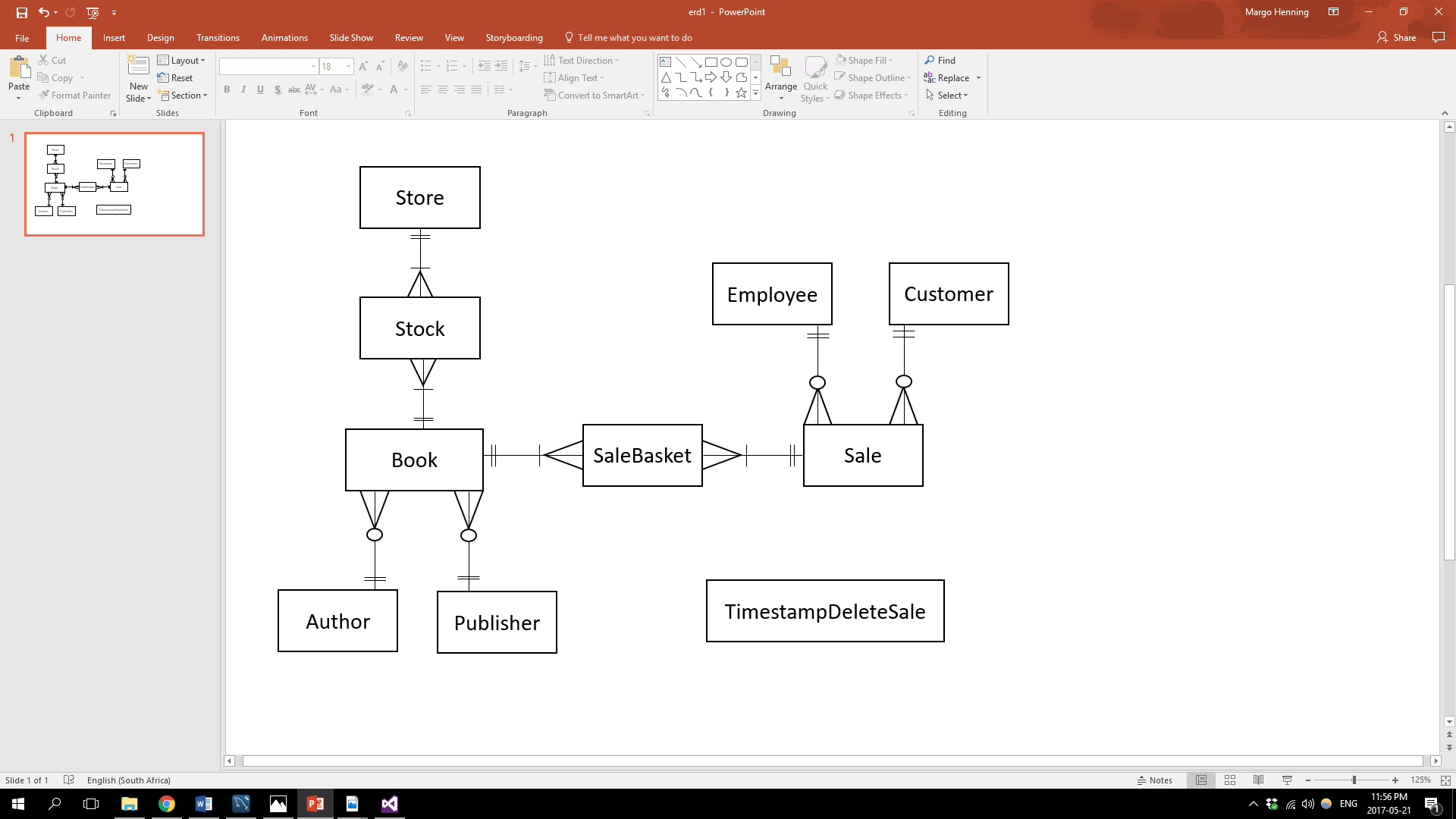
- sale

- salebasket

- stock

- store

- timestampdeletesale



# Identification of relationship types for entities

|  |  |  |
| --- | --- | --- |
| **Entities** | **Relationship type** | **Description** |
| Store to Stock | Store entity have one to many relationship with Stock | For each row in Store there can be many related rows in Stock. A stock record cannot exist without being related to a specific Store record. |
| Book to Stock | Book entity have one to many relationship with Stock | For each row in Book there can be many related rows in Stock. A stock record cannot exist without being related to a specific Book record. |
| Author to Book | Author entity have one to many relationship with Book | For each row in Author there can be many related rows in Book. A Book record cannot exist without being related to a specific Author record.  An Author record can exist without being related to any Book records. |
| Publisher to Book | Publisher entity have one to many relationship with Book | For each row in Publisher there can be many related rows in Book. A Book record cannot exist without being related to a specific Publisher record. A Publisher record can exist without being related to any Book records. |
| Book to SaleBasket | Book entity have one to many relationship with SaleBasket | For each row in Book there can be many related rows in SaleBasket. A SaleBasket record cannot exist without being related to a specific Book record. |
| Sale to SaleBasket | Sale entity have one to many relationship with SaleBasket | For each row in Sale there can be many related rows in SaleBasket. A SaleBasket record cannot exist without being related to a specific Sale record. |
| Customer to Sale | Customer entity have one to many relationship with Sale | For each row in Customer there can be many related rows in Sale. A Sale record cannot exist without being related to a specific Customer record.  A Customer record can exist without being related to any Sale records. |
| Employee to Sale | Employee entity have one to many relationship with Sale | For each row in Employee there can be many related rows in Sale. A Sale record cannot exist without being related to a specific Employee record.  An Employee record can exist without being related to any Sale records. |

# Determination of attribute domains

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Attribute Domain where applicable** |
| Store | id\_store | 1 to |
| storeName |  |
| storeContactEmail |  |
| storeContactNumber |  |
| Stock | id\_stock | 1 to |
| Quantity | 1 to |
| Book | id\_book | 1 to |
| Name |  |
| Isbn |  |
| Price |  |
| Status | ‘Sold’ or ‘Available’ |
| Author | id\_author | 1 to |
| authorName |  |
| AuthorSurname |  |
| Publisher | id\_publisher | 1 to |
| publisherName |  |
| SaleBasket | id\_salebasket | 1 to |
| quantity | 1 to |
| Sale | Id\_sale | 1 to |
| date |  |
| totalPrice |  |
| Customer | id\_customer | 1 to |
| customerName |  |
| customerSurname |  |
| customerEmail |  |
| Employee | id\_employee | 1 to |
| employeeName |  |
| employeeSurname |  |
| employeeEmail |  |
| employeeUsername |  |
| employeePassword |  |
| AdminPrivileges | ‘True’ or ‘False’ |
| activated | ‘True’ or ‘False’ |
| Timestampdeletesale | timestampID | 1 to |
| Timestamp |  |
| transactionName |  |
| customerID | 1 to |
| employeeID | 1 to |

# Determination of candidate, primary and alternative key attributes

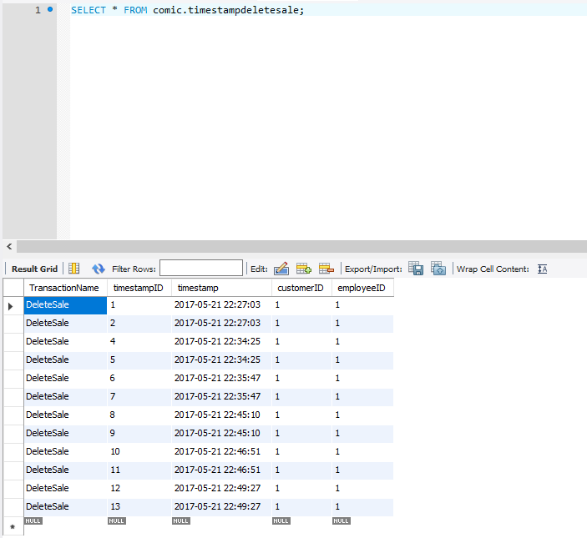
|  |  |  |
| --- | --- | --- |
| **Entity** | **Attributes** | **Keys** |
| Store | id\_store | Candidate Key then chosen as Primary Key |
| storeName |  |
| storeContactEmail |  |
| storeContactNumber |  |
| Stock | id\_stock | Candidate Key then chosen as Primary Key |
| Quantity |  |
| Id\_book | Foreign Key |
| Id\_store | Foreign Key |
| Book | id\_book | Candidate Key then chosen as Primary Key |
| Name |  |
| Isbn | Candidate Key then became alternative key |
| Price |  |
| Status |  |
| Id\_author | Foreign Key |
| Id\_publisher | Foreign Key |
| Author | id\_author | Candidate Key then chosen as Primary Key |
| authorName |  |
| AuthorSurname |  |
| Publisher | id\_publisher | Candidate Key then chosen as Primary Key |
| publisherName |  |
| SaleBasket | id\_salebasket | Candidate Key then chosen as Primary Key |
| quantity |  |
| Id\_sale | Foreign Key |
| Id\_book | Foreign Key |
| Sale | Id\_sale | Candidate Key then chosen as Primary Key |
| date |  |
| totalPrice |  |
| Id\_customer | Foreign Key |
| Id\_employee | Foreign Key |
| Customer | id\_customer | Candidate Key then chosen as Primary Key |
| customerName |  |
| customerSurname |  |
| customerEmail | Candidate Key then Alternative Key |
| Employee | id\_employee | Candidate Key then chosen as Primary Key |
| employeeName |  |
| employeeSurname |  |
| employeeEmail |  |
| employeeUsername | Candidate Key then Alternative Key |
| employeePassword |  |
| AdminPrivileges |  |
| activated |  |
| Timestampdeletesale | timestampID | Candidate Key then chosen as Primary Key |
| Timestamp |  |
| transactionName |  |
| customerID |  |
| employeeID |  |

# Timestamps

Timestamps are used to record events that occurred at a specific time in a computer system.

In this project, a timestampdeletesale table was created as a log table to record the time and date of when a sale record was deleted as well as other useful information regarding the sale record that was deleted.

The following screenshot show the timestampdeletesale table:

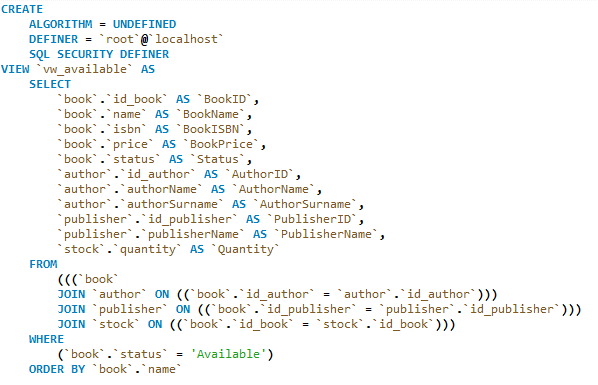


# Views, Triggers, Procedures

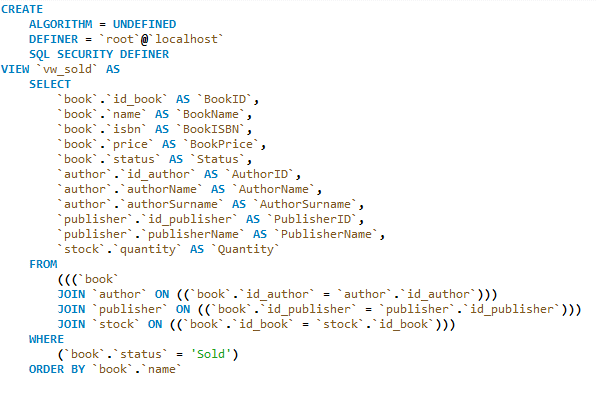
Views:

vw\_available: This view selects information of all the comic books where the status is equal to ‘Available’.

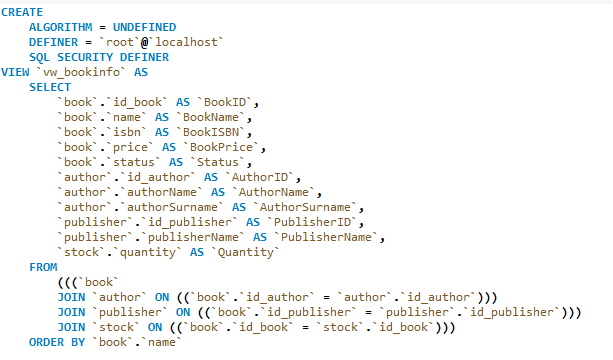
The following shows the query to create the view:



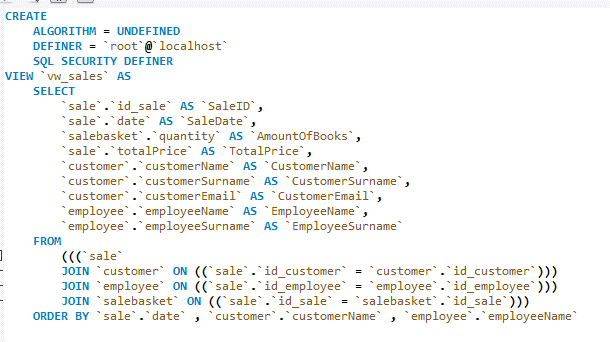
vw\_sold: This view selects information of all the comic books where the status is equal to ‘Sold’.

The following shows the query to create the view:

vw\_bookinfo: This view selects attributes from multiple tables that are related to the Book table.

 The following shows the query to create the view:

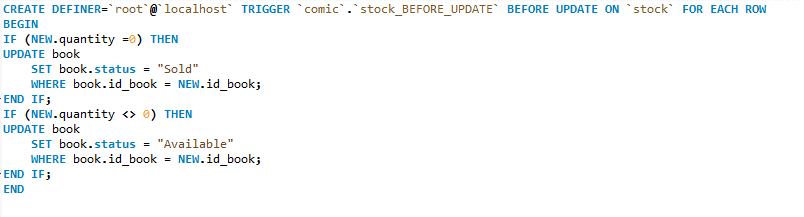
vw\_sales: This view selects attributes from multiple tables that are related to the Sale table.

The following shows the query to create the view:

Triggers:

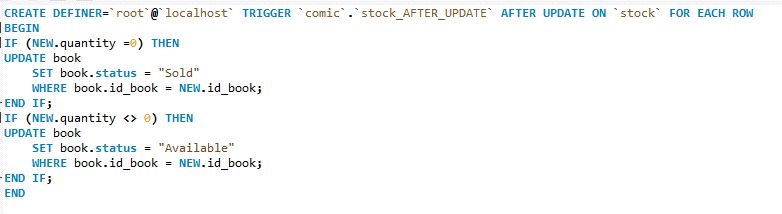
stock\_BEFORE\_UPDATE:

This trigger ensures that when the quantity is changed to 0, the status of the book will be changed to ‘Sold’, and when the quantity is changed to more than 0, the status of the book will be changed to ‘Available’.

The following shows the query to create the trigger:

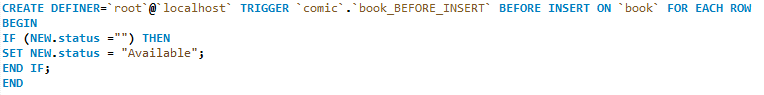
stock\_AFTER\_UPDATE:

This trigger ensures that when the quantity is changed to 0, the status of the book will be changed to ‘Sold’, and when the quantity is changed to more than 0, the status of the book will be changed to ‘Available’.

The following shows the query to create the trigger:

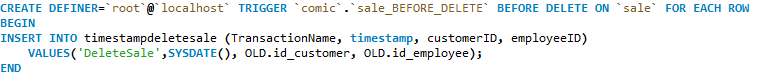
book\_BEFORE\_INSERT:

This trigger ensures that when a record is inserted into the book table and the status equals “ ” then the status will be made “Available”.

The following shows the query to create the trigger:

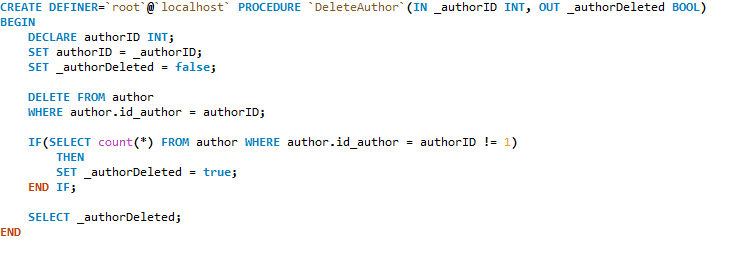
sale\_BEFORE\_DELETE:

This trigger ensures that when a record is deleted from the sale table that a record is created in the timestampdeletesale table.

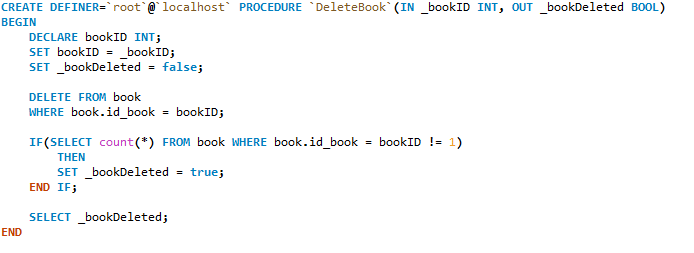
The following shows the query to create the trigger:

Stored Procedures:

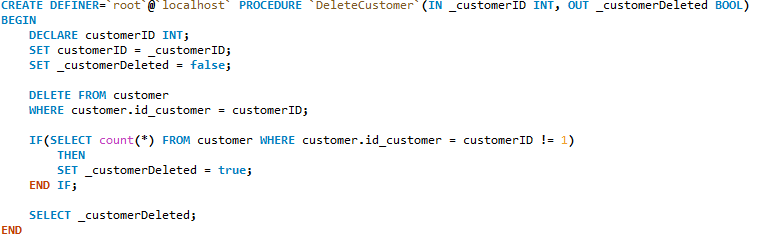
DeleteAuthor: This stored procedure deletes an author record.

The following shows the query to create the stored procedure:

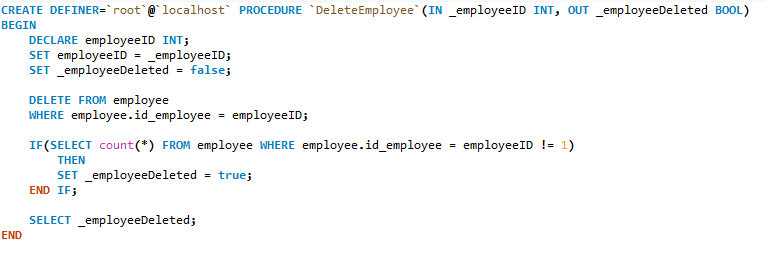
DeleteBook: This stored procedure deletes a book record.

The following shows the query to create the stored procedure:

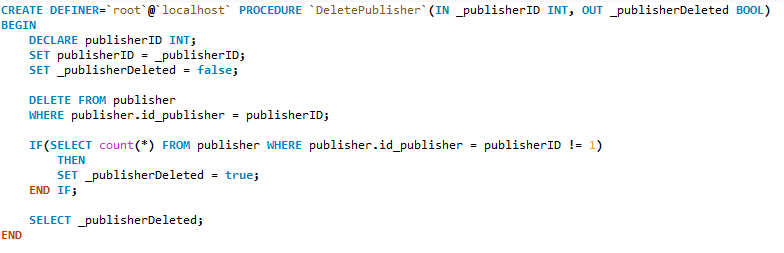
DeleteCustomer: This stored procedure deletes a customer record.

The following shows the query to create the stored procedure:

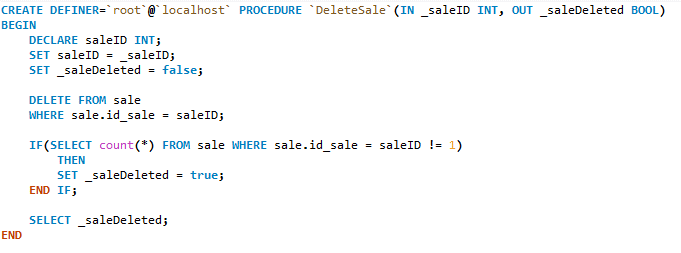
DeleteEmployee: This stored procedure deletes an employee record.

The following shows the query to create the stored procedure:

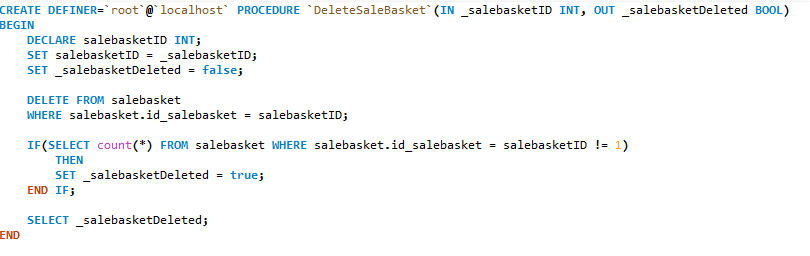
DeletePublisher: This stored procedure deletes a publisher record.

The following shows the query to create the stored procedure:

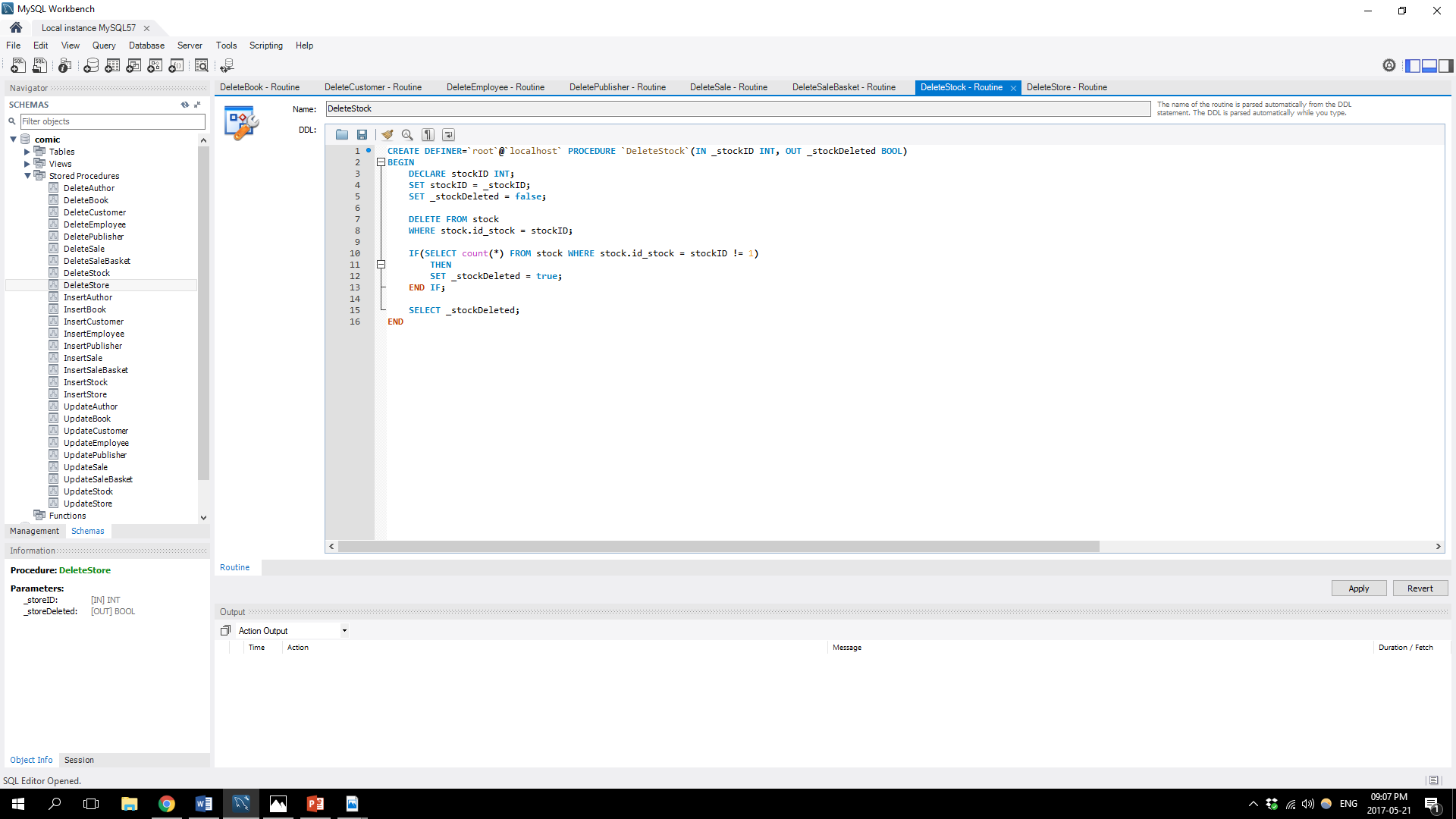
DeleteSale: This stored procedure deletes a sale record.

The following shows the query to create the stored procedure:

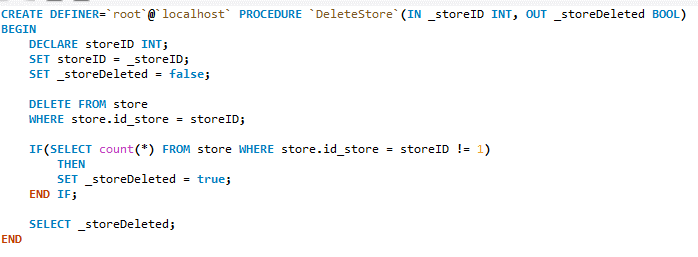
DeleteSaleBasket: This stored procedure deletes an saleBasket record.

The following shows the query to create the stored procedure:

DeleteStock: This stored procedure deletes a stock record.

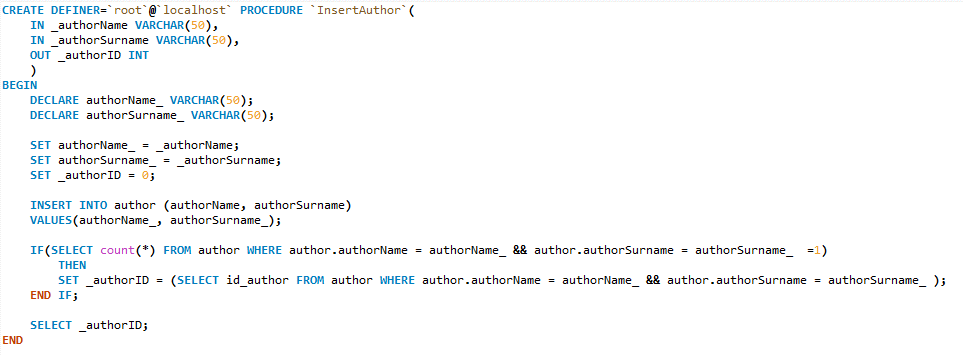
The following shows the query to create the stored procedure:

DeleteStore: This stored procedure deletes a store record.

The following shows the query to create the stored procedure:

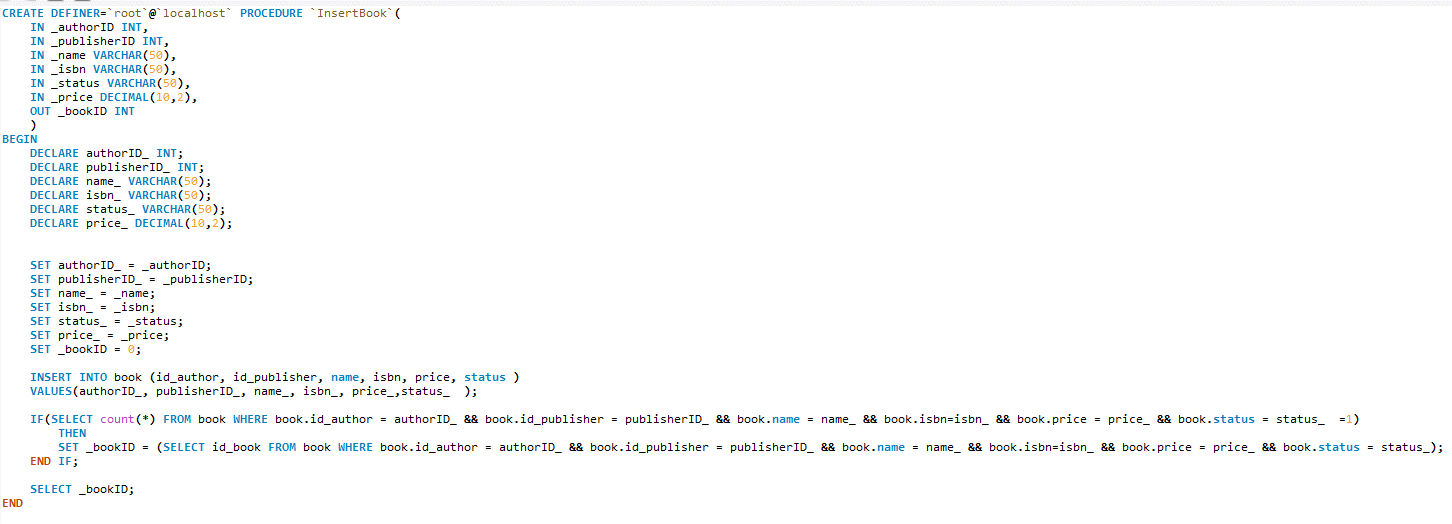
InsertAuthor:

This stored procedure inserts an author record.

The following shows the query to create the stored procedure:

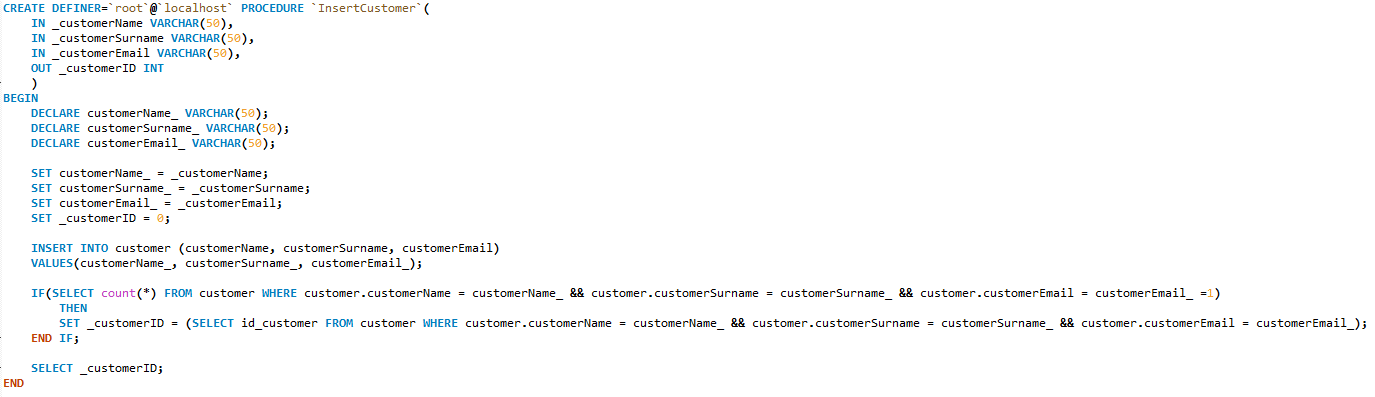
InsertBook:

This stored procedure inserts a book record.

The following shows the query to create the stored procedure:

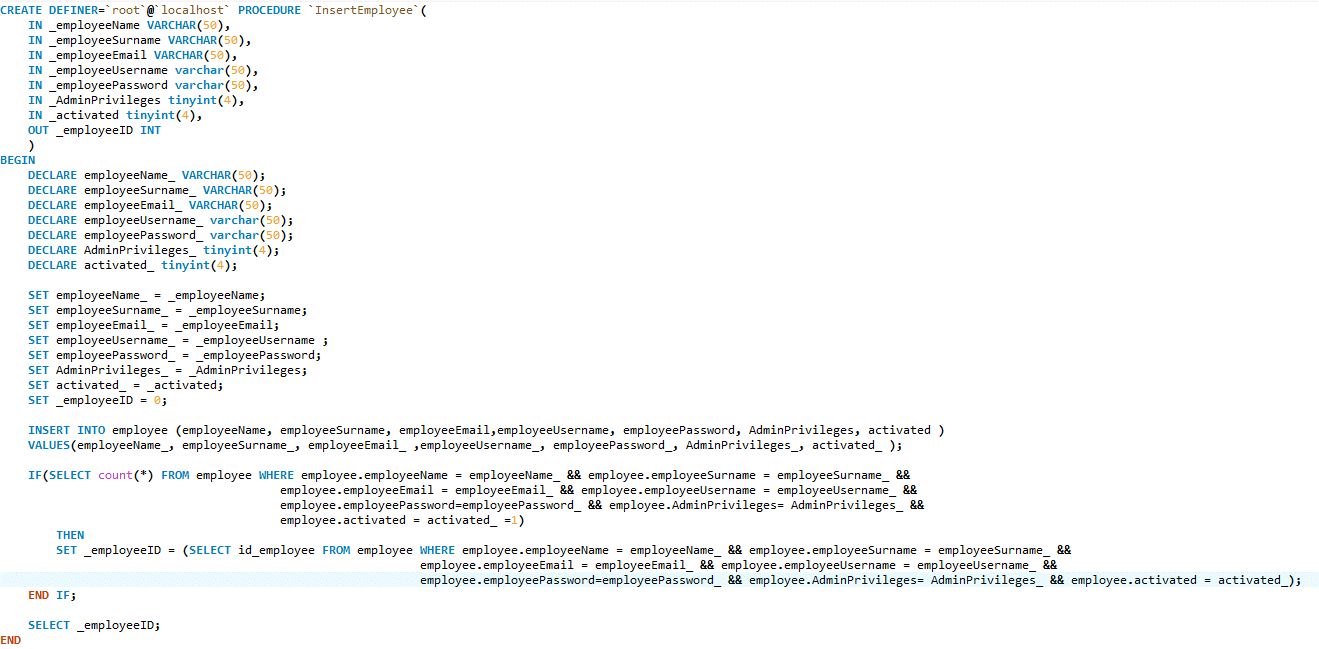
InsertCustomer:

This stored procedure inserts a customer record.

The following shows the query to create the stored procedure:

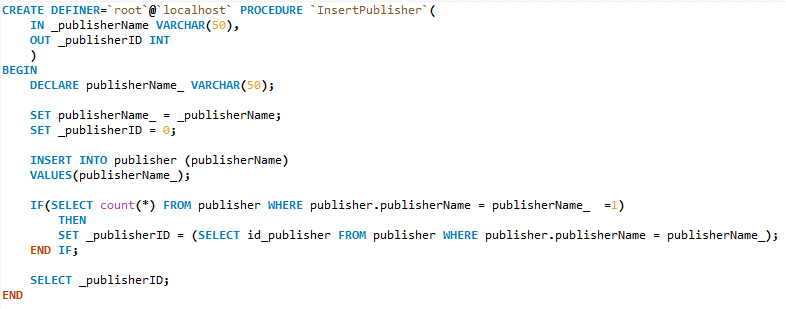
InsertEmployee:

This stored procedure inserts an employee record.

The following shows the query to create the stored procedure:

InsertPublisher:

This stored procedure inserts a publisher record.

The following shows the query to create the stored procedure:

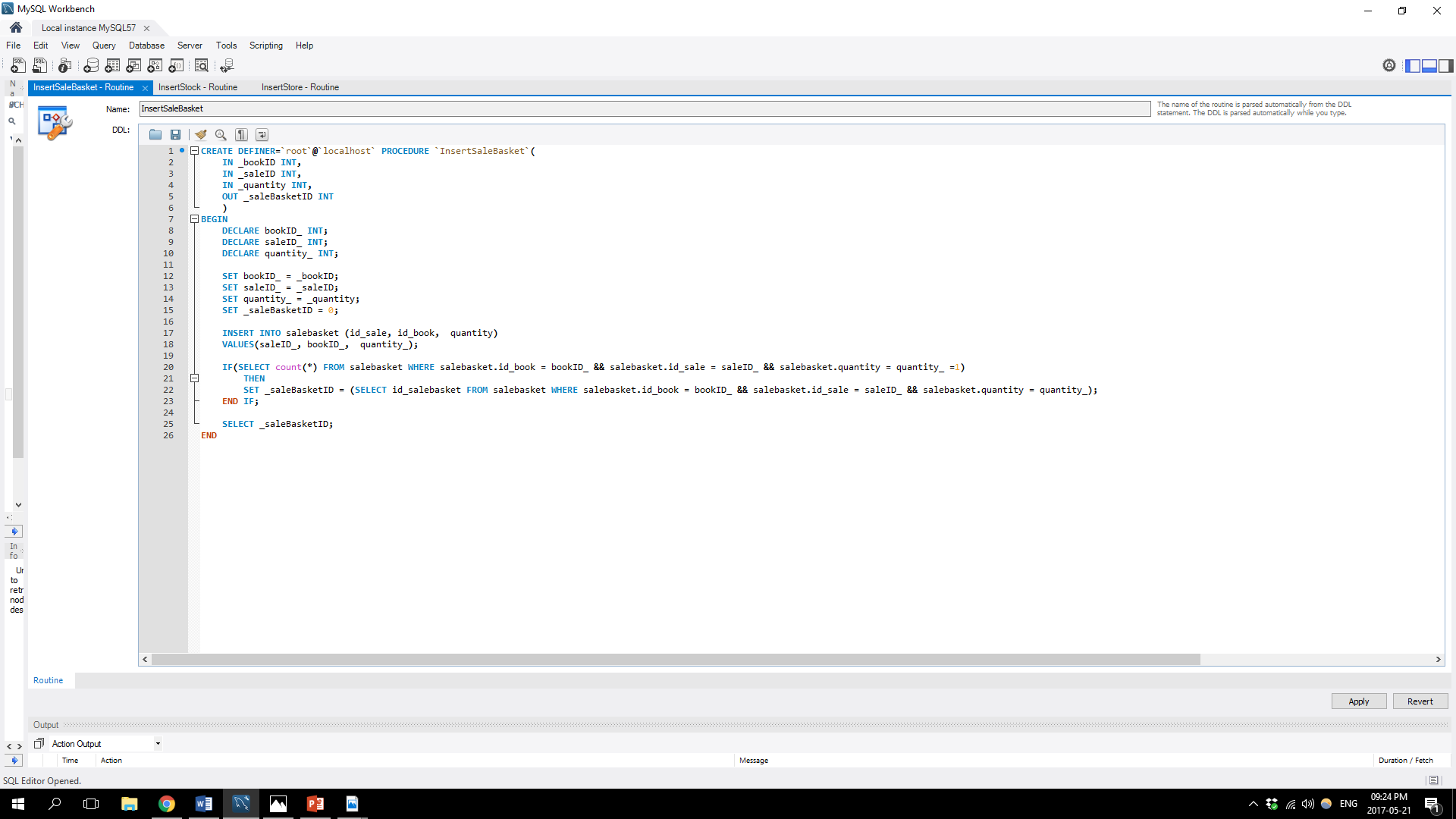
InsertSale:

This stored procedure inserts a sale record.

The following shows the query to create the stored procedure:

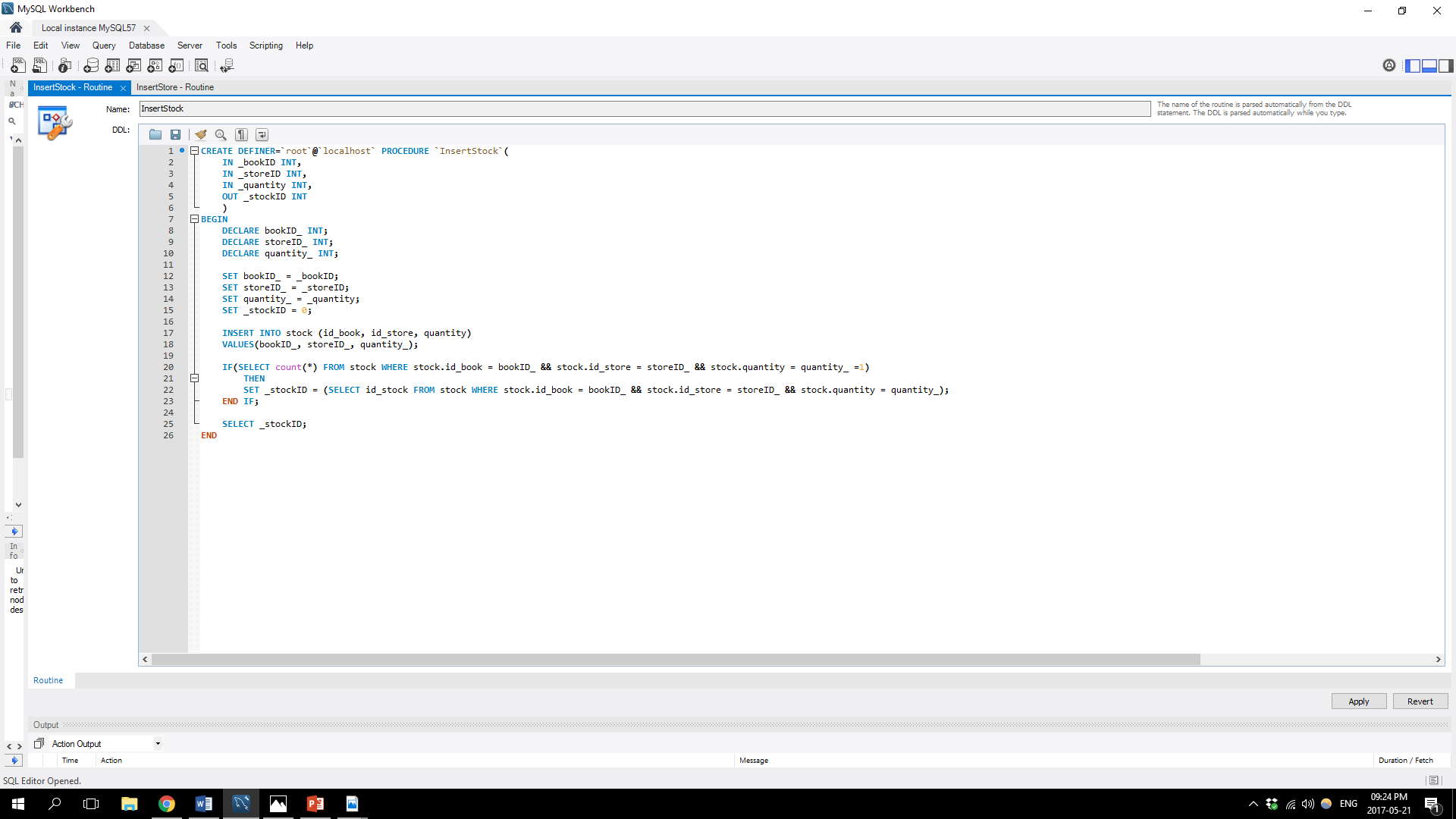
InsertSaleBasket:

This stored procedure inserts a saleBasket record.

The following shows the query to create the stored procedure:

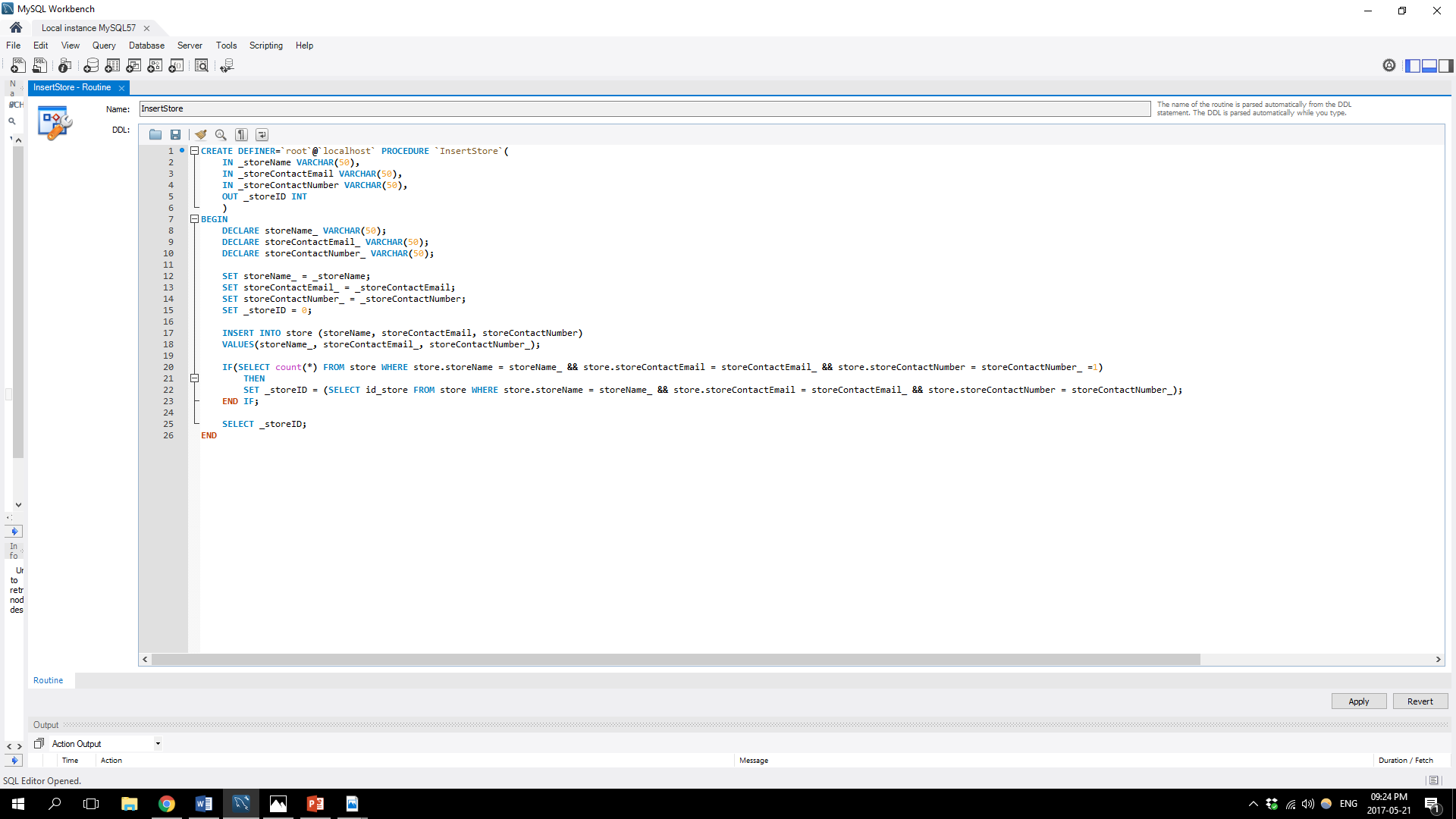
InsertStock:

This stored procedure inserts a stock record.

The following shows the query to create the stored procedure:

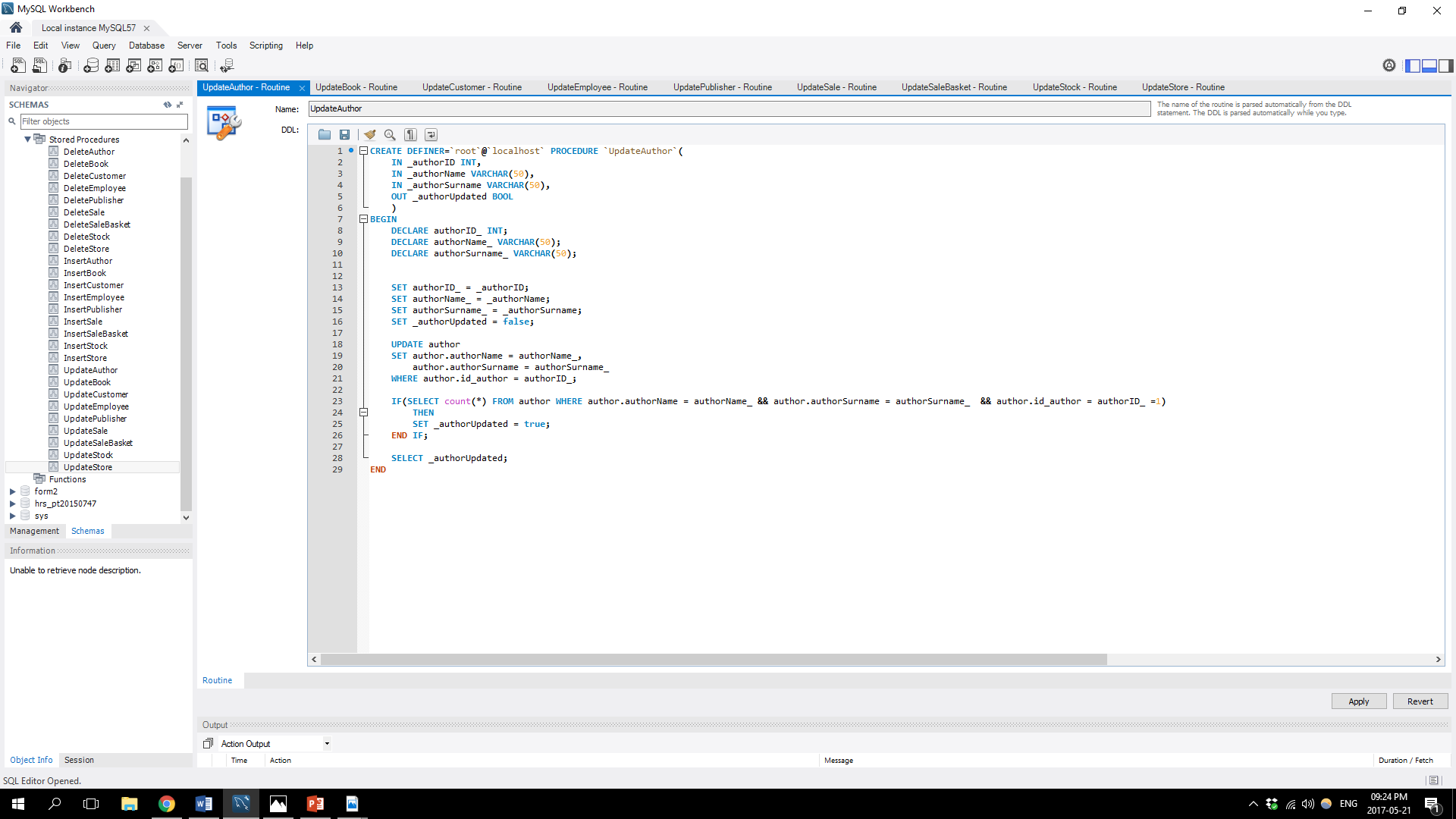
InsertStore:

This stored procedure inserts a store record.

The following shows the query to create the stored procedure:

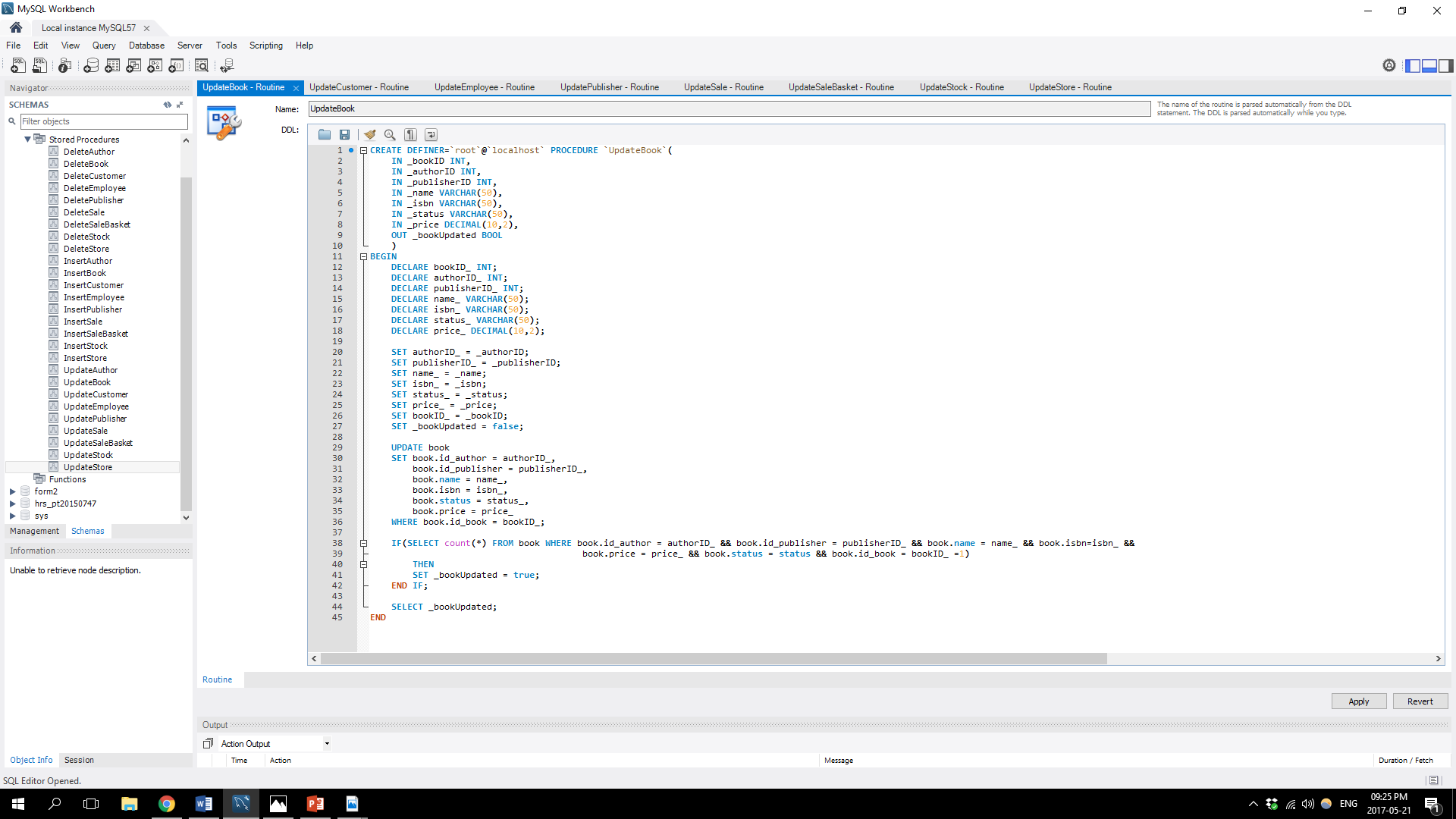
UpdateAuthor:

This stored procedure updates an author record.

The following shows the query to create the stored procedure:

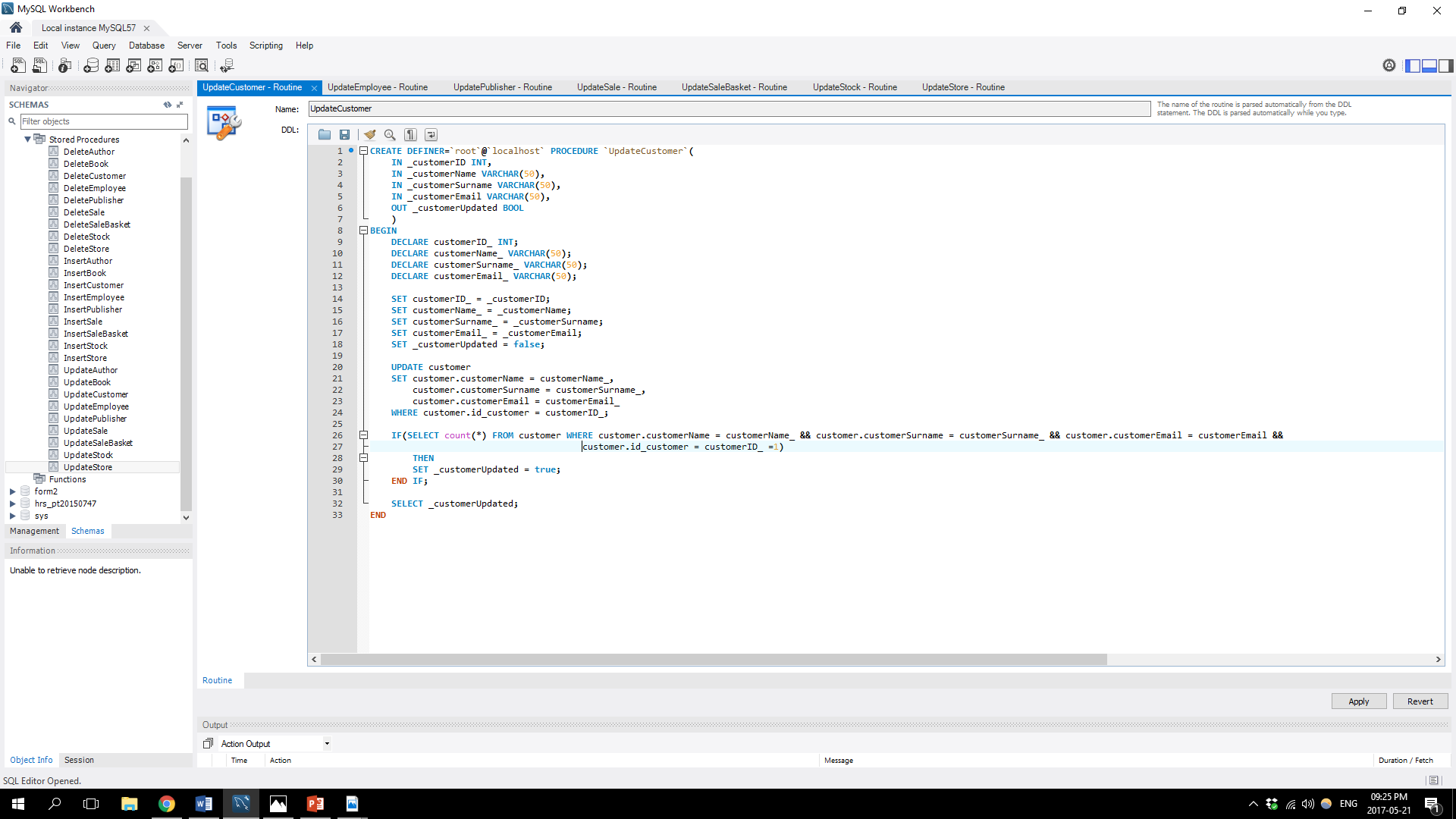
UpdateBook:

This stored procedure updates a book record.

The following shows the query to create the stored procedure:

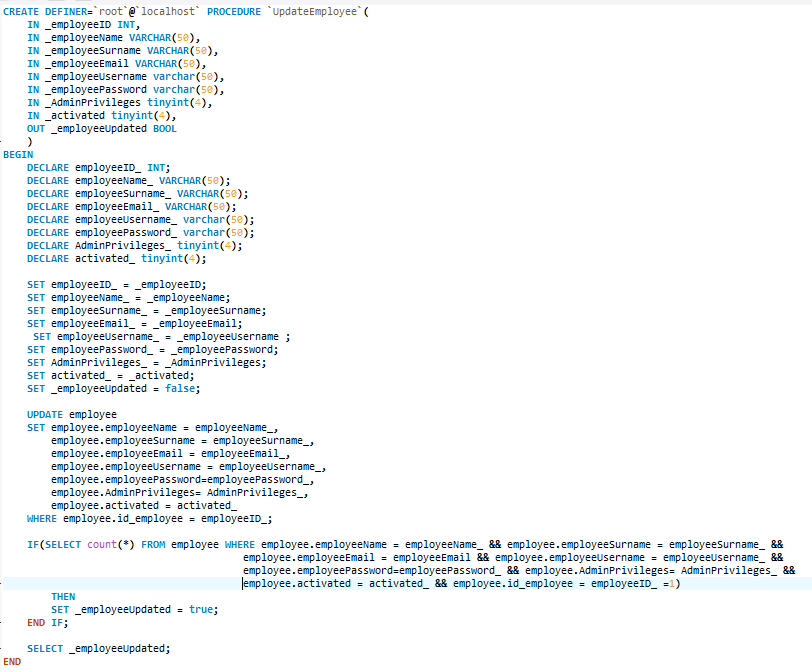
UpdateCustomer:

This stored procedure updates a customer record.

The following shows the query to create the stored procedure:

UpdateEmployee:

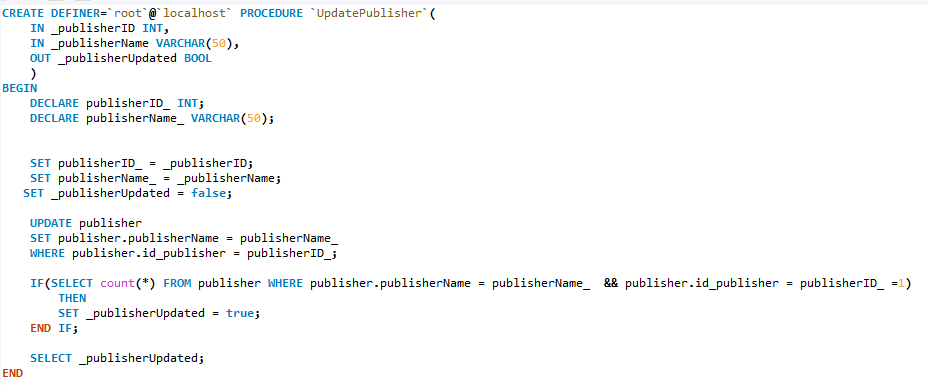
This stored procedure updates an employee record.

The following shows the query to create the stored procedure:

UpdatePublisher:

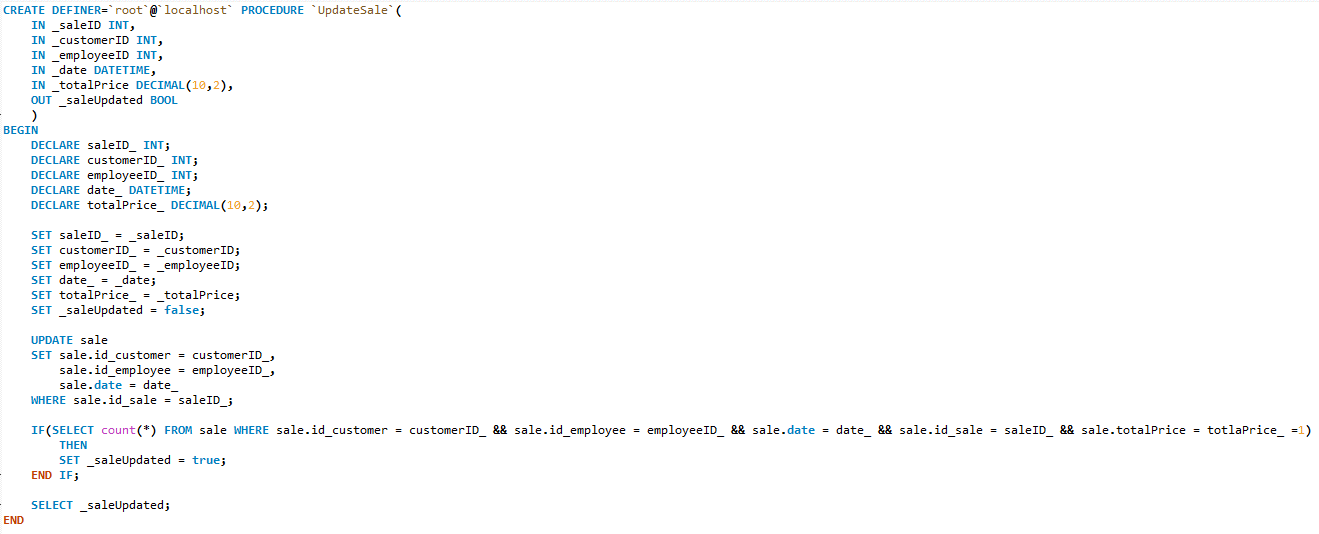
This stored procedure updates a publisher record.

The following shows the query to create the stored procedure:



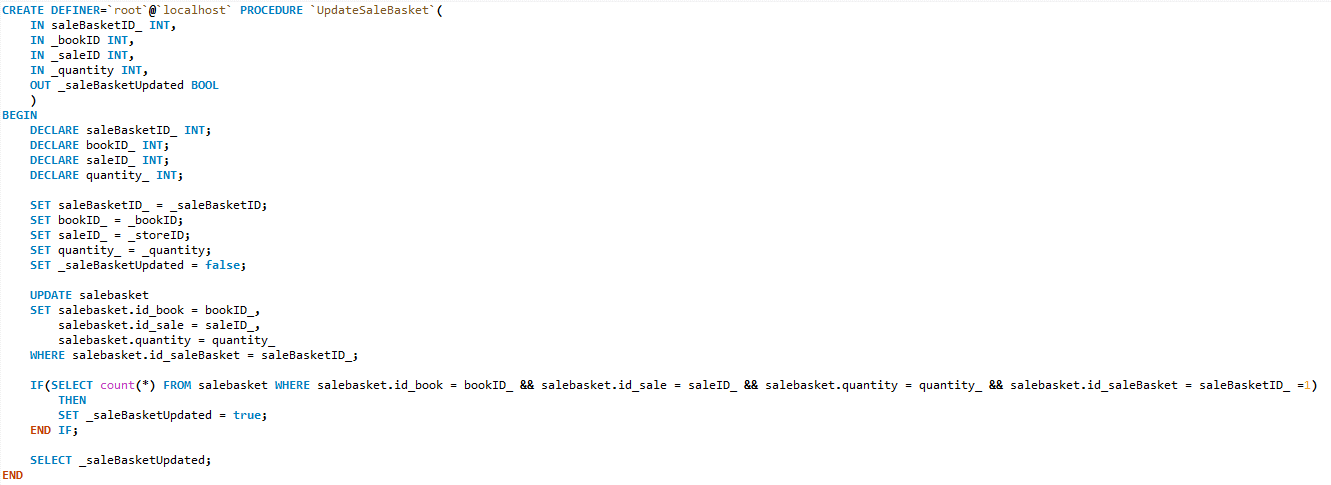
UpdateSale:

This stored procedure updates a sale record.

The following shows the query to create the stored procedure:

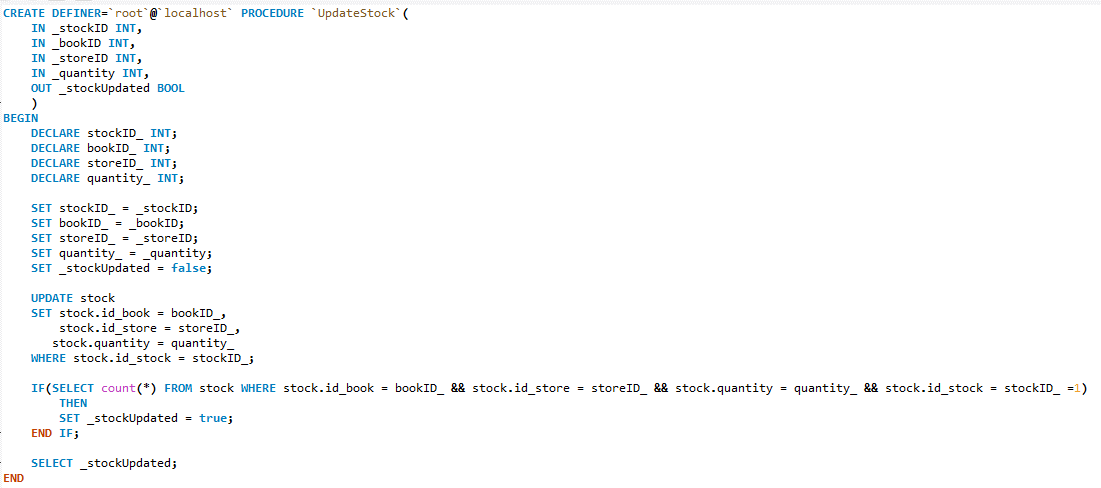
UpdateSaleBasket:

This stored procedure updates an saleBasket record.

The following shows the query to create the stored procedure:

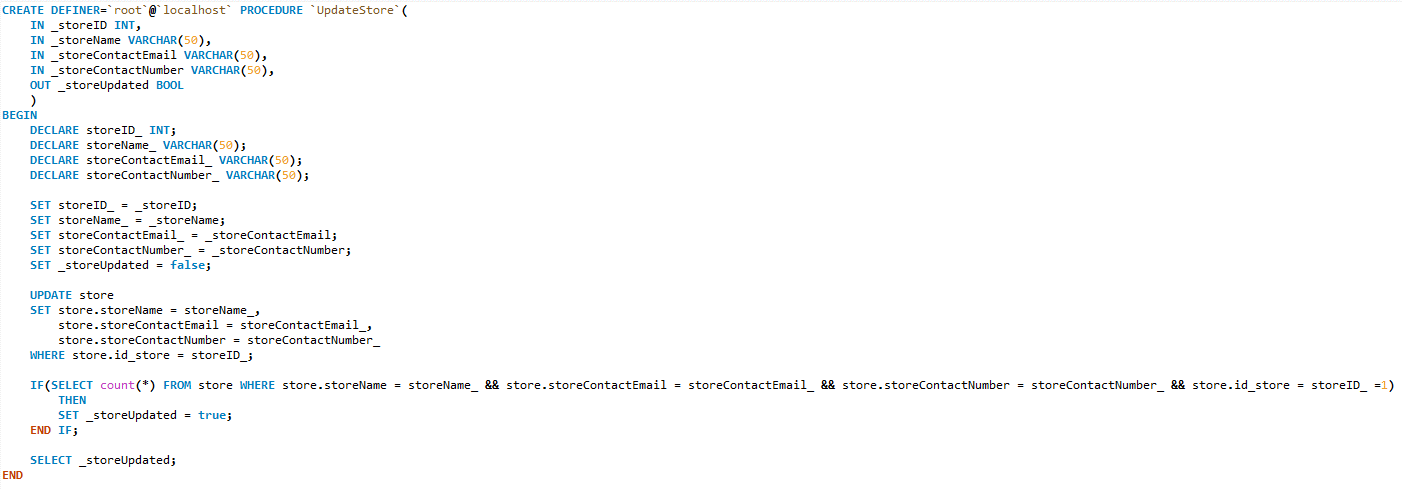
UpdateStock:

This stored procedure updates a stock record.

The following shows the query to create the stored procedure:

UpdateStore:

This stored procedure updates a store record.

The following shows the query to create the stored procedure:

# Developing the database

* Data Definition Language (DDL)

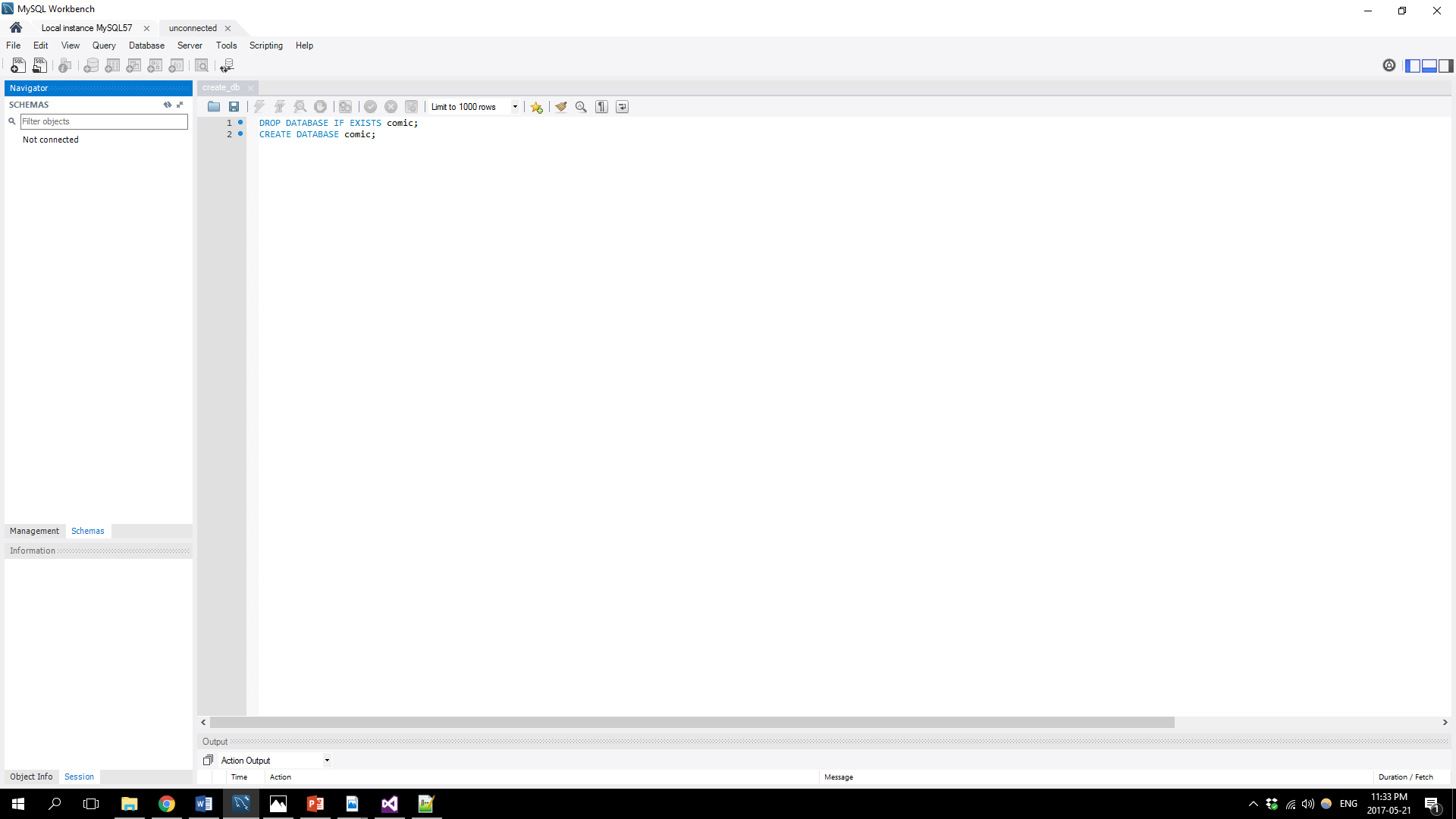
DDL statements are mostly used to build and modify the structure of database tables and other objects in a database.

When executing these DDL statements it takes effect immediately.

DDL statements include, create, alter, drop, truncate, comment and rename statements. (Jewett, 2006)

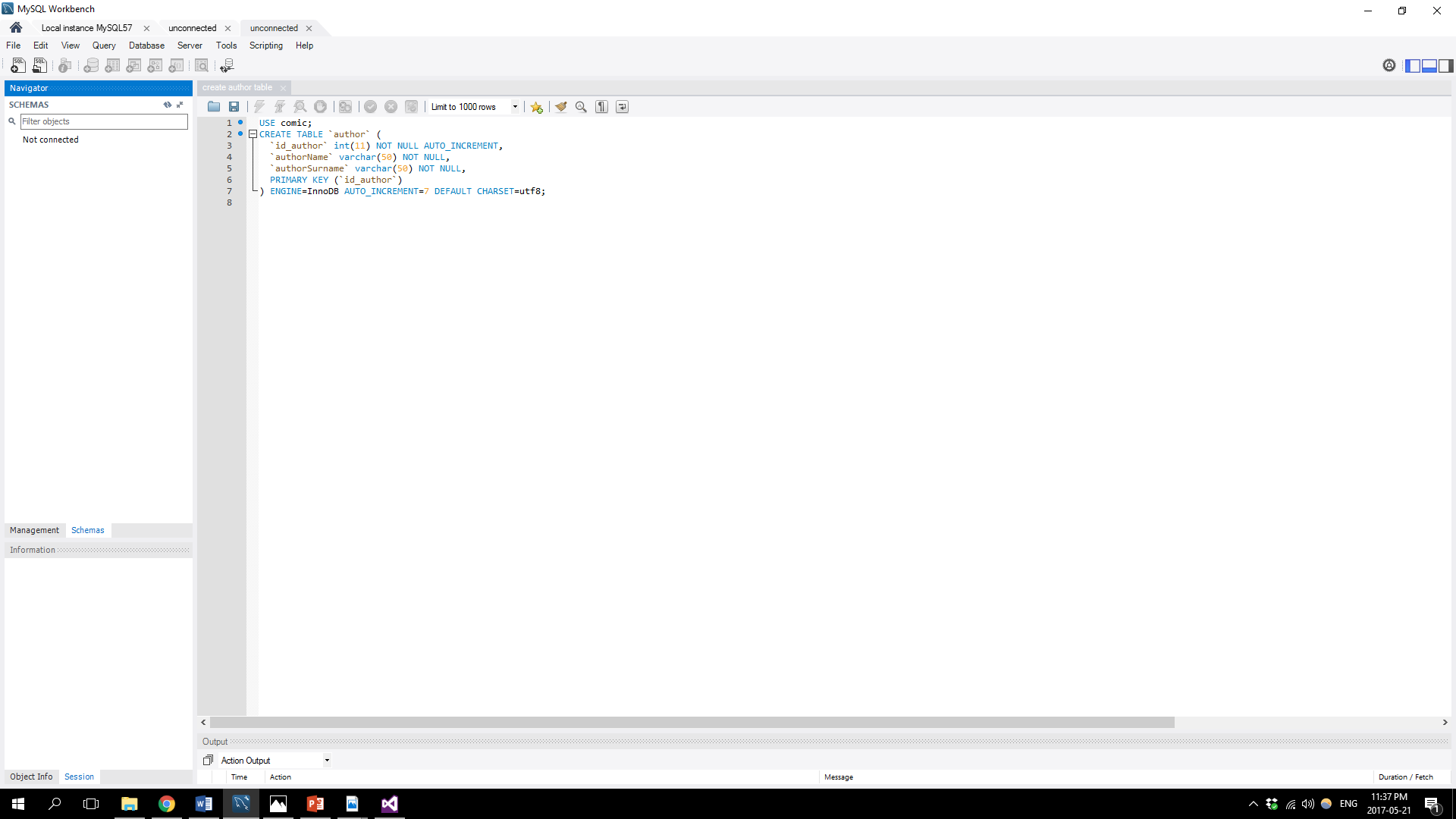
During this project DDL statements were used to create and alter tables as well as creating the database self.

The following shows the query used to create the database:

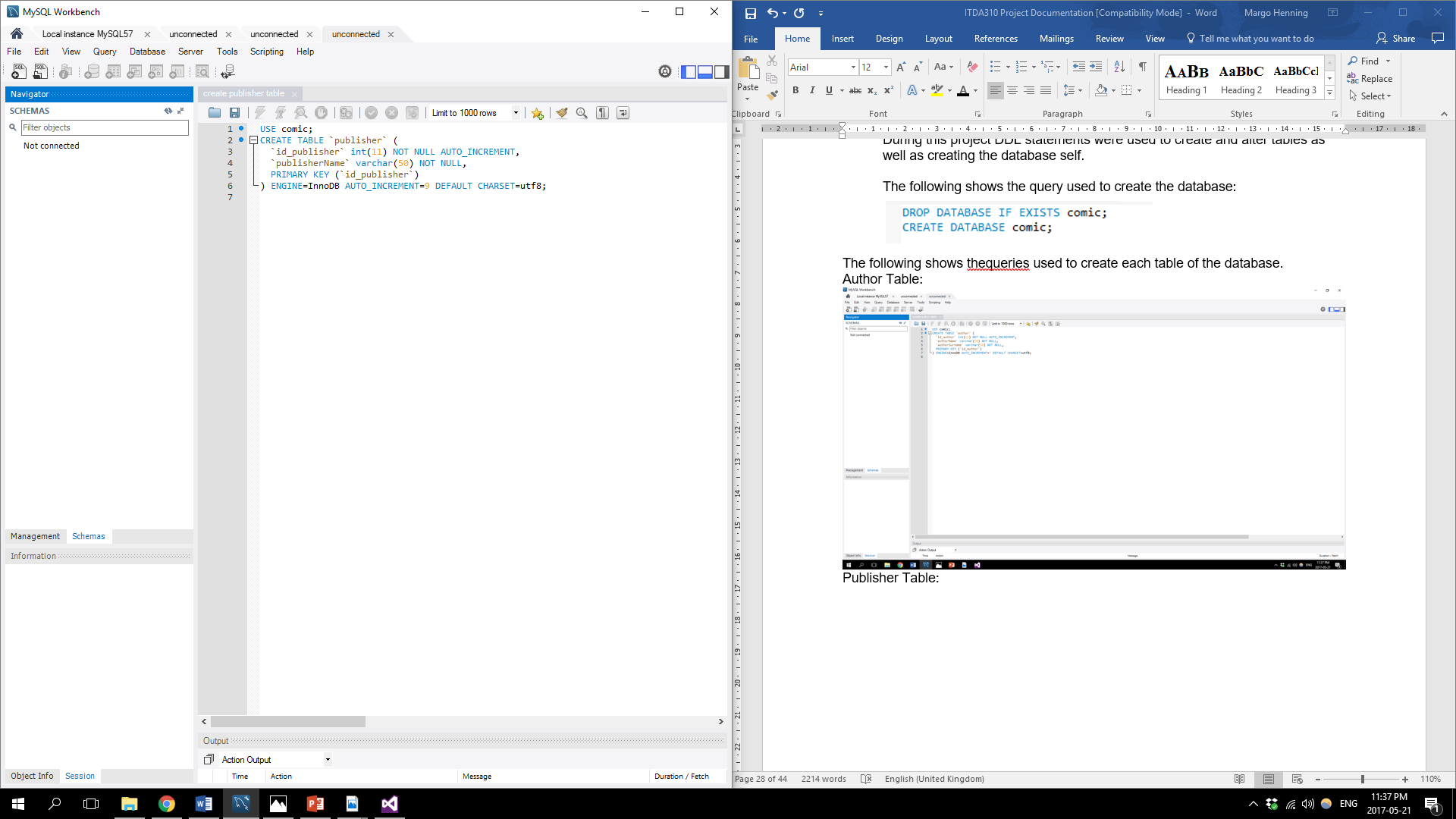


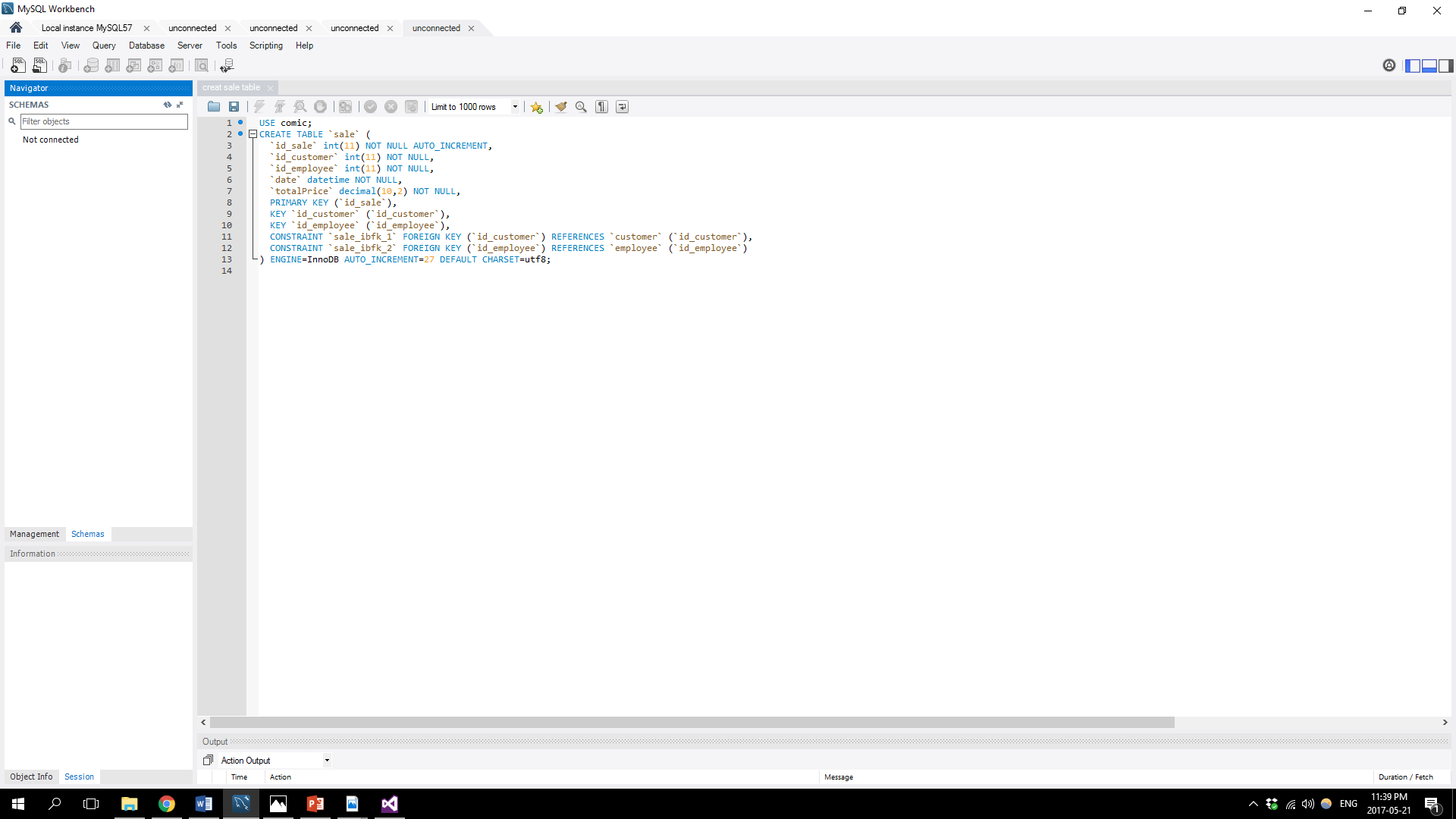
The following shows the queries used to create each table of the database.

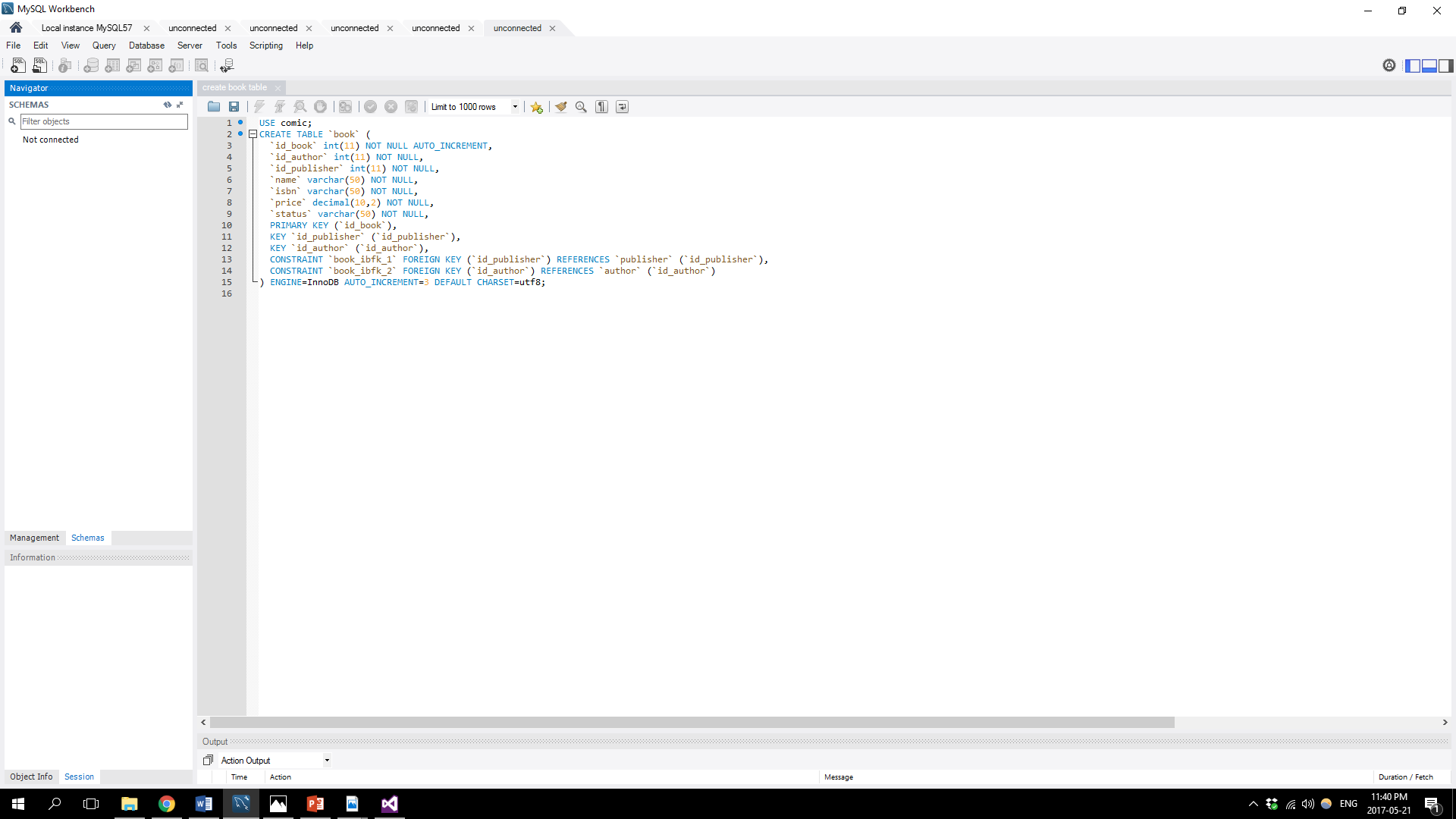
Author Table:

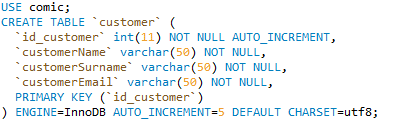


Publisher Table:

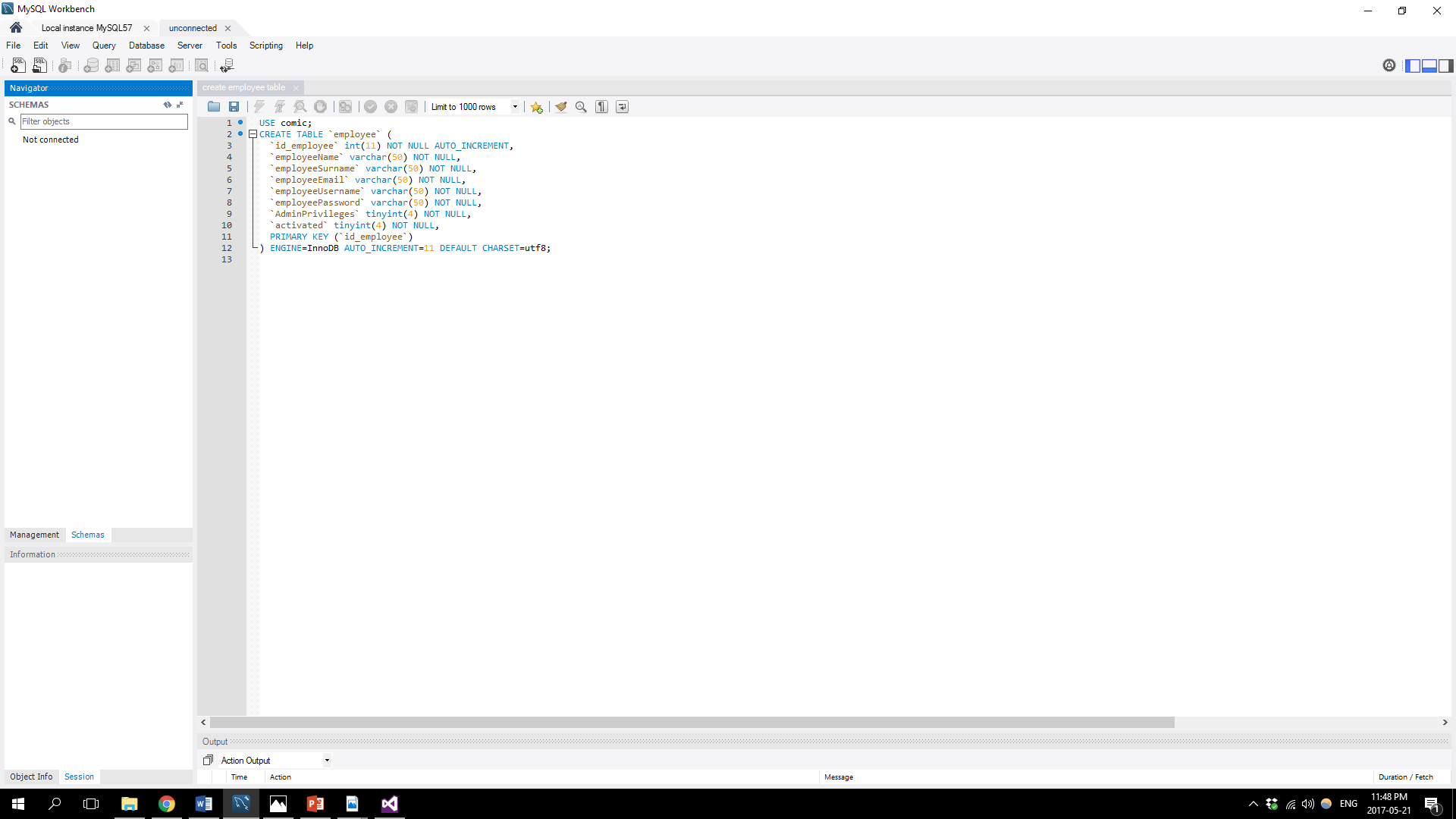


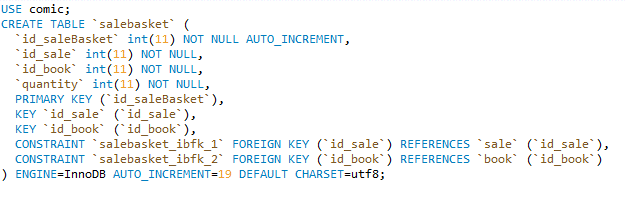
Sale Table:

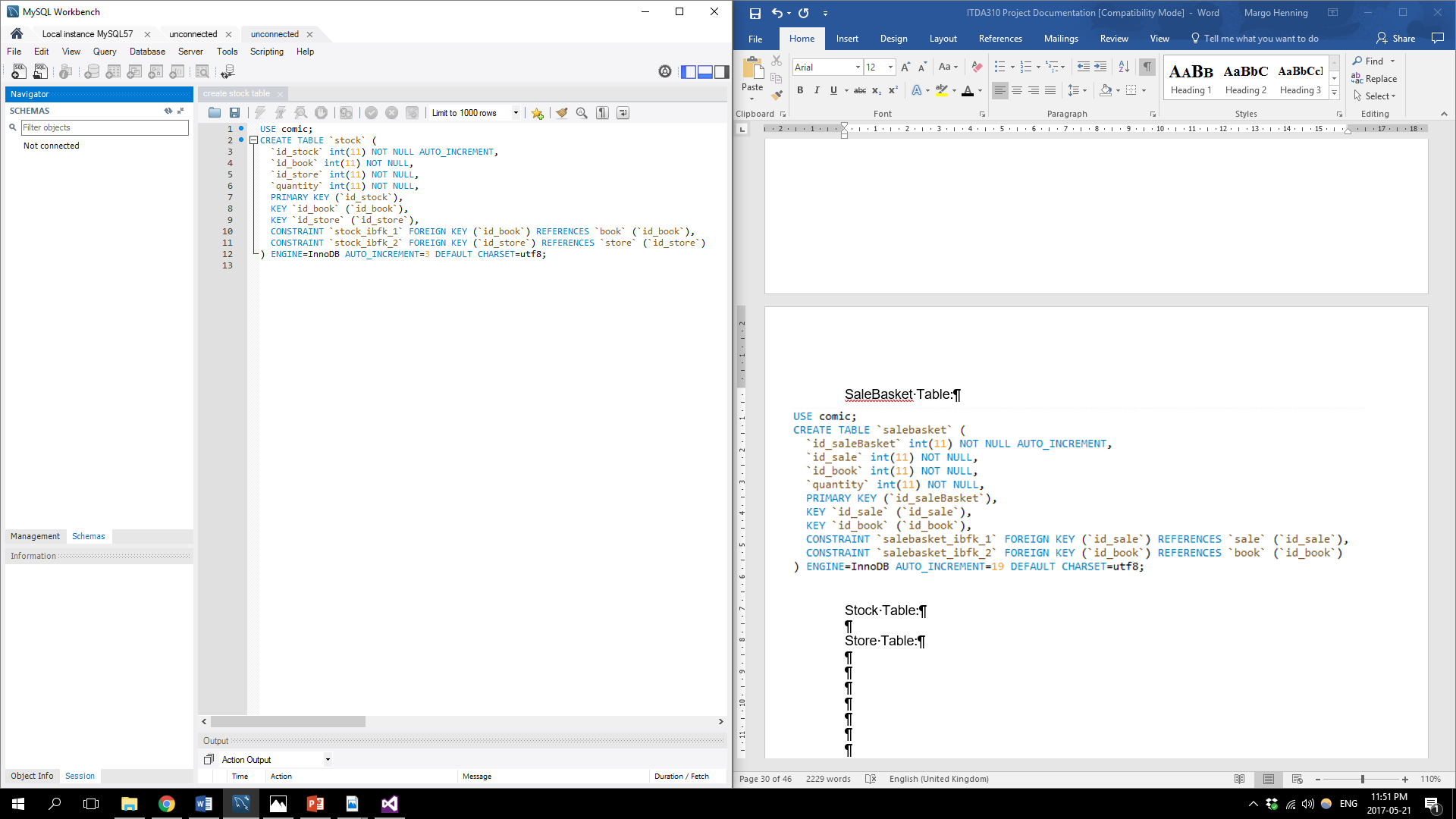
Book Table:

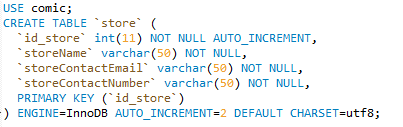
Customer Table:

Employee Table:

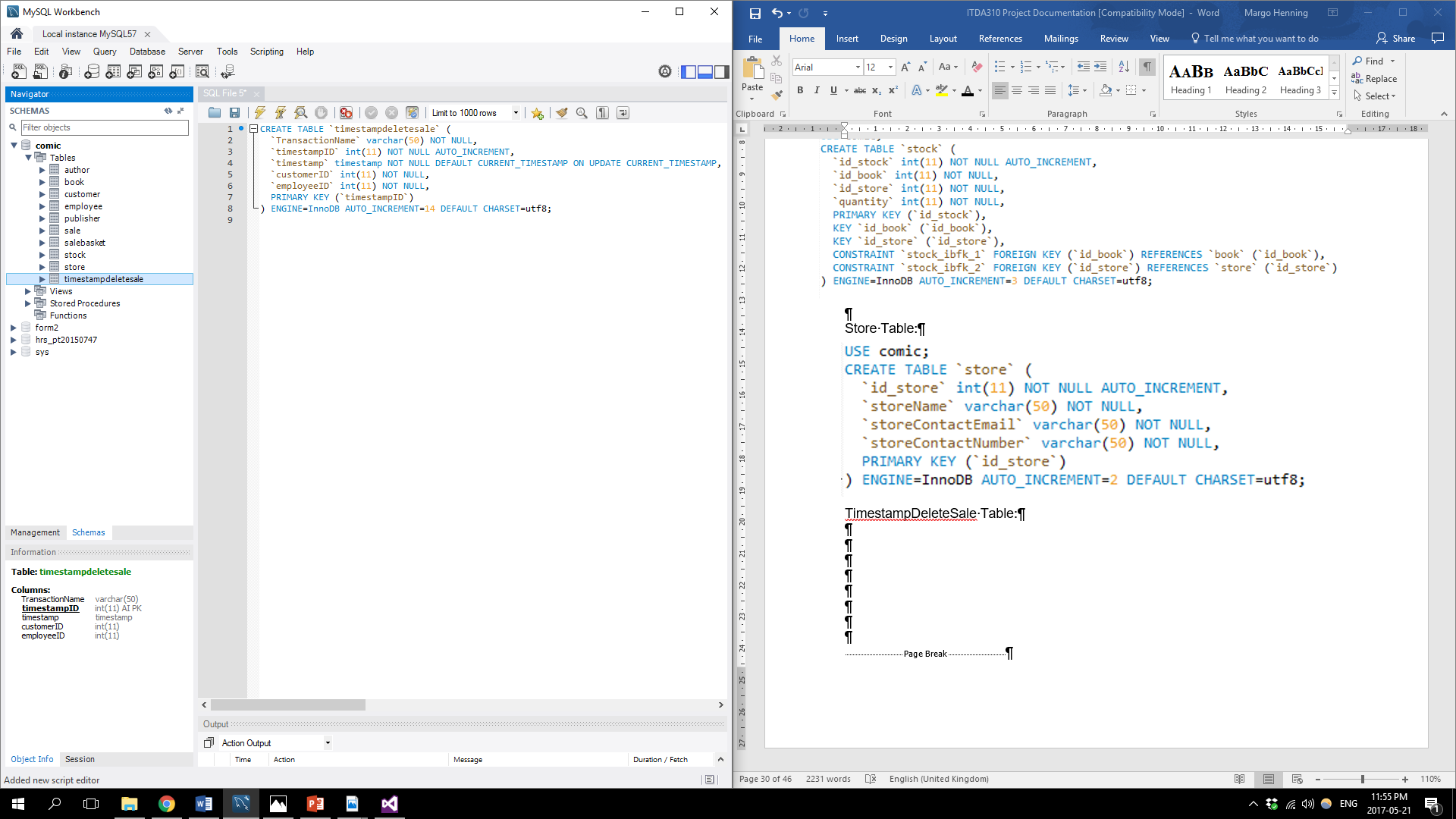


SaleBasket Table:

Stock Table:

Store Table:

TimestampDeleteSale Table:



* Data Manipulation Language (DML)

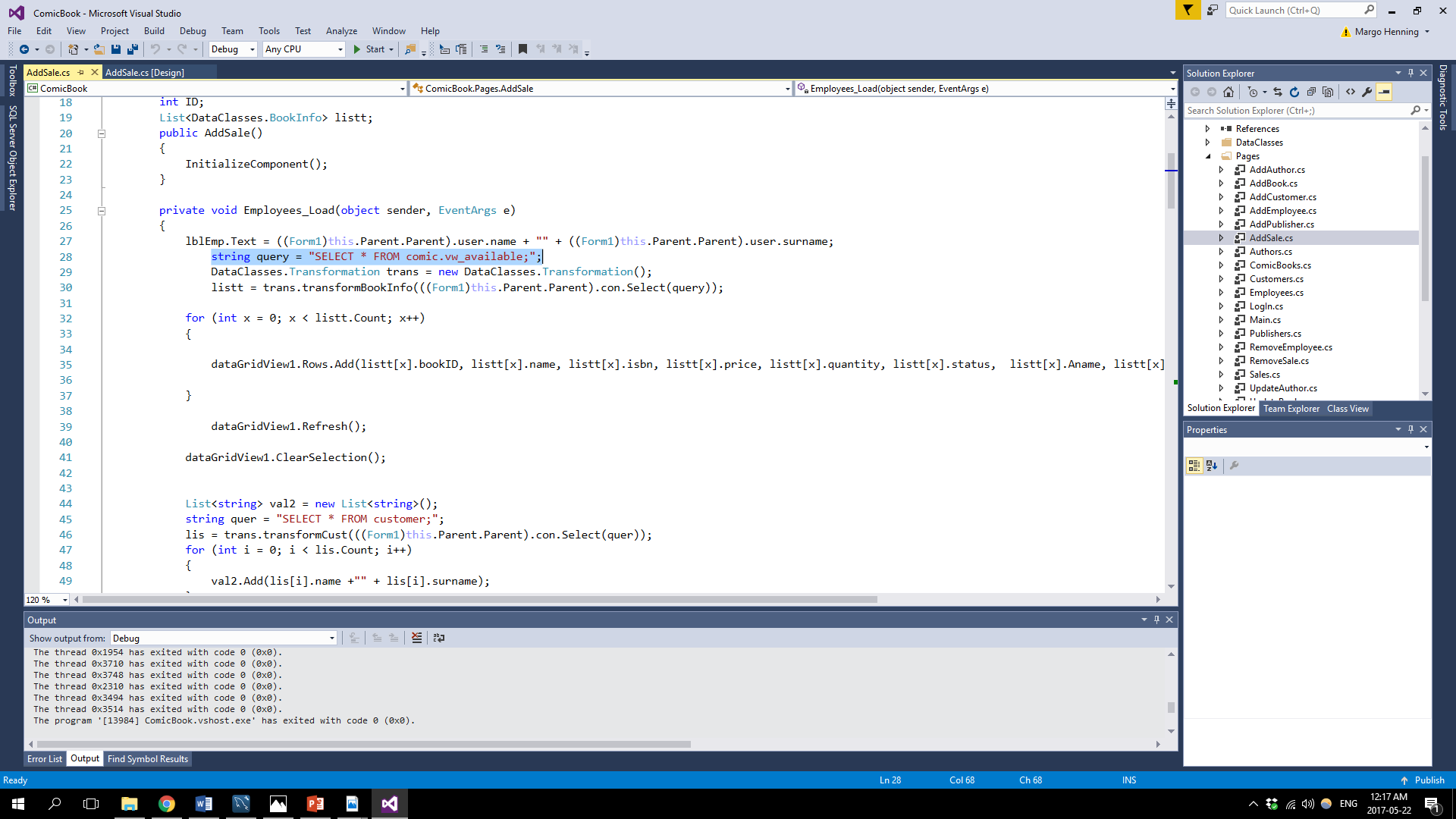
DML statements are mostly used to work with the data in the database tables.

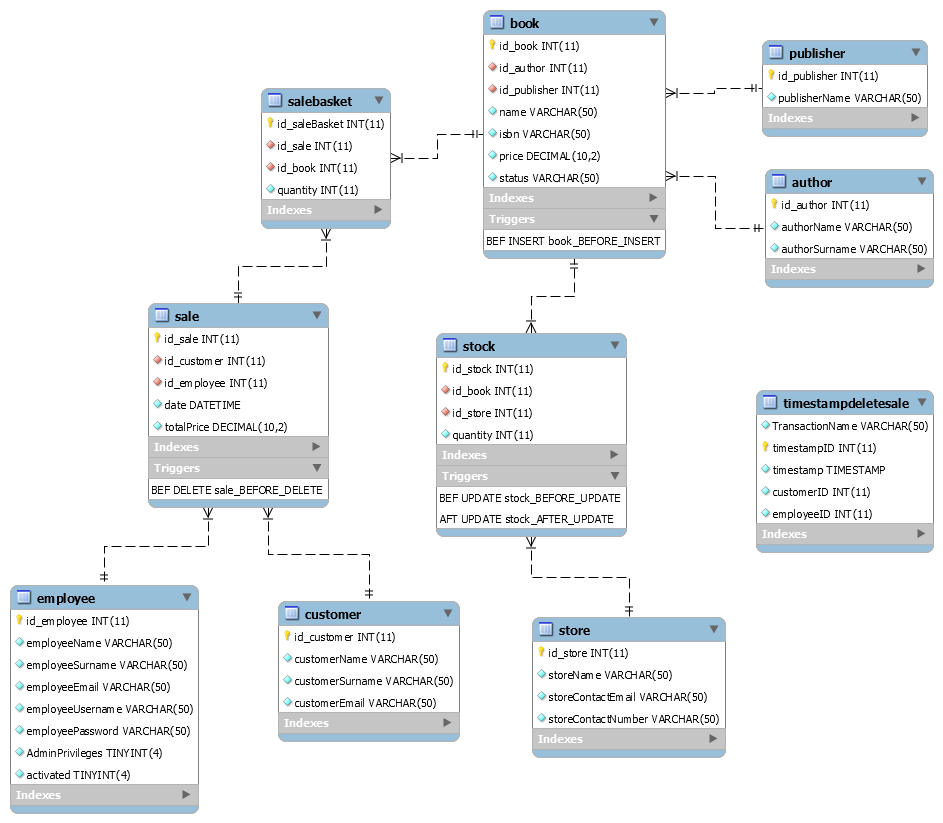
Statements that are considered DML statements includes SELECT, INSERT, UPDATE and DELETE. ((Jewett, 2006))

In this project, stored procedures where created using insert, update and delete DML statements in order for the user and application using the database to efficiently insert data, update data and delete data in each table.

The stored procedures where featured earlier in the documentation under the section, Views, Triggers, Procedures.

Select statements however, was not pre-programmed in stored procedures and were used in views as well as from the application side.

In the screenshot below, it shows how the application used the select DML statement:

After Developing the database, the following diagram demonstrates the final ERD:

# Developing Application and integrating with database

The system that was developed in this project was developed with a database first approach.

The database structure was planned and then developed accordingly.

Once the database was developed and in place the application was developed.

The application is a Windows Desktop Application which allows The Comic Book Store to go about their daily tasks with efficiency and productivity.

The application was developed using the programming language C#.

The IDE (integrated development environment) used to develop the application is known as Visual Studio 2015.

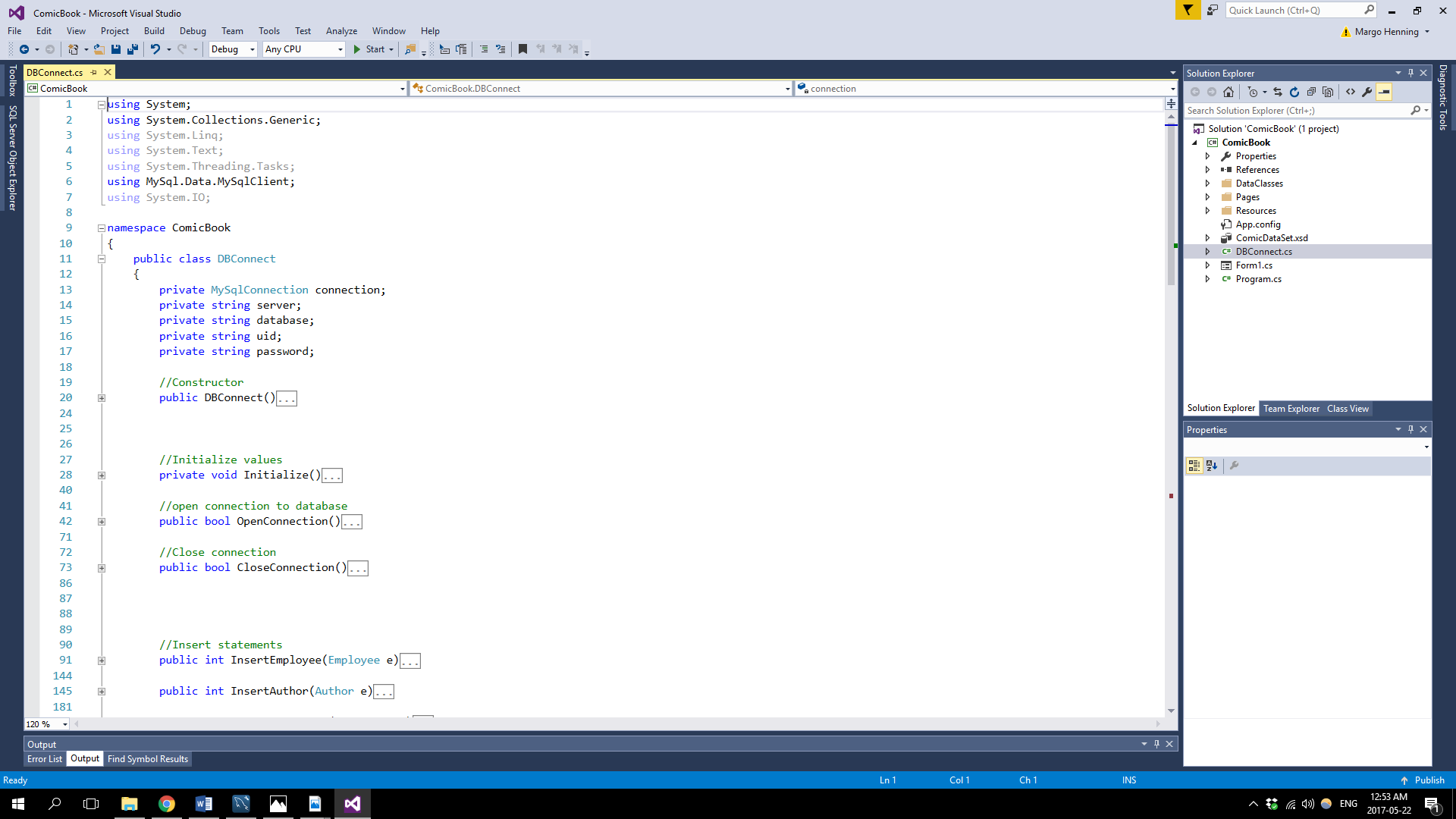
In order for the application to integrate with the MySQL database a few preparations had to be made.

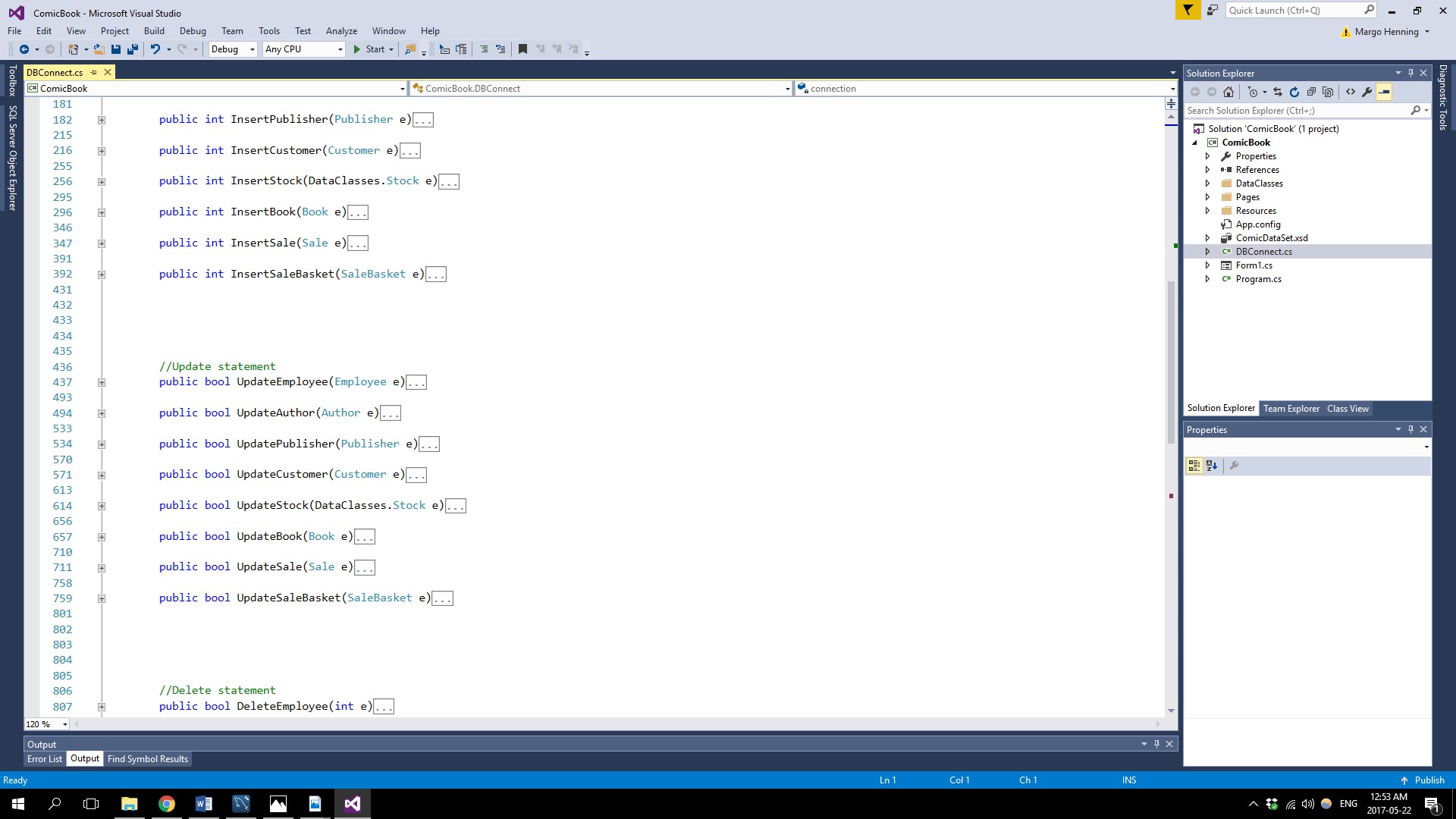
These preparations include the following:

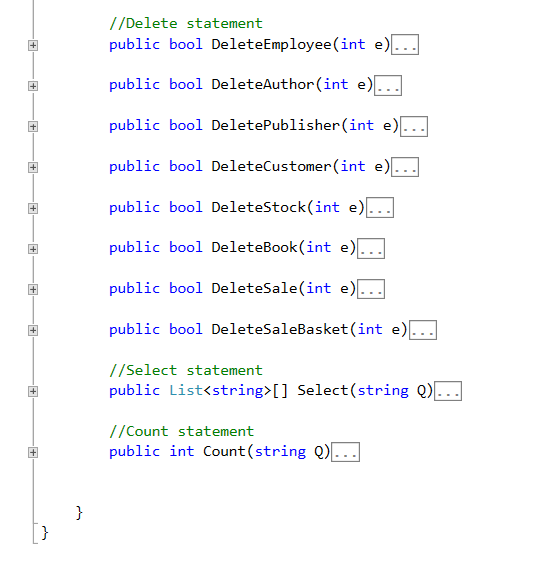
* MySQL Connector Net 6.9.9 had to be installed
* MySql.Data.dll file had to be added to the application as a reference
* MySql.Data.Entity.EF6.dll file had to be added to the application as a reference
* A class was created in the application that served the purpose of connecting and disconnecting to the database as well as calling and executing queries to the database.

The class that was created is known as DBConnect.cs

The following screenshot shows the structure of the DBConnect.cs class:

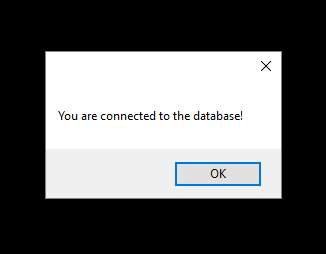


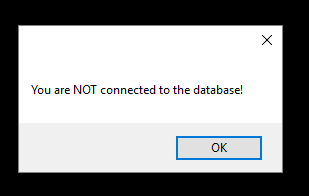




# Connecting to the database

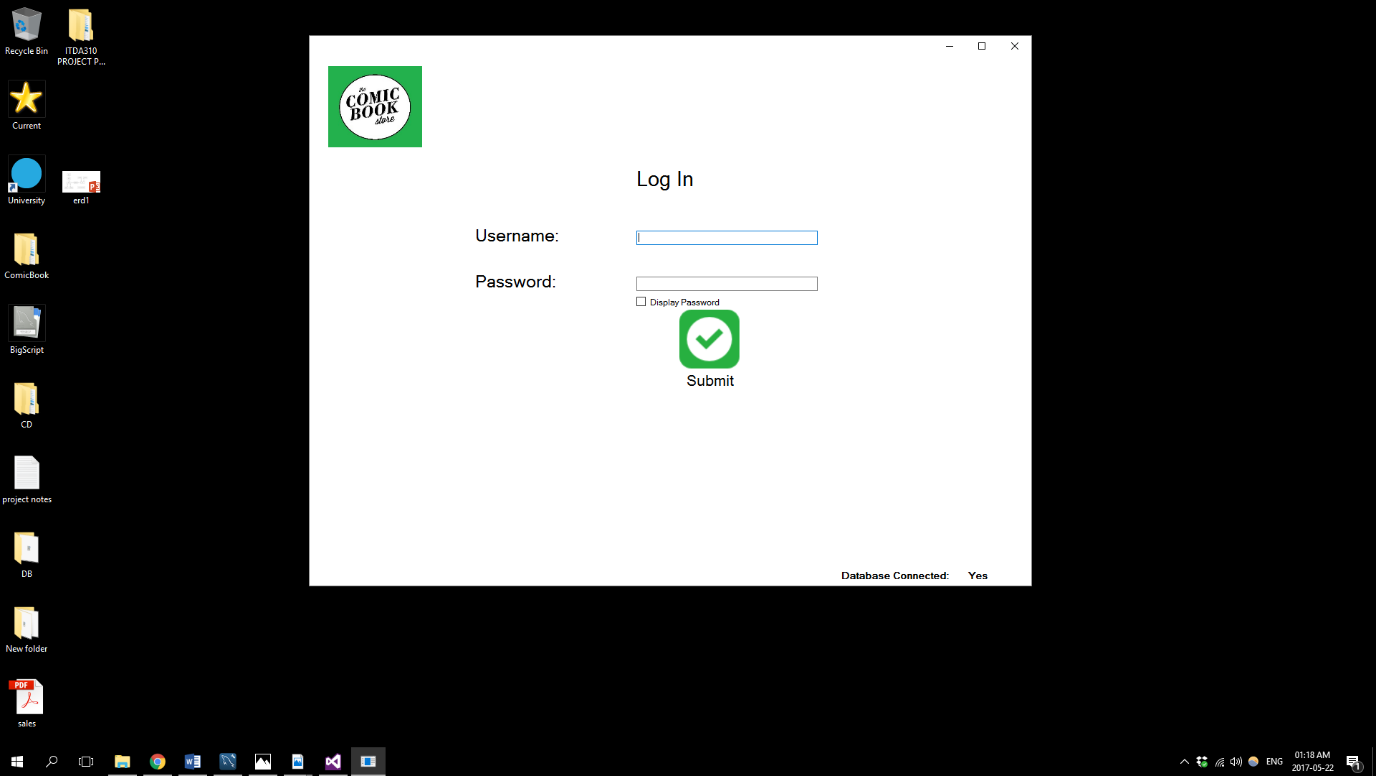
When you start up the application the following window will show if the application made a successful connection to the database.



In the case of starting up the application and connecting with the database was unsuccessful the following window will show:

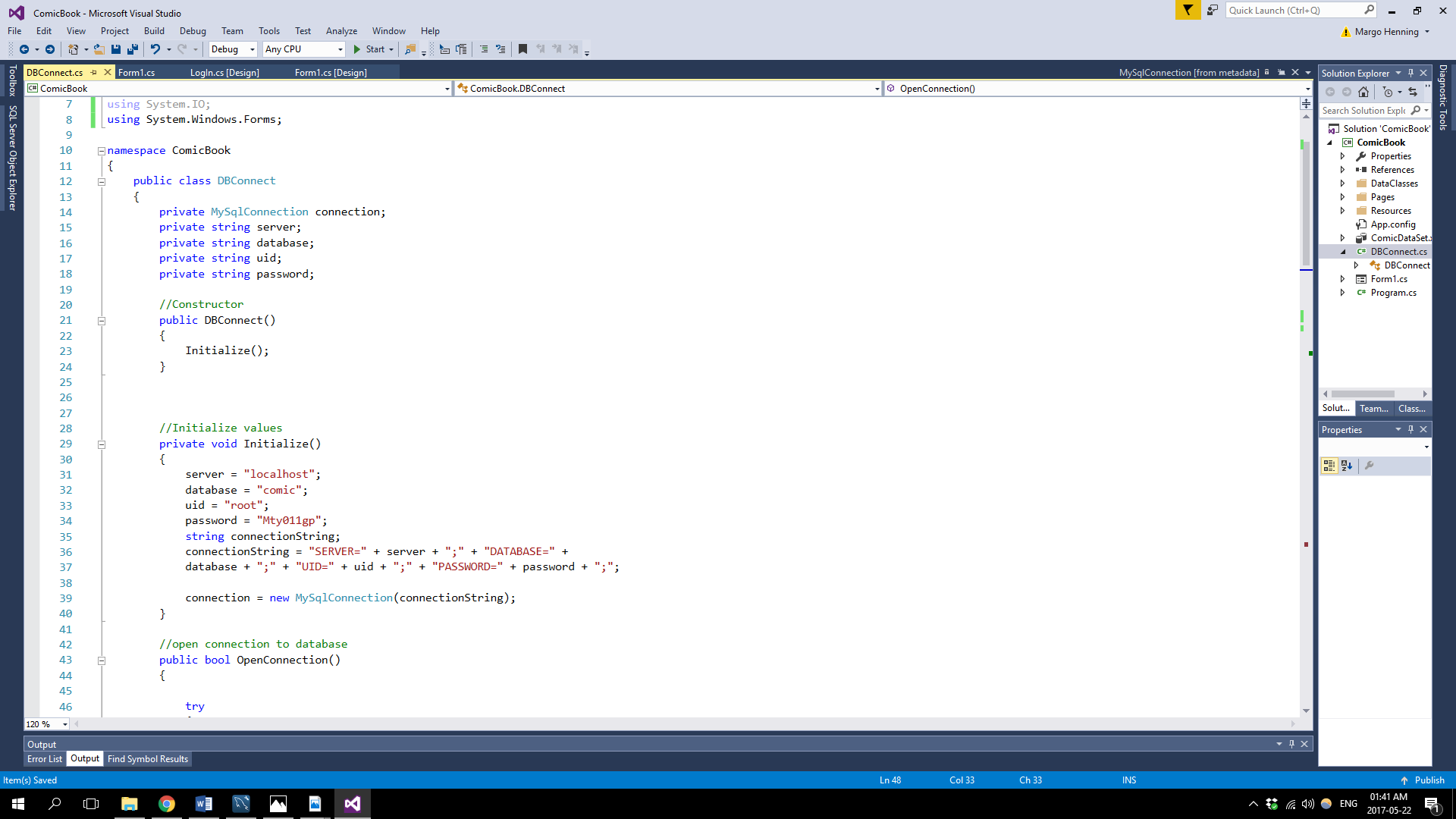
During the time that the application is running a message will be visible at the bottom of the application frame indicating whether the database is connected or not.

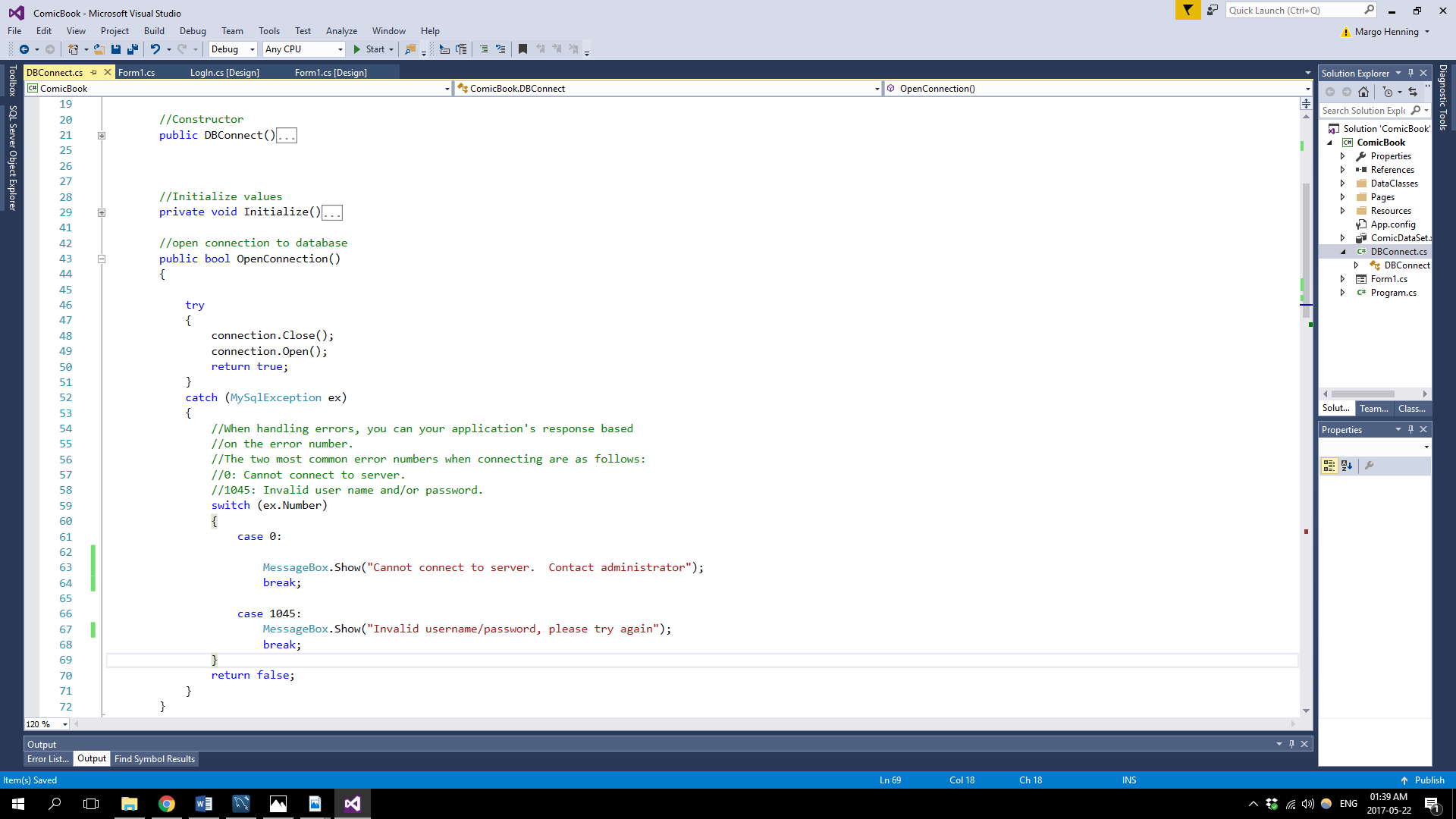
The following screenshot shows the message at the bottom of the application frame:



The process of connecting to the database from the application’s side includes the following steps.

* Create an instance of the DBConnect class
* In the constructor of the DBConnect class it calls a method Initialize() which executes the following code:

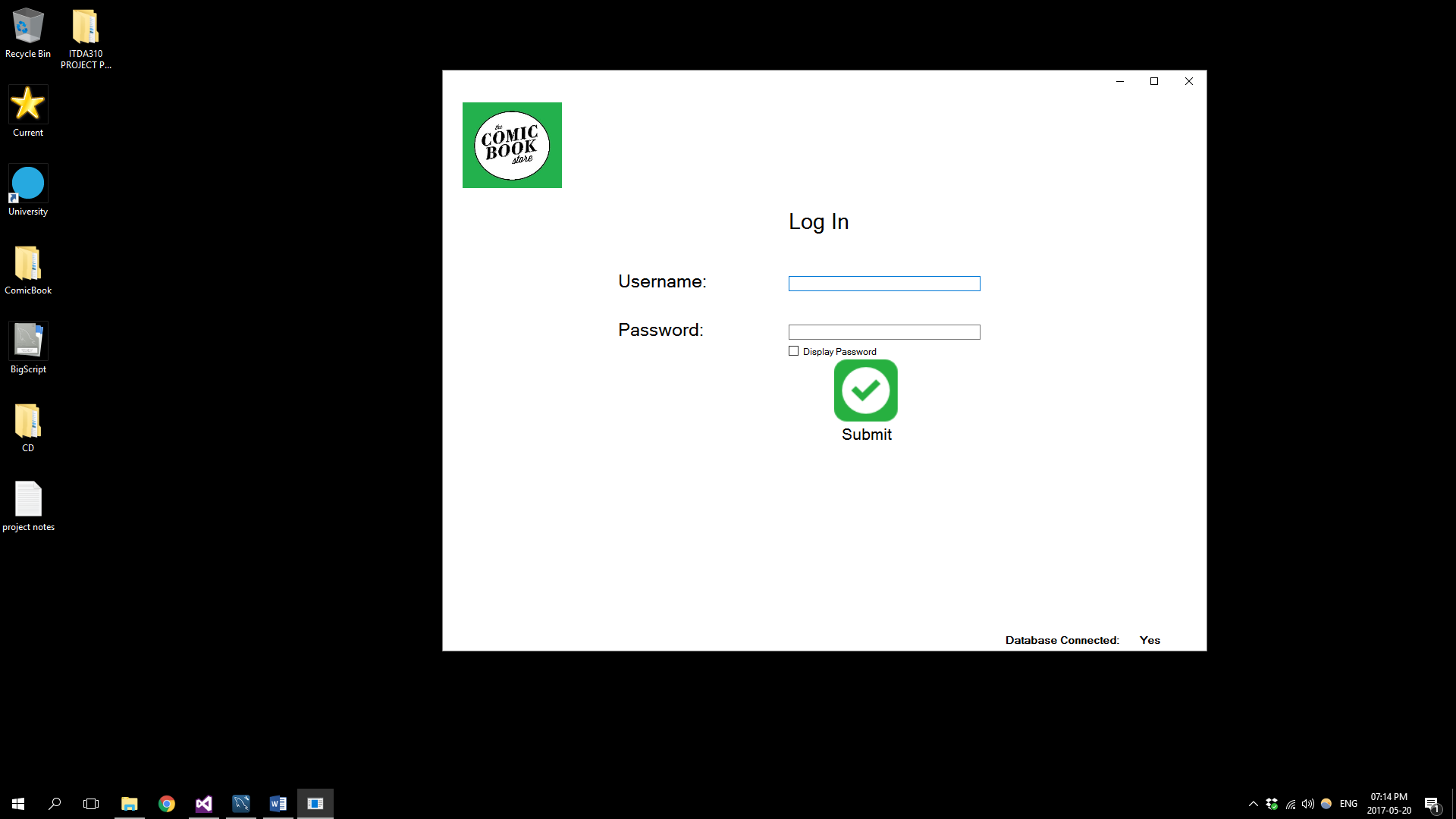


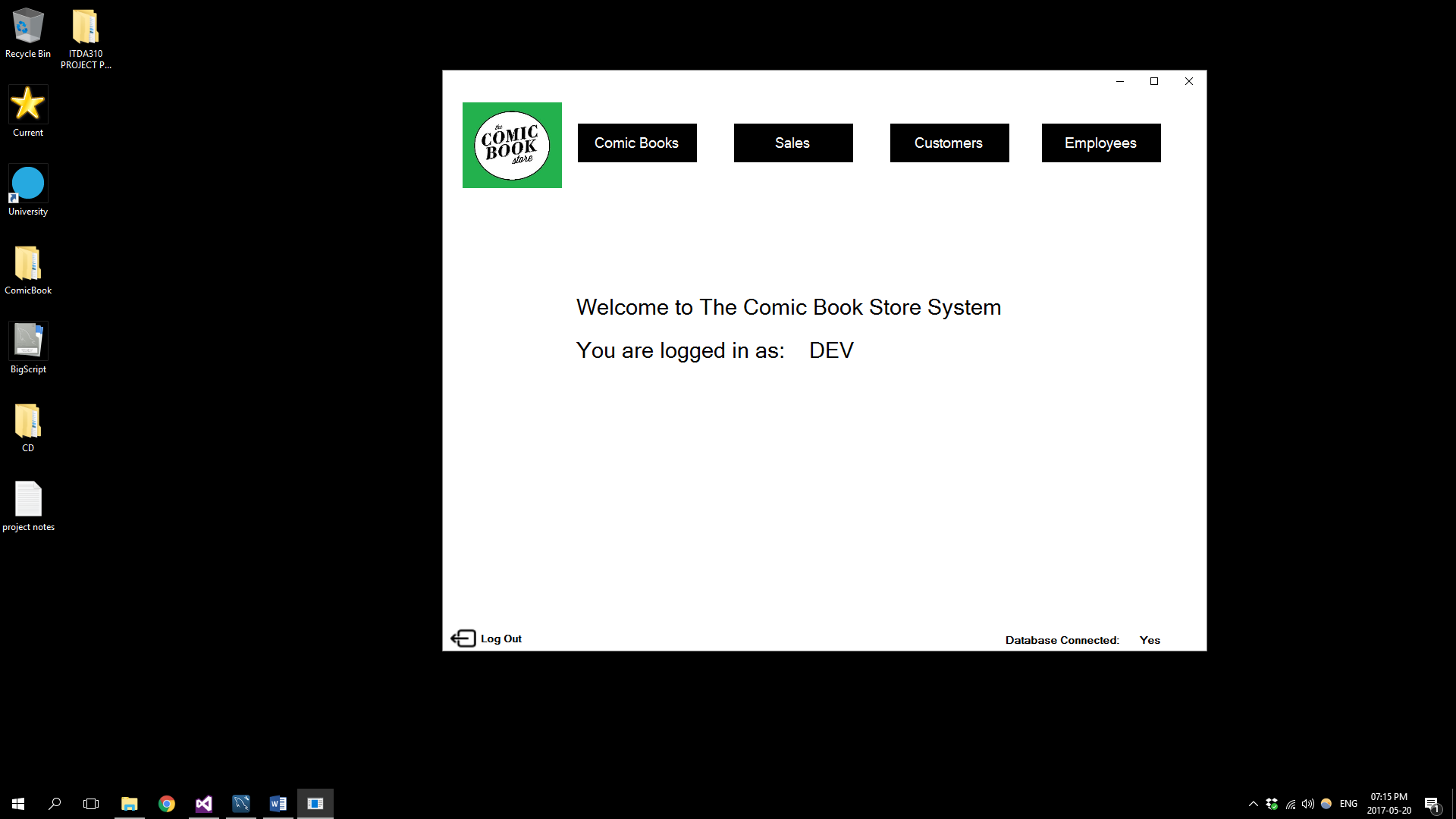
* Use the instance to call the OpenConnection() method
* Inside the OpenConnection() method the following code are executed

# User Manual

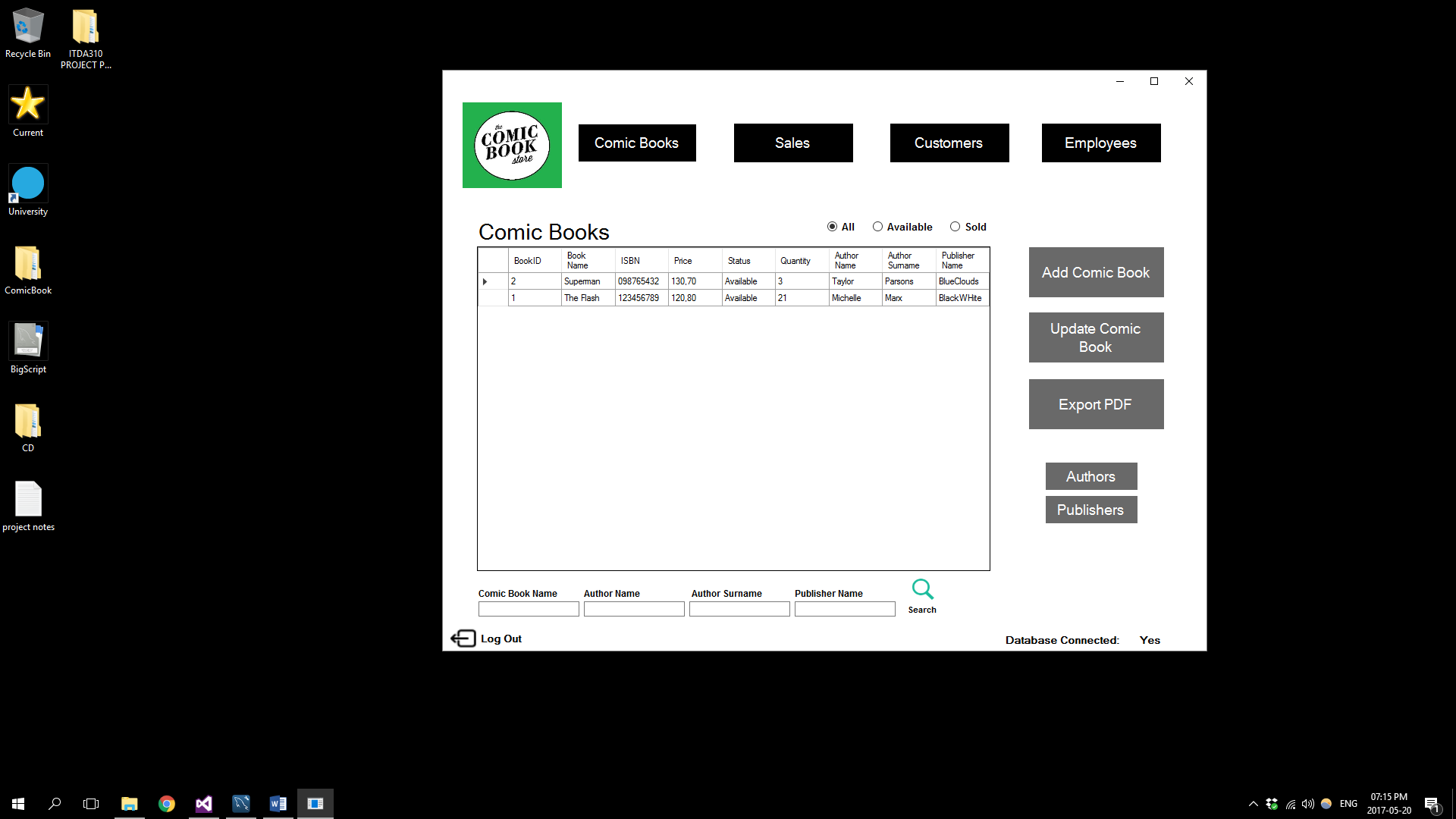
When the application starts up the user will be informed whether the application connected to the database successfully.

The Log In Screen will then appear:

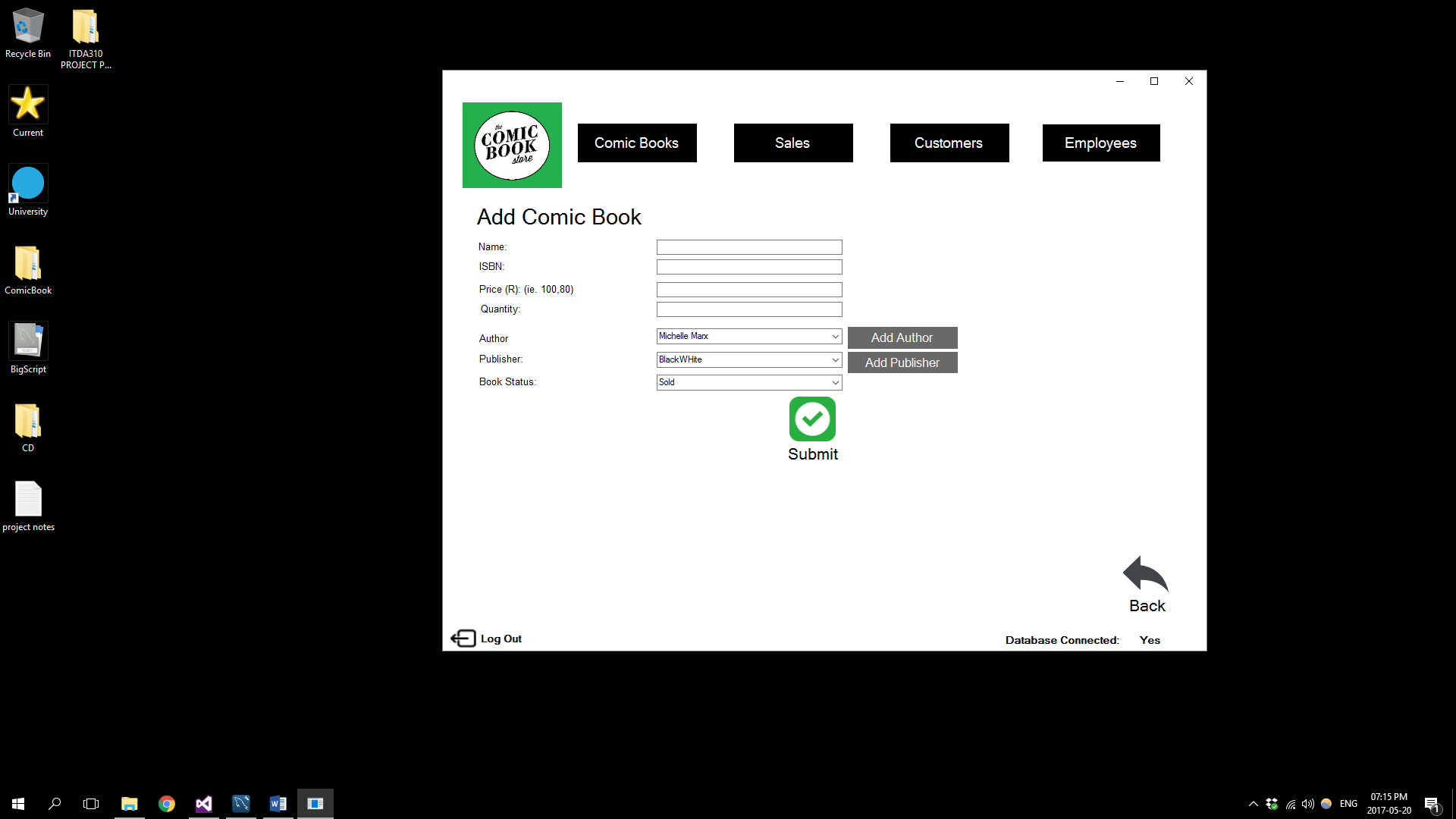


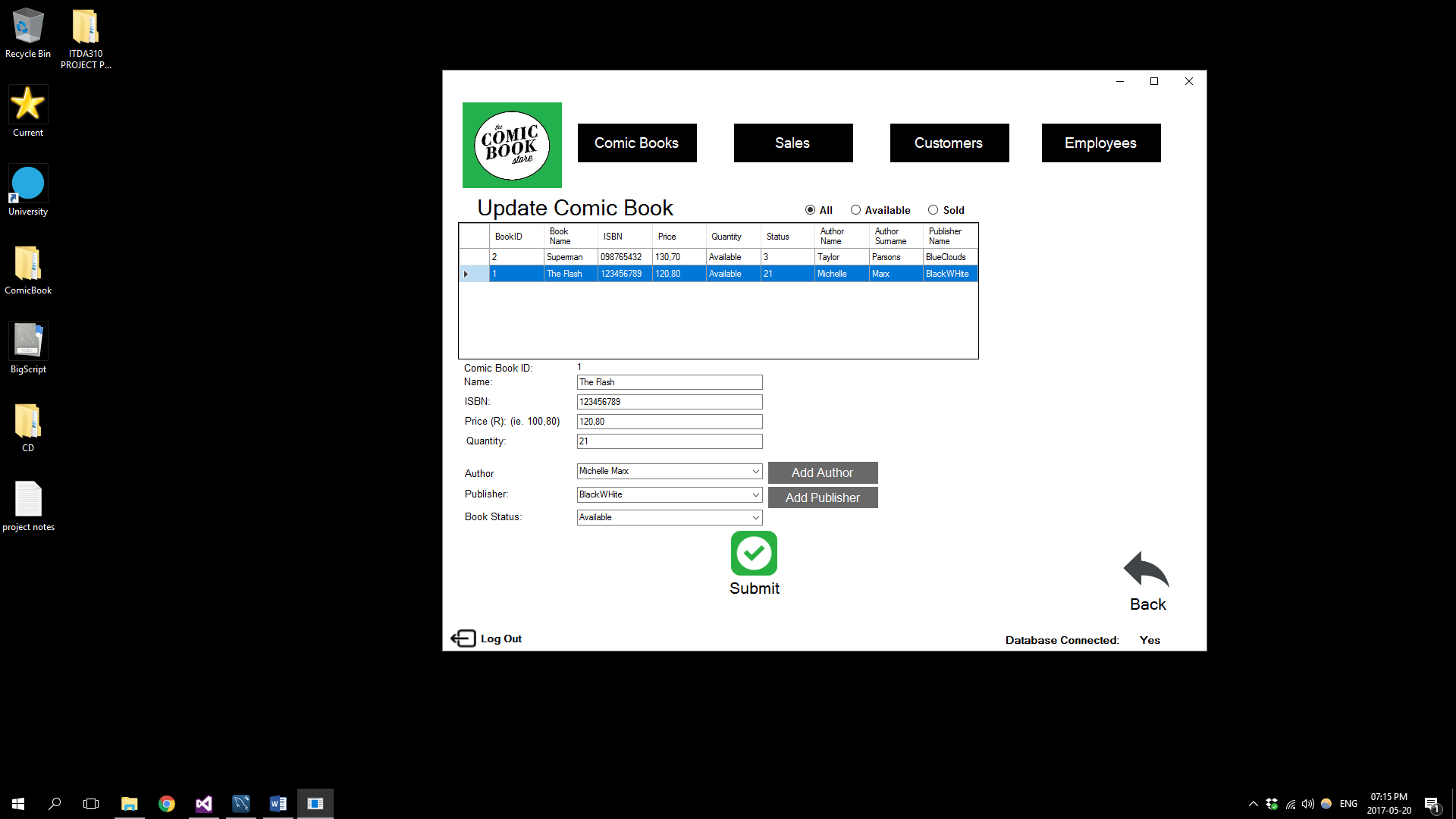


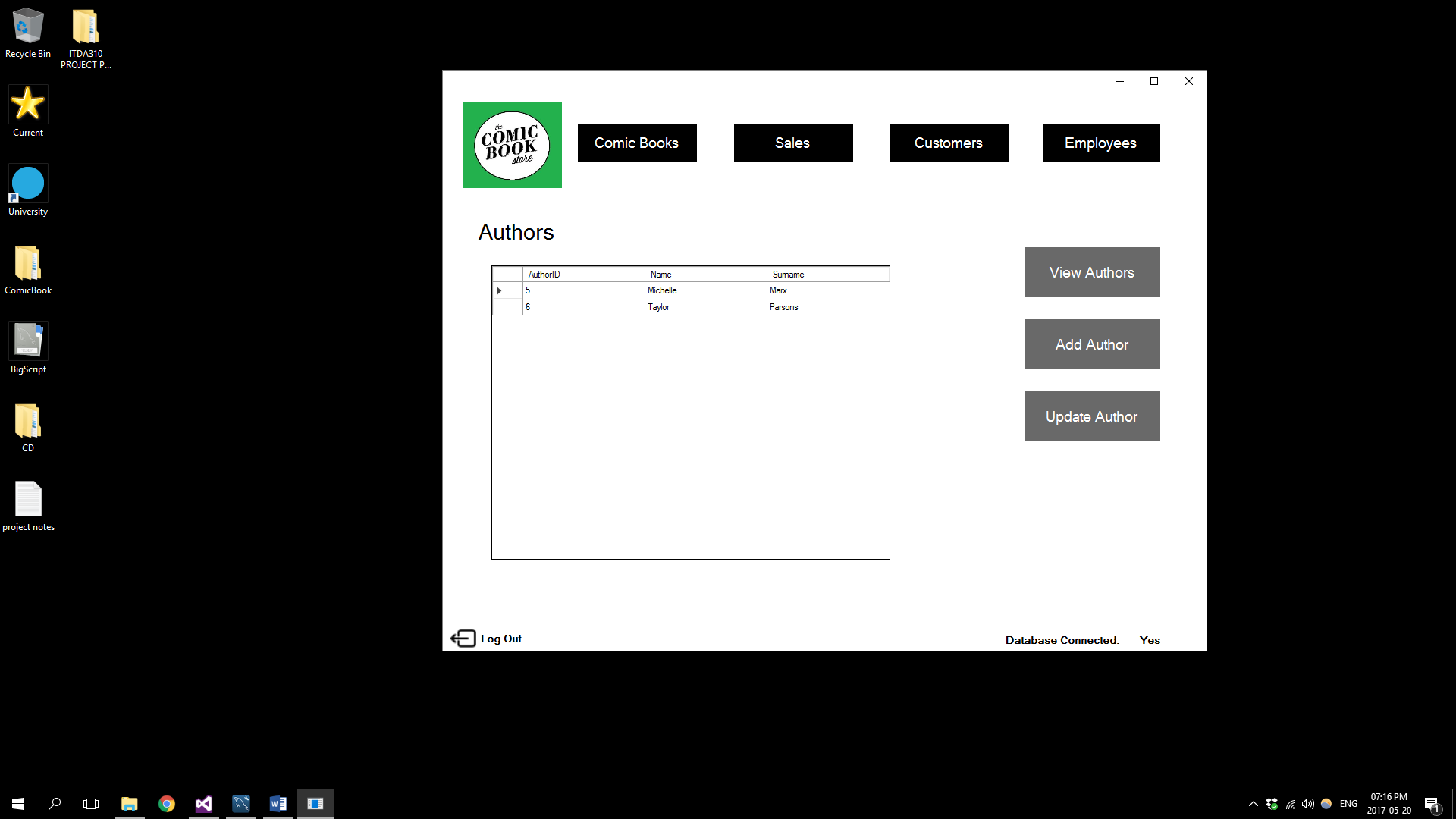
After you have logged in succesfully the user will be created by the following log in screen

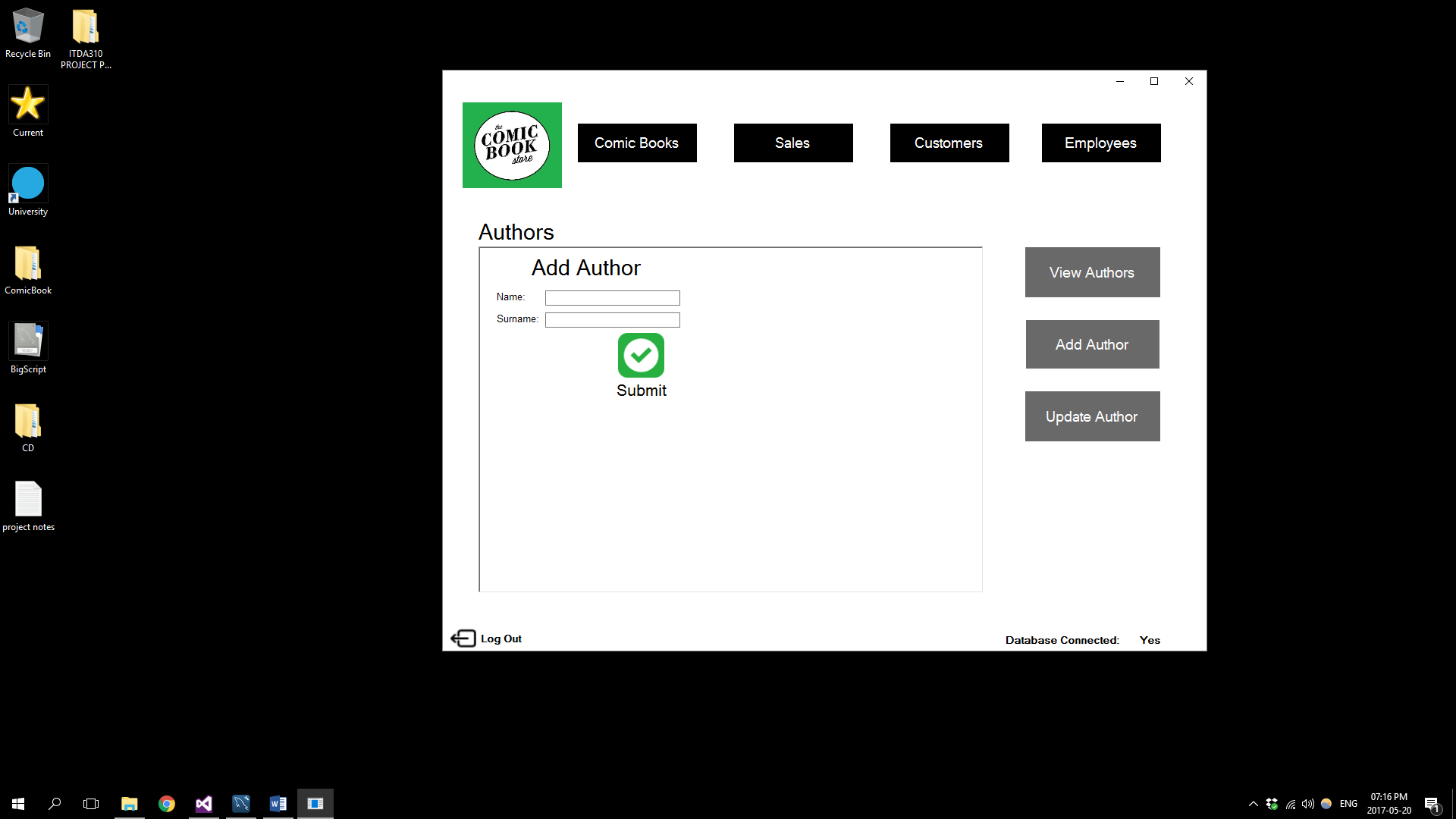
Selecting the Comic Books button will open the following screen which is the main Comic Books screen:

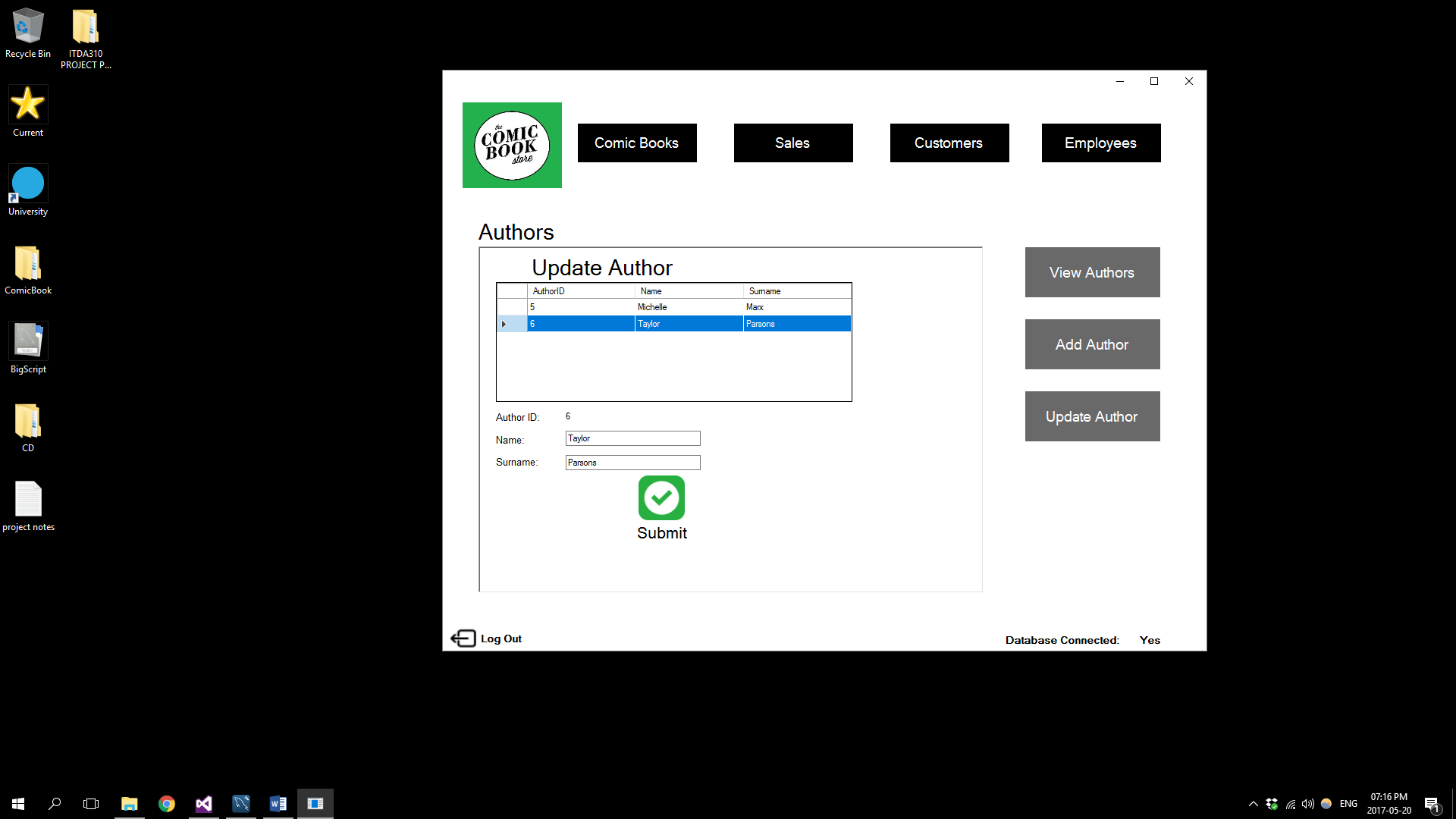
Selecting the Add Comic Book button will take you to the following Add Comic Book screen

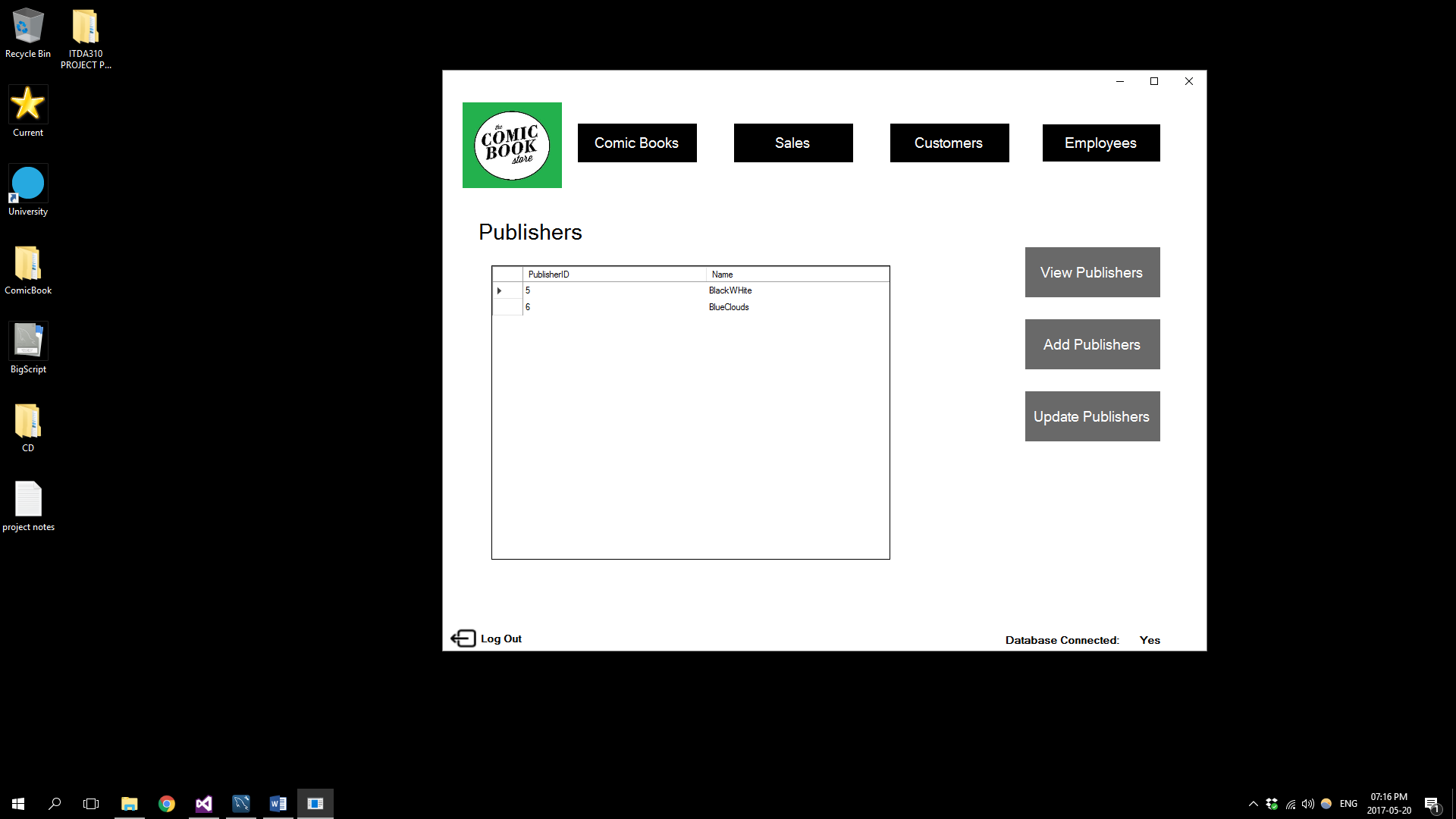


Selecting the Update Comic Book button will take you to the following Update Comic Book Page

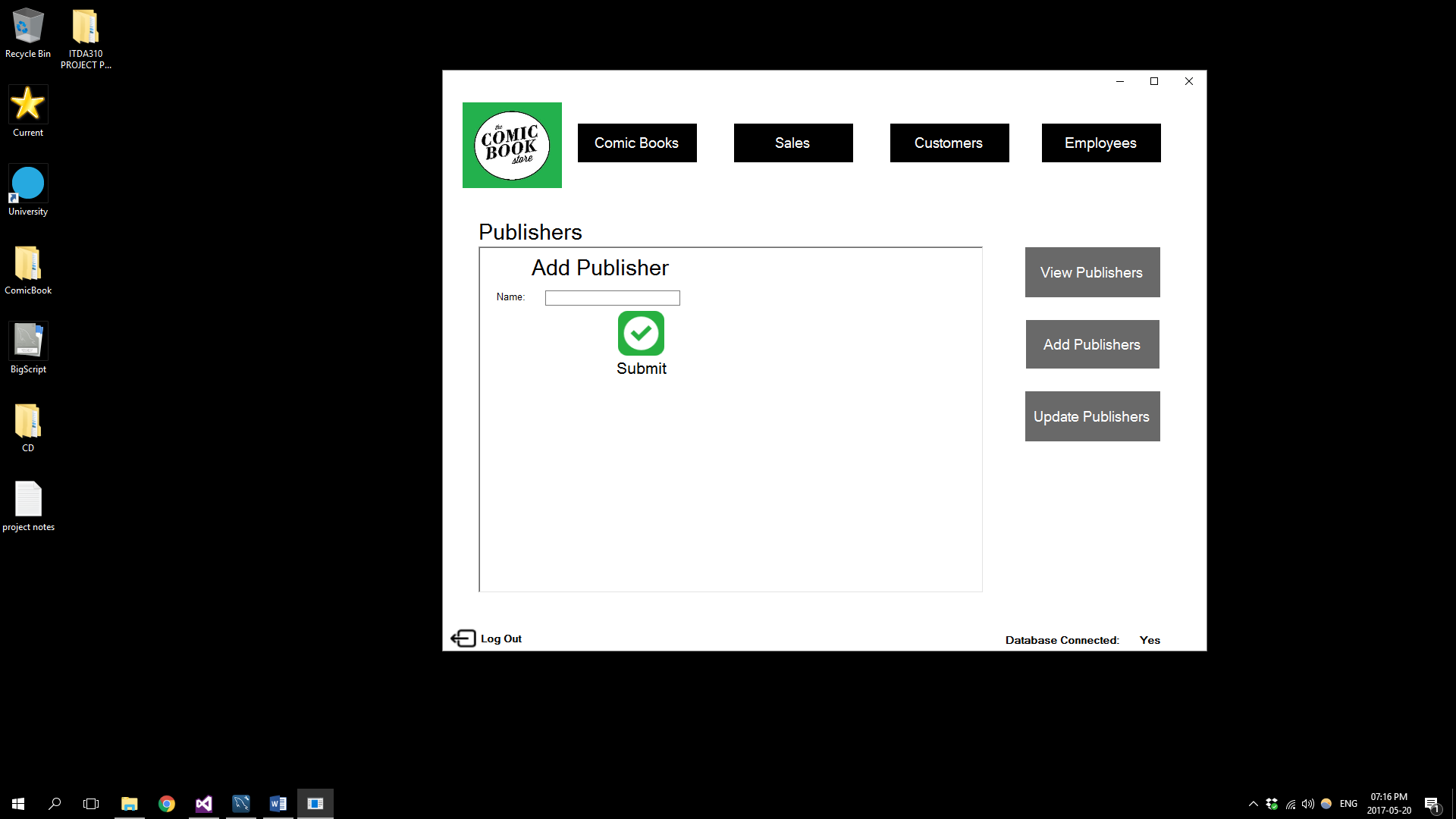
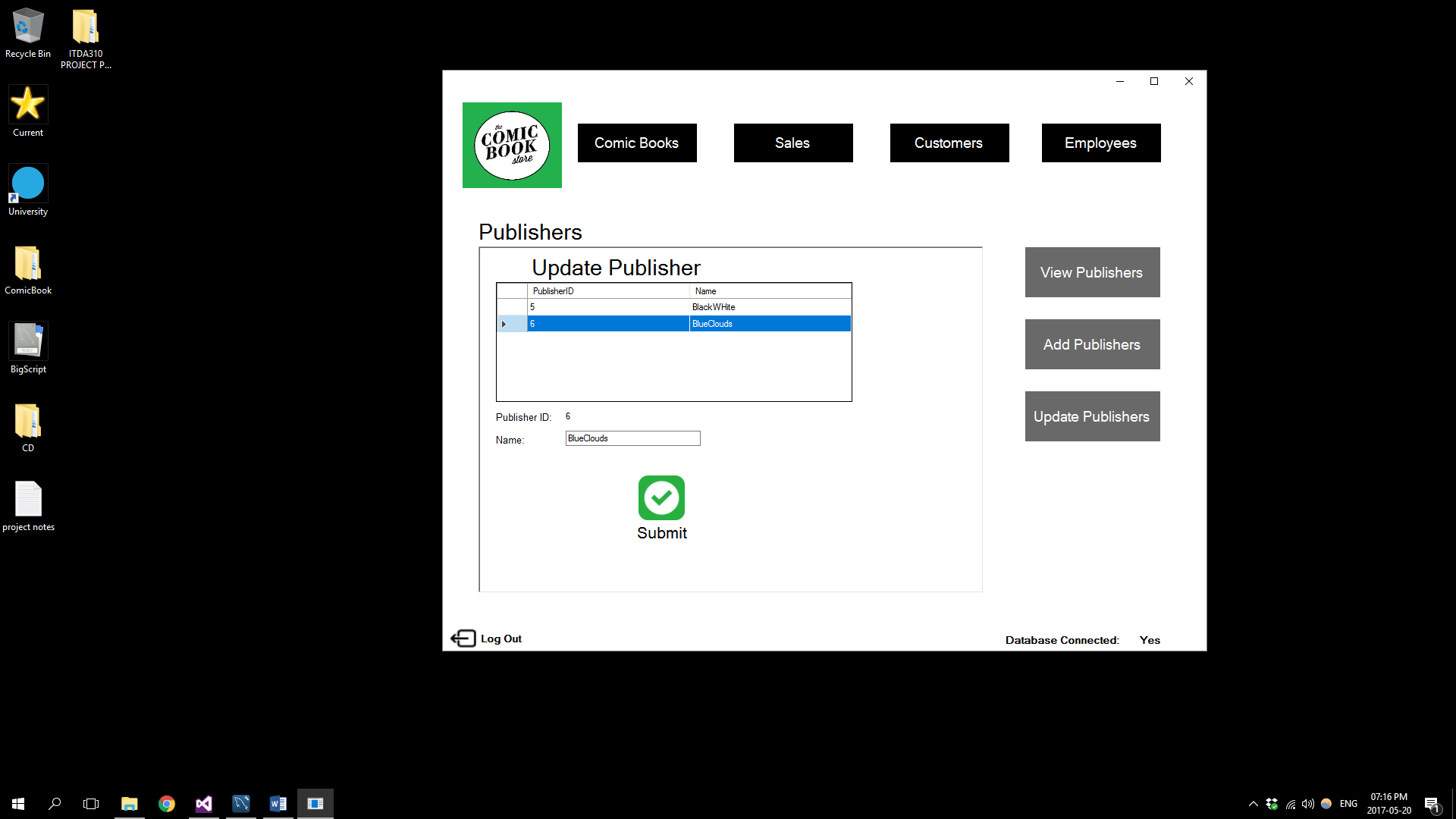
Selecting the Authors button on the main Comic Book screen will take you to the following main Authors screen

Selecting the Add Authors button will take you to the following Add Author screen

Selecting the Update Author button will take you to the Update Author page:

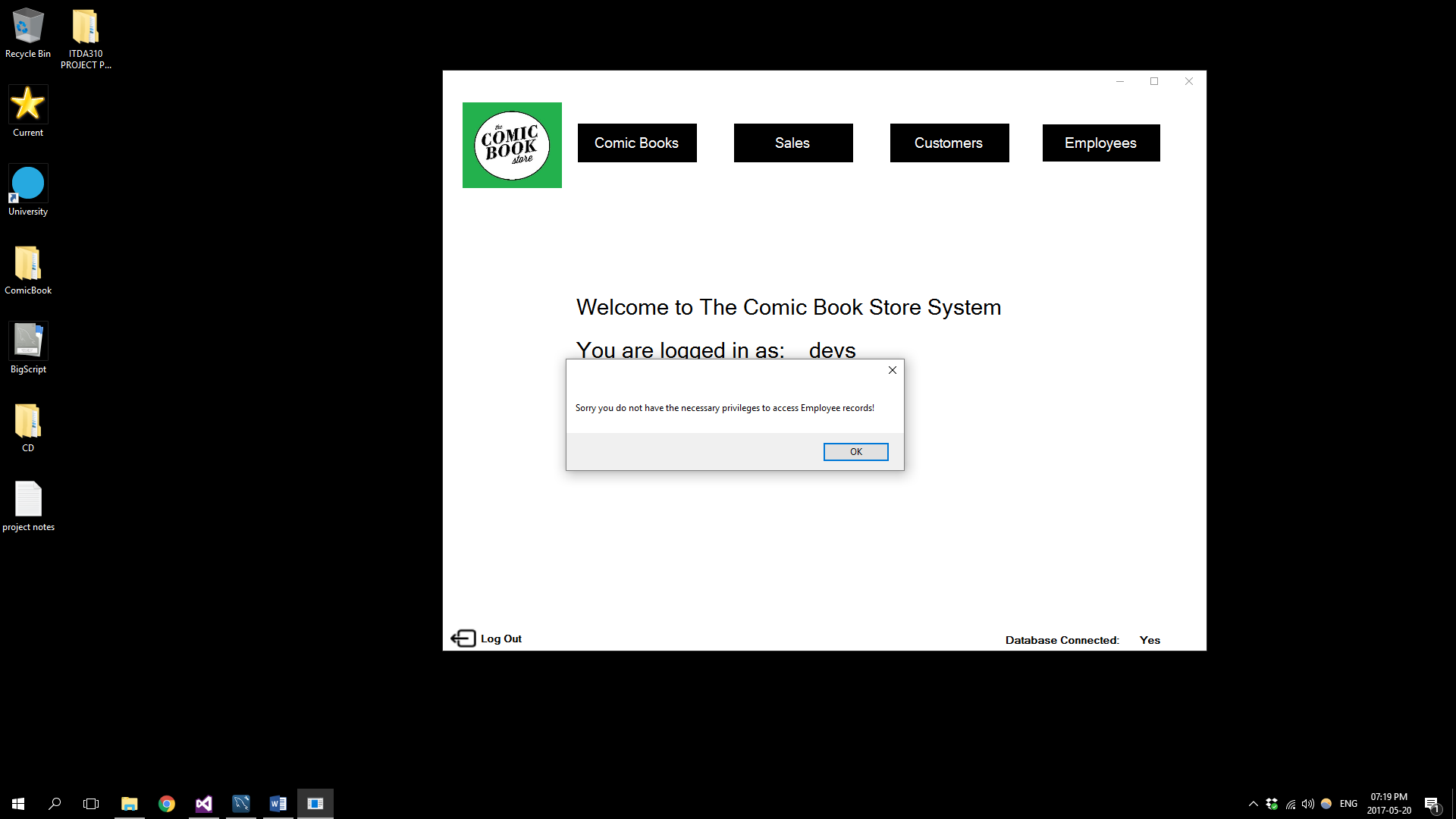
Selecting the Publishers button on the main Comic Book screen will take you to the following main Publishers screen

Selecting the Add Publishers button will take you to the following Add Publisher screen

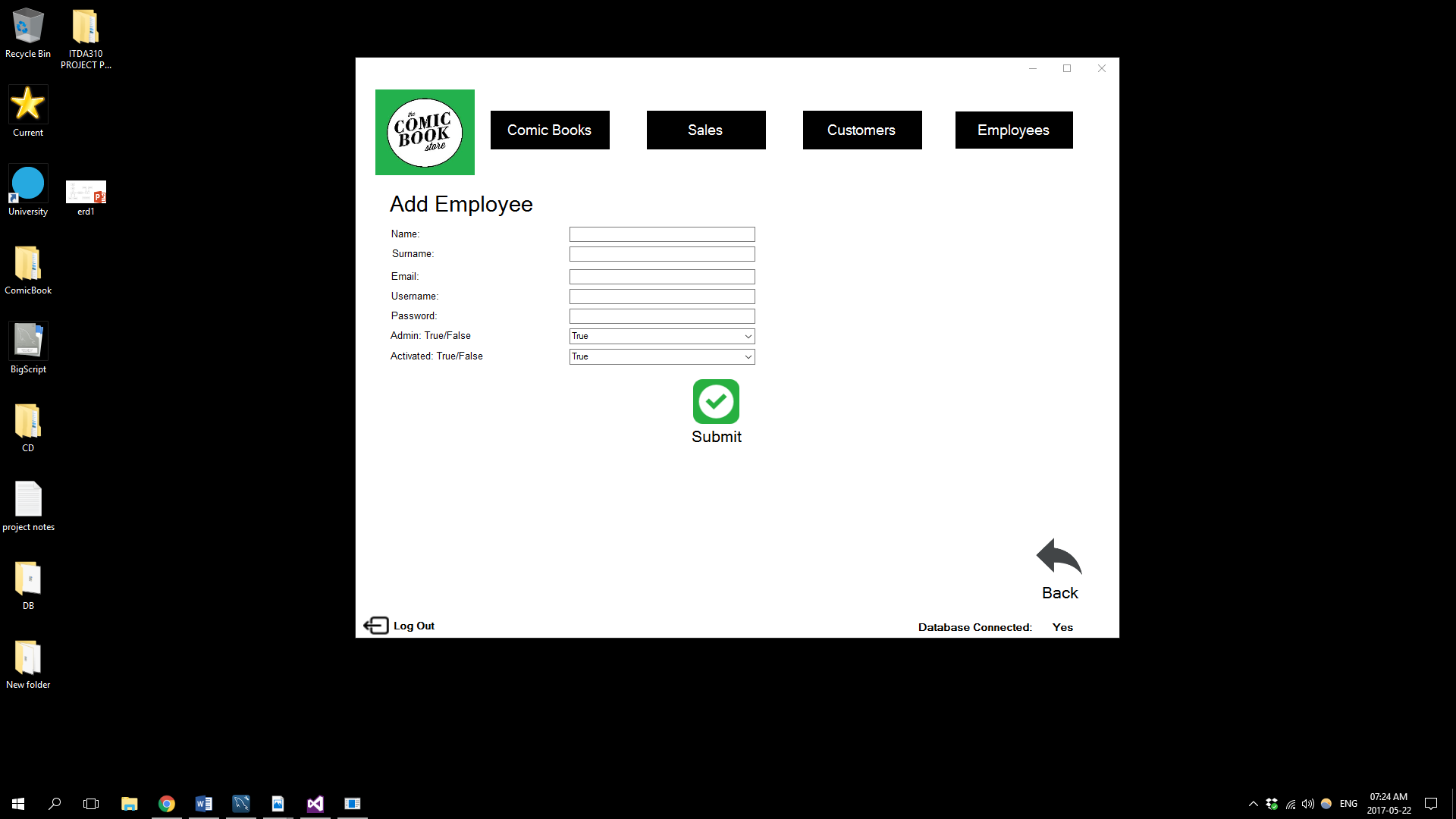
Selecting the Update Publisher button wil take you to the following Update Publisher Screen:

Selecting the Employees button will take you to the following main Employees page if you have admin privileges:

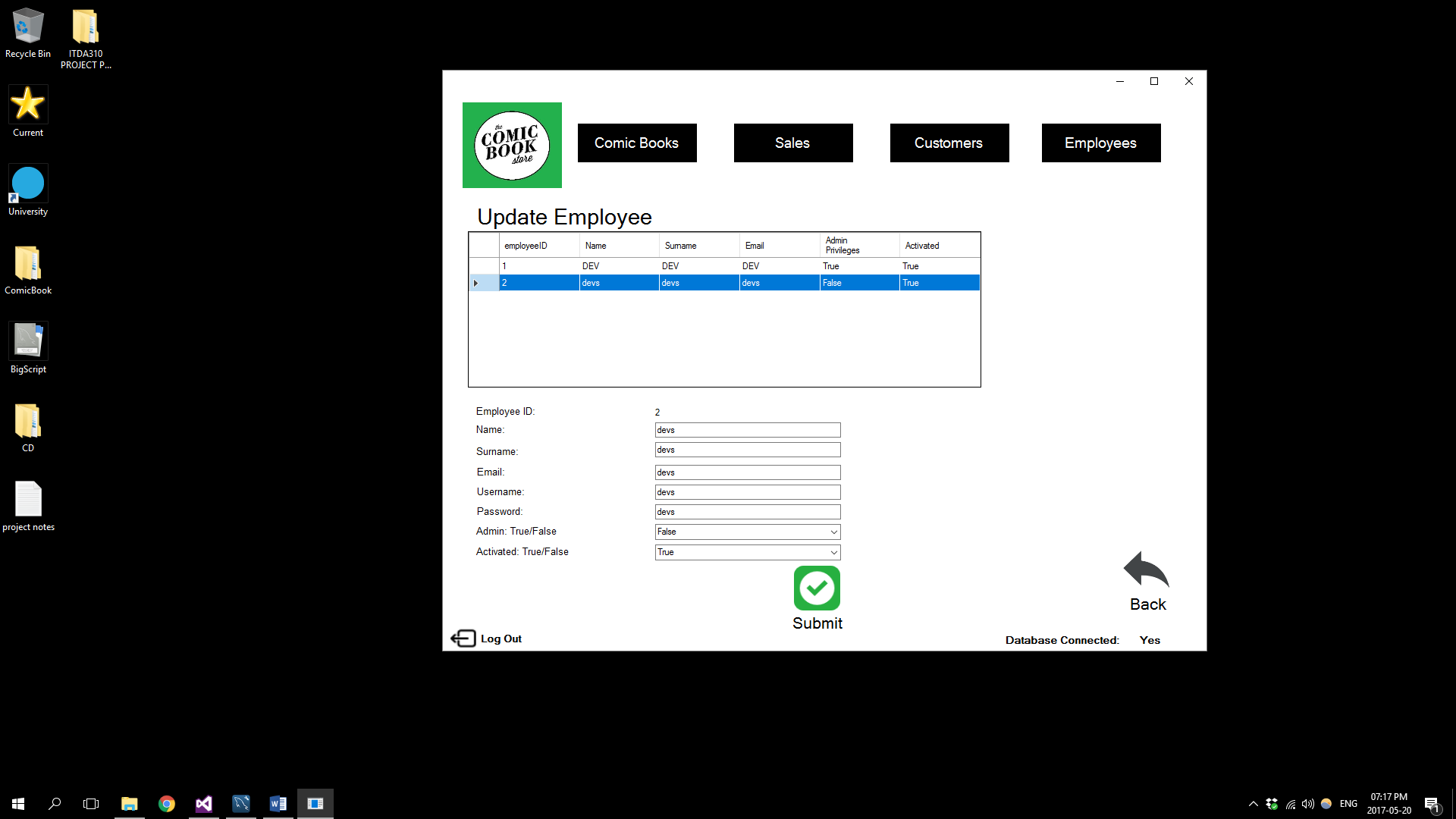


If you do not have admin privileges the following message will be seen:

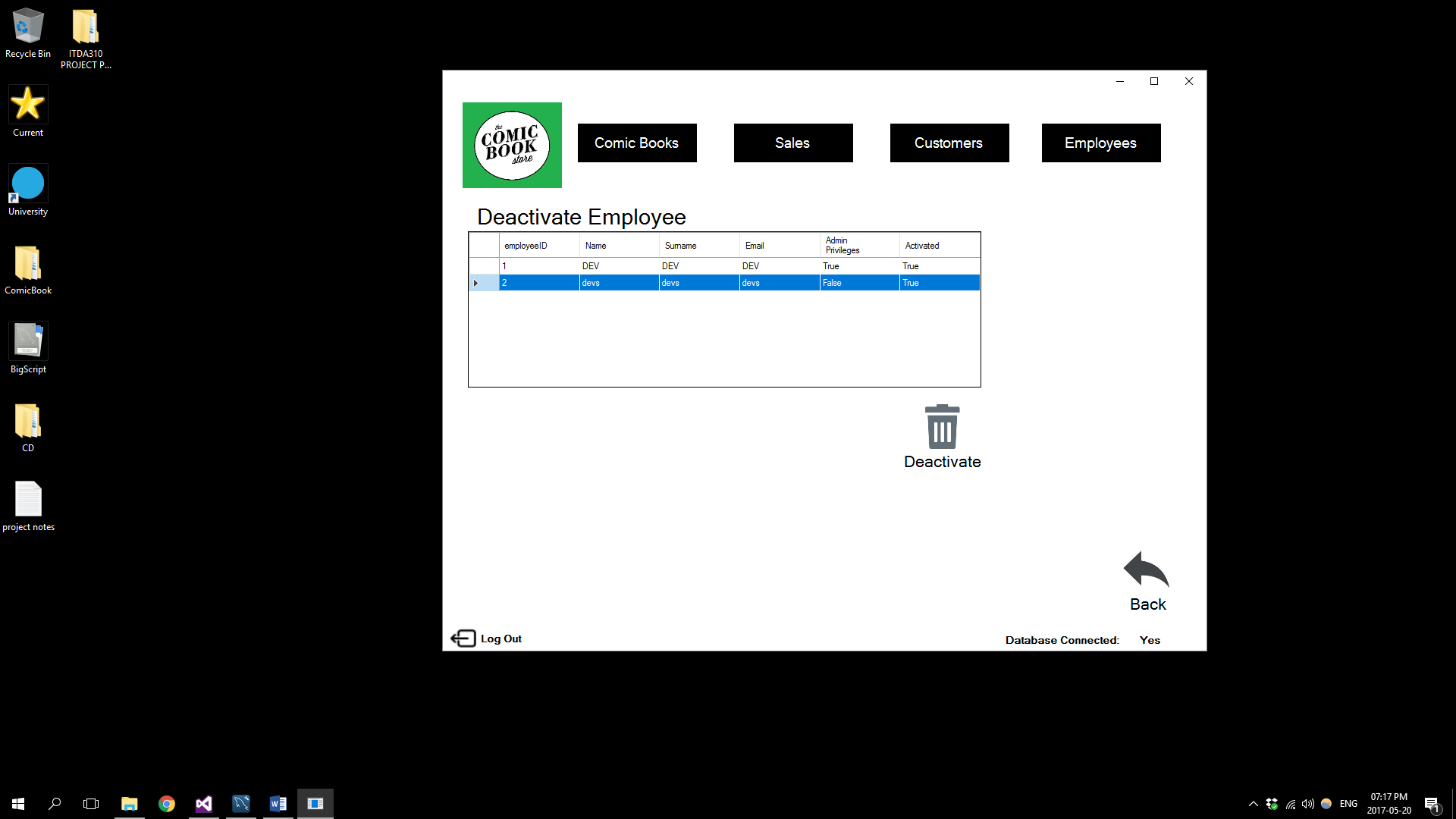
Selecting the Add Employee button on the main Employees screen will take you to the following Add Employee screen



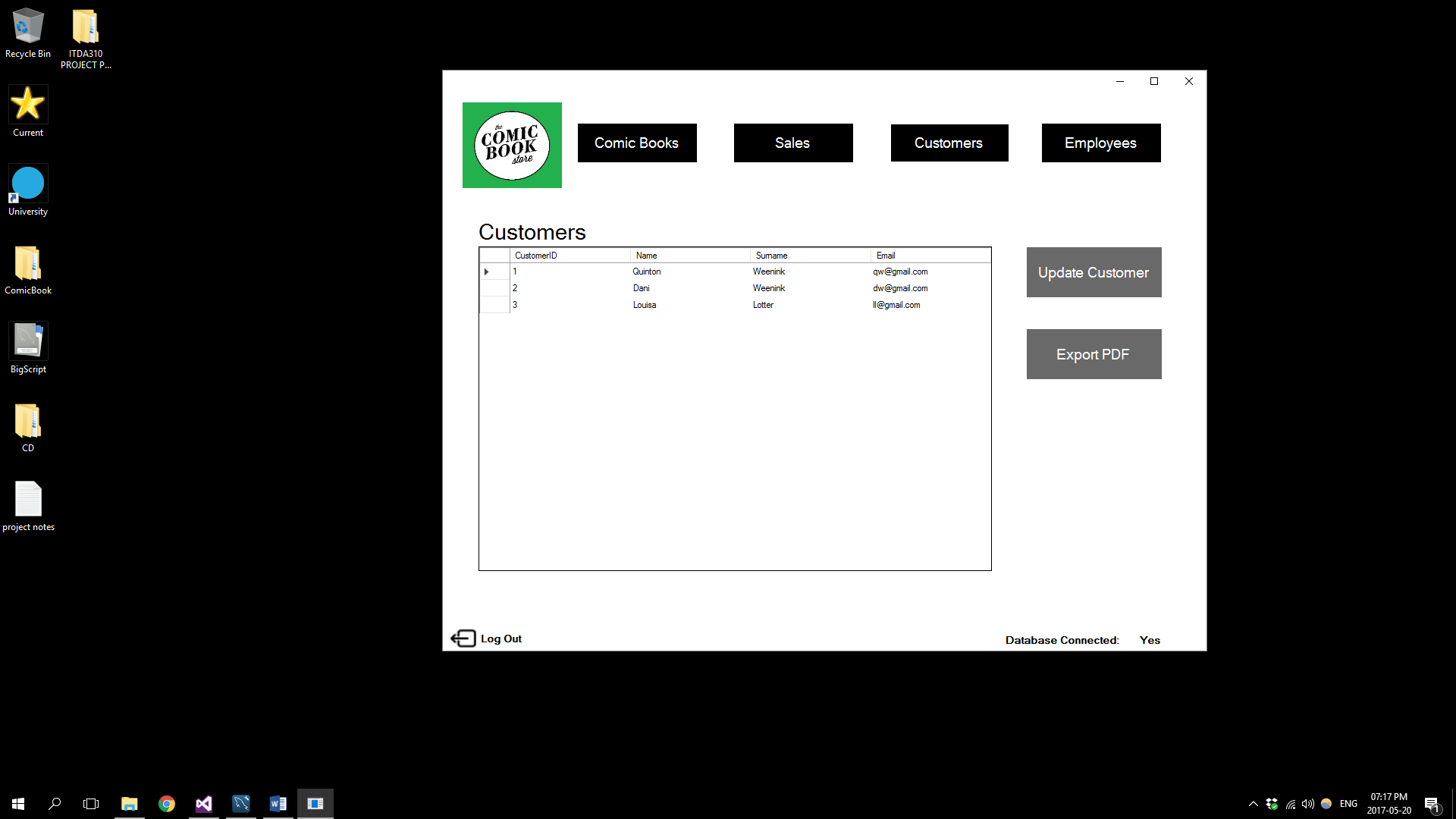
Selecting the Update Employee button on the main Employees screen will take you to the following Update Employee screen:



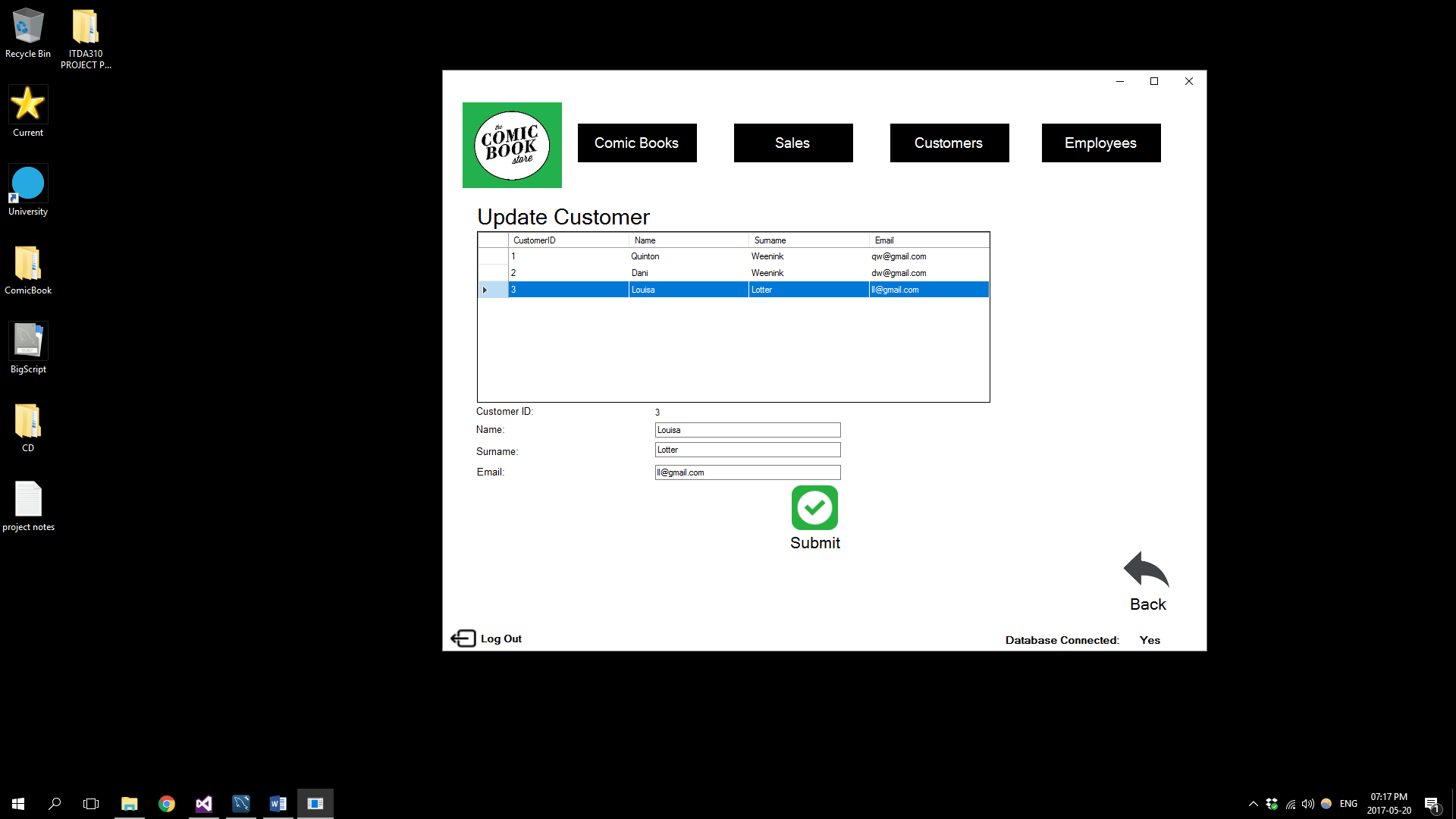
Selecting the Deactivate button on the Employees main screen will take you to the followingDeactivate Employee screen:



Selecting the Customers Button will take you to the following main Customers screen:



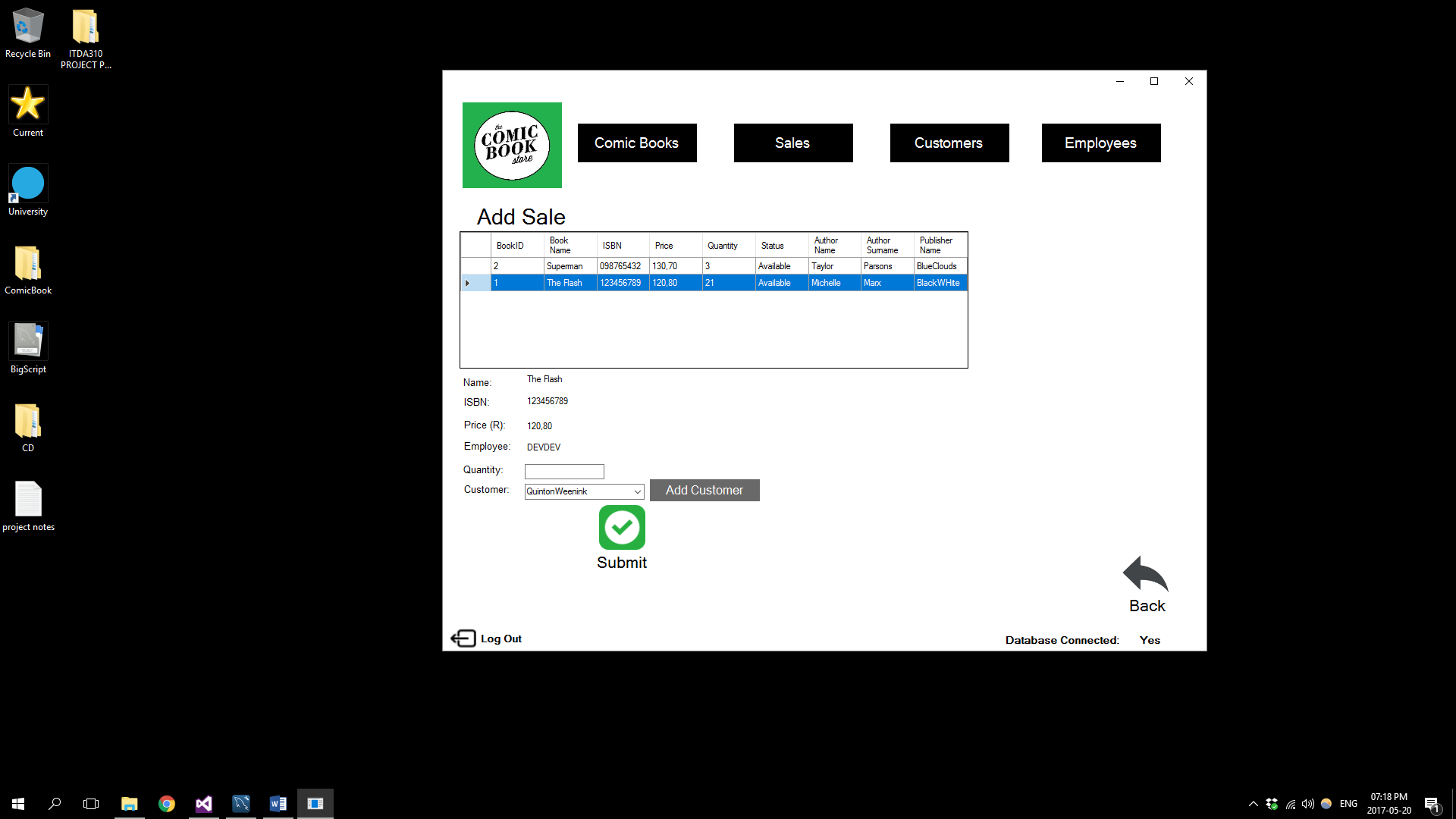
Selecting the Update Customer button will take you to the following Update Customer screen:

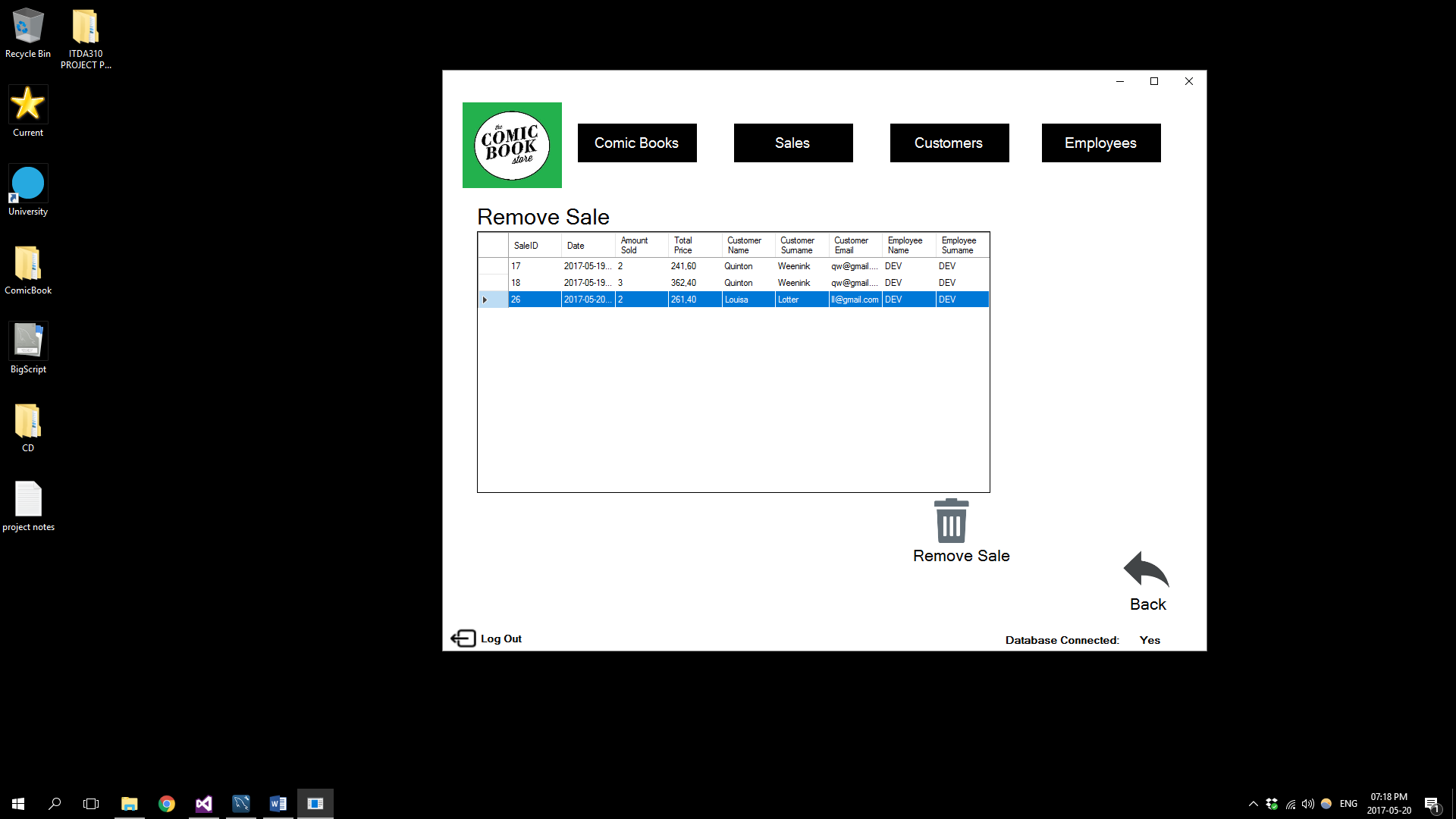


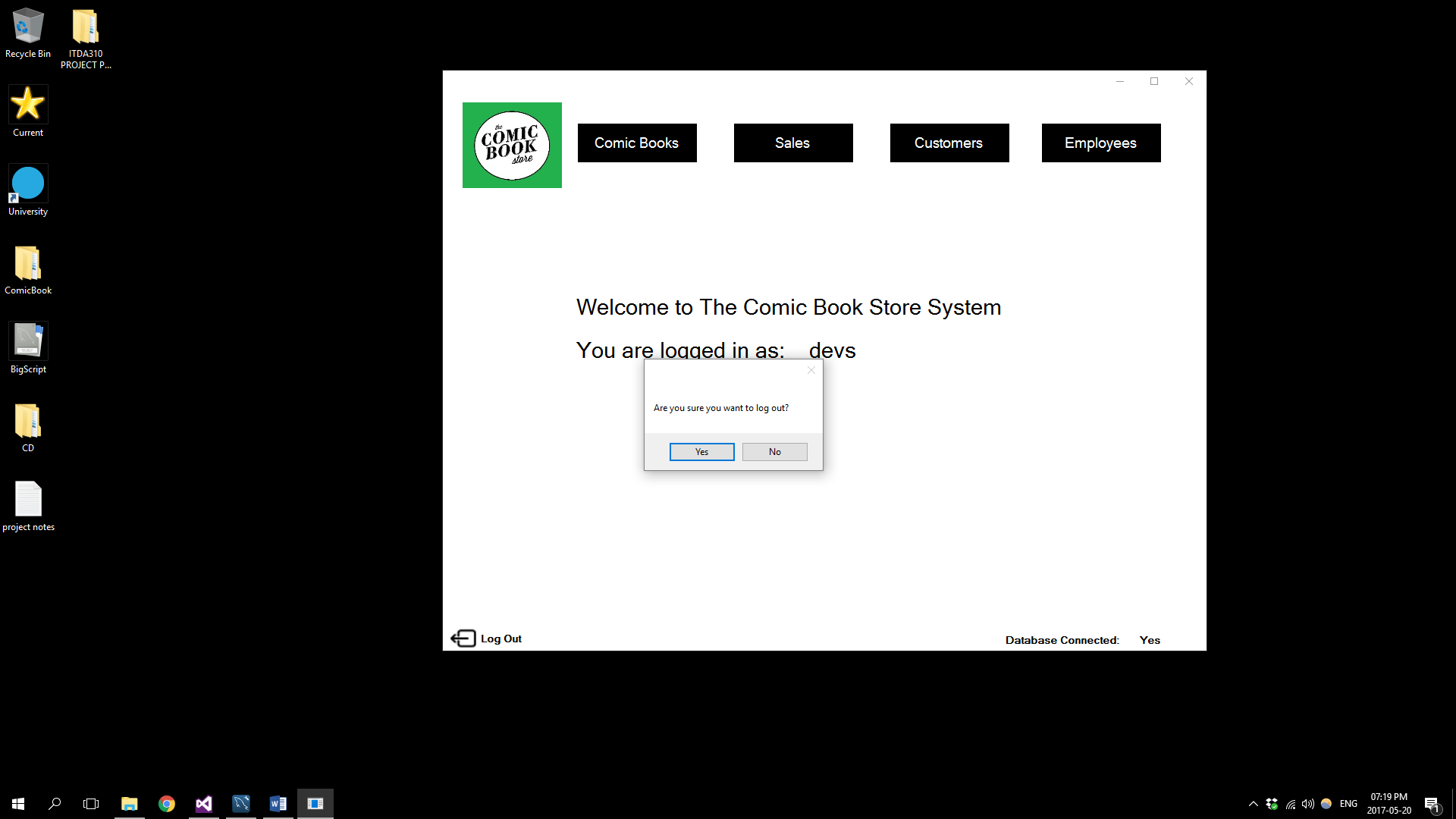
Selecting the Sales button will take you to the following main Sales screen:



Selecting the Add sale button will take you to the following Add Sale screen:



Selecting the Remove Sale button will take you to the following Remove Sale screen:

Selecting the Log out Button will display the following message, if you select yes in the message you will be logged out:

# Conclusion

During the development of this project and the system it entails a database was developed and documented as well as a application which is supported by the database.

This document concludes all the necessary documentation to support the process of developing the system.

# References

Rouse, M. 2005. DDBMS (distributed database management system). [Online]. Available at: <http://searchsqlserver.techtarget.com/definition/DDBMS>. Accessed on: 12 May 2017.

Distributed Database Management System. 2017. [Online]. Available at: <https://cs.uwaterloo.ca/~tozsu/courses/cs856/F02/lecture-1-ho.pdf>. Accessed on: 12 May 2017.

MySQL DBMS.2017. [Online]. Available at: <http://dbms.ca/mysql/>. Accessed on: 12 May 2017.

Jewett,T. 2006. Database design with UML and SQL. [Online]. Available at: <http://www.tomjewett.com/dbdesign/dbdesign.php?page=ddldml.php>. Accessed on: 13 May 2017.