***GROUP 4***

**ASSIGNMENTNT**

**DBI202 – Bookstore management system**

# TABLE OF CONTENTS

## Problem statement …………………………………………………………………………………………….

### Describe the problem ……………………………………………………………………………………………………….

### System Business Description …………………………………………………………………………………………………..

## entity – er …………………………………………………………………………………………

### List of data constraints …………………………………………………………………………………………..

### ER Diagram

### …………………………………………………………………………………………….

## data dictionary ………………………………………………………………………………………………………….

### Database and table ……………………………………………………………………………………….………………….

### full diagram ……………………………………………………………………………………….………………….

### create trigger ……………………………………………………………………………………….………………….

## SQL COMMAND

### sell report ………………………………………………………………………………………………………

### number of book sold report ……………………………………………………………………………………………………

### number of buyer report

### …………………………………………………………………………………

### book with highest sale volume

### ……………………………………………………………

### top 5 buyer report

### ………………………………………………………………….

# Problem statement:

## Describe the problem:

In the present era, purchasing books directly from physical stores poses several challenges. Firstly, the limited operating hours of these stores reduce the number of potential buyers. Additionally, the availability of a specific book may not be guaranteed, forcing buyers to search elsewhere, consuming more time and effort. Moreover, the location of the store can pose difficulties for those without convenient access. For bookstores with limited space, the inability to display a wide range of books may restrict choices, potentially depriving potential buyers of finding the desired book.

This project focuses on addressing the management challenges of traditional bookstores by developing an efficient e-commerce system. Buying books online offers numerous advantages, such as saving time, easy search and purchase from anywhere at any time. Furthermore, automating transactions helps minimize errors and enhances communication with both suppliers and customers, thereby improving efficiency and reliability in bookstore management.

## System Features Description:

2.1. Book Management:

\_ Add books:

+Administrators have the right to add information about new books to the system.

+Required information includes book title, author, price, inventory quantity, and other details.

\_ Edit books:

+Administrators have the right to edit information about books.

+Update information such as book title, author, price, inventory quantity, and other details.

\_ Delete books: Administrators can delete book information from the system.

\_ Update quantity: The system will update the quantity of books remaining in stock.

2.2. Book Category Management:

\_ Create book categories:

+Administrators have the right to create new book categories.

+Each category has a name to describe its content.

\_ Manage book categories:

+Administrators have the ability to add or delete books from categories.

+Users can search for books based on categories for easy browsing and selection.

2.3. Customer Management:

\_ Manage personal information:

+The system will update basic customer information when customers fill out information in the purchase form.

+For convenience in contacting to confirm orders and identify returning customers.

\_ Registration:

+Customers can register an account on the website.

+Required information includes username, password, first and last name, email, address, and phone number.

\_Login: Users can log in with their username and password.

2.4. Order Processing:

\_Receive orders:

+The system receives orders from users after they have selected books and proceeded to payment.

+Record information about the order, the products ordered, and customer contact information.

2.5. Order Status Management:

\_ Update order status:

+Administrators can update the status of orders from "Processing" to "Shipped" and "Received."

+The system automatically updates the status based on the order processing.

2.6. Payment:

\_ Payment processing:

+The system processes payments for orders using online payment methods such as credit cards, e-wallets, or direct payments.

+Record payment information and update the payment status of orders.

2.7. Statistical Reporting:

\_ Export daily, monthly, yearly reports:

+Revenue report.

+Report on the number of books sold.

+Report on the number of customers.

+Books with the highest sales.

+Top purchasing customers.

# II. entity – er:

1. List of data constraints:Base on the problem description and management objectives, we can present several entities and attributes of the entity as follow:

\_ tblBook:

+ Primary Key: BookID

+ Price >= 0

+ Quantity >= 0

\_ tblCategory:

+ Primary Key: CategoryID

+ Category must be unique

\_ tblBookCategoryMapping:

+ Primary Key: CMappingID

+ Foreign Keys:

. BookID references tblBook(BookID)

. CategoryID references tblCategory(CategoryID)

\_ tblRestock:

+ Primary Key: RestockID

+ Quantity >= 0

+ BookID references tblBook(BookID)

\_ tblUser:

+ Primary Key: UserID

+ Username must be unique

+ Gmail must be unique

+ Phone must be unique

\_ tblOrder:

+ Primary Key: OrderID

+ UserID references tblUser(UserID)

\_ tblOrderDetail:

+ Primary Key: OrderDetailID

+ Quantity >= 0

+ Foreign Keys:

. OrderID references tblOrder(OrderID)

. BookID references tblBook(BookID)

\_ tblOrderStatus:

+ Primary Key: StatusID

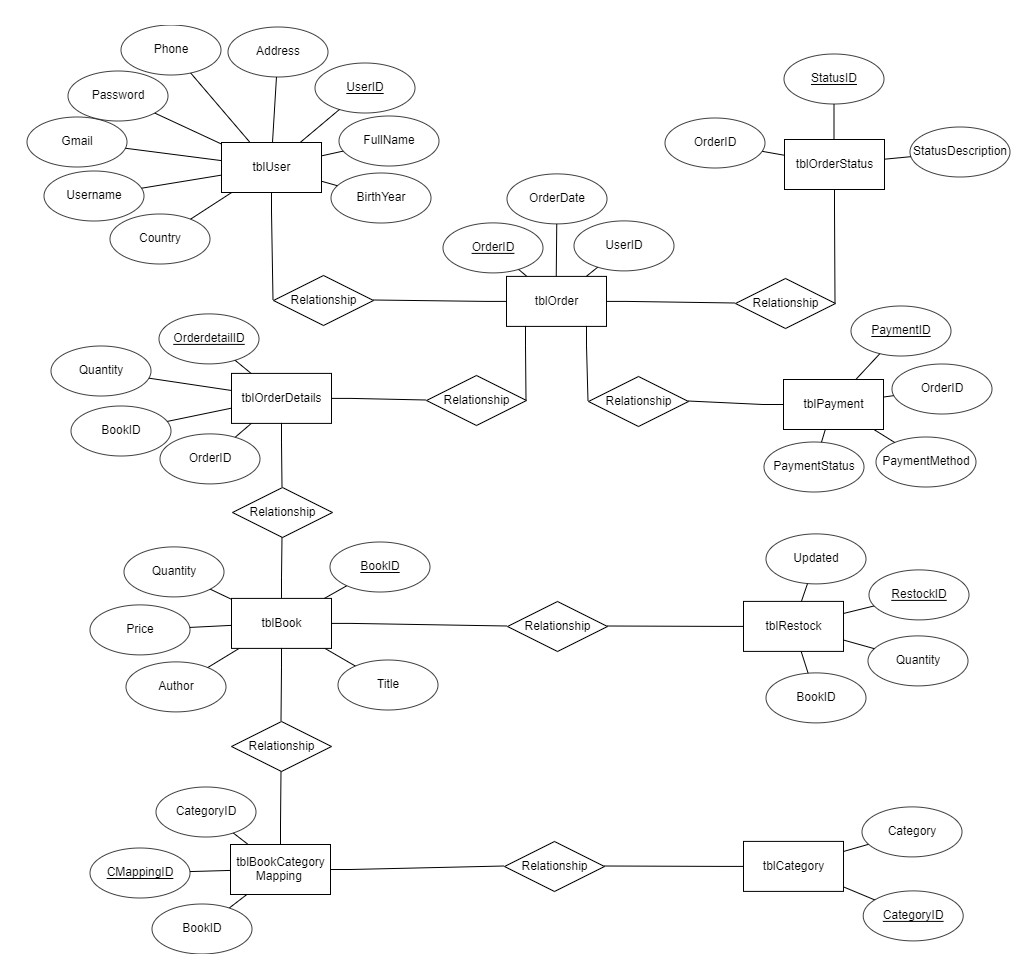
+ Foreign Key: OrderID references tblOrder(OrderID)

\_ tblPayment:

+ Primary Key: PaymentID

+ Foreign Key: OrderID references tblOrder(OrderID)

## 2. ER Diagram:



# data dictionary

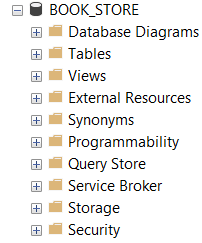
Just for example on some tables (other table are similar, you have to define all the tables in your database). Note: to run the query you have to define the table 1 first then go to the side tables much

## 1. database and table

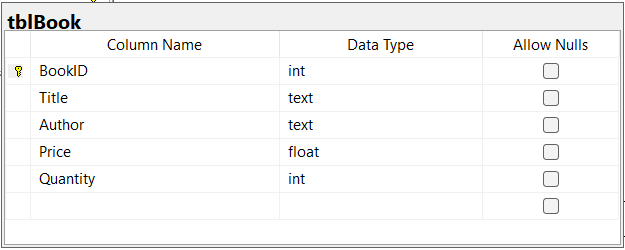
### cREATE DATABASE **PROJECTDBI202**

--create database

CREATE DATABASE BOOK\_STORE



### **Create table book**

******

***Code:***

--create table Book

CREATE TABLE [tblBook] (

[BookID] INT PRIMARY KEY,

[Title] TEXT NOT NULL,

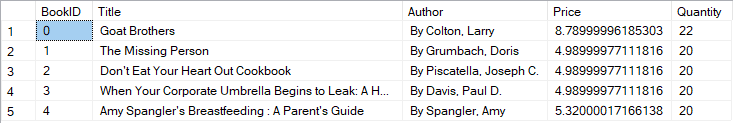
[Author] TEXT,

[Price] FLOAT NOT NULL CHECK (Price > 0),

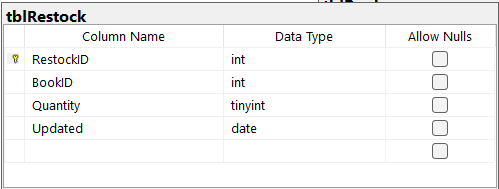
[Quantity] INT NOT NULL CHECK (Quantity > 0)

);

***Example:***



### **Create table REstock**



***Code:***

--create table restock

CREATE TABLE [tblRestock] (

[RestockID] INT PRIMARY KEY IDENTITY,

[BookID] INT,

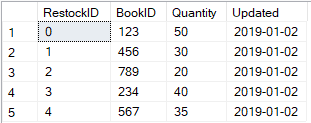
[Quantity] TINYINT NOT NULL CHECK (Quantity > 0),

[Updated] DATE NOT NULL,

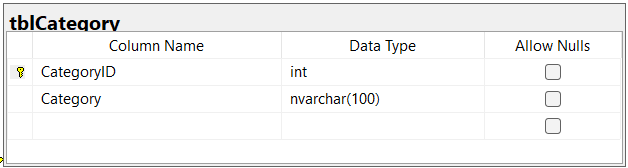
FOREIGN KEY (BookID) REFERENCES [tblBook](BookID)

);

***Example:***



### **Create table Category**



***Code:***

--create table category

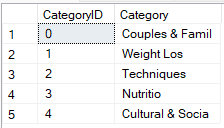
CREATE TABLE [tblCategory] (

[CategoryID] INT PRIMARY KEY IDENTITY,

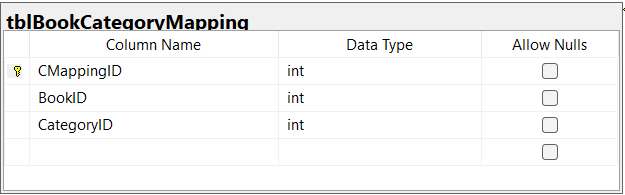
[Category] NVARCHAR(100) UNIQUE

);

***Example:***



### Create TABLE **BOOK category mapping**



***Code:***

--create table Book Category Mapping

CREATE TABLE [tblBookCategoryMapping] (

[CMappingID] INT PRIMARY KEY IDENTITY,

[BookID] INT ,

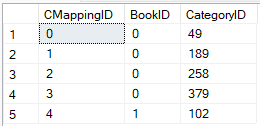
[CategoryID] INT,

FOREIGN KEY ([BookID]) REFERENCES [tblBook]([BookID]),

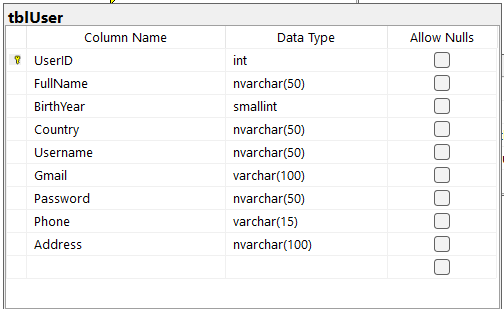
FOREIGN KEY ([CategoryID]) REFERENCES [tblCategory]([CategoryID])

);

***Example:***



### create table user



***Code:***

--create table user

CREATE TABLE [tblUser] (

[UserID] INT PRIMARY KEY IDENTITY,

[FullName] NVARCHAR(50) NOT NULL,

[BirthYear] SMALLINT,

[Country] NVARCHAR(50),

[Username] NVARCHAR(50) UNIQUE,

[Gmail] VARCHAR(100) NOT NULL UNIQUE,

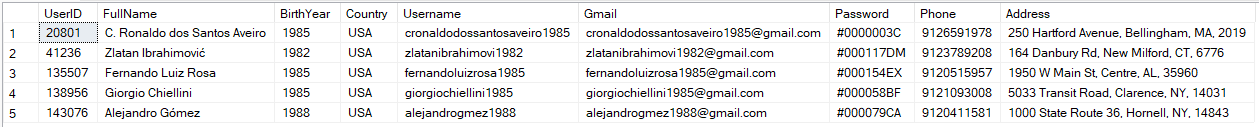
[Password] NVARCHAR(50) ,

[Phone] VARCHAR(15) NOT NULL UNIQUE,

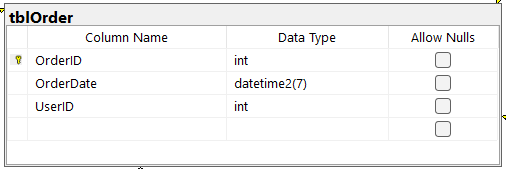
[Address] NVARCHAR(100) NOT NULL

);

***Example:***



### create table **order**



***Code:***

--create table order

CREATE TABLE [tblOrder] (

[OrderID] INT PRIMARY KEY IDENTITY,

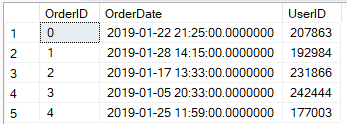
[OrderDate] DATETIME2(7)NOT NULL,

[UserID] INT ,

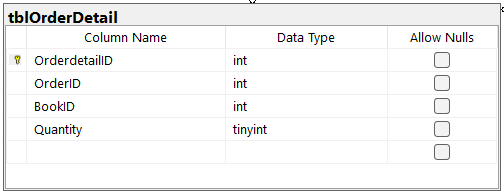
FOREIGN KEY ([UserID]) REFERENCES [tblUser]([UserID])

);

***Example:***



### create table **order detail**



***Code:***

--create table order detail

CREATE TABLE [tblOrderDetail] (

[OrderdetailId] INT PRIMARY KEY IDENTITY,

[OrderID] INT,

[BookID] INT ,

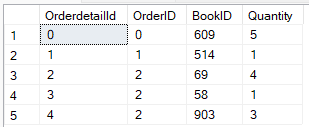
[Quantity] TINYINT NOT NULL CHECK (Quantity > 0),

FOREIGN KEY ([OrderID]) REFERENCES [tblOrder]([OrderID]),

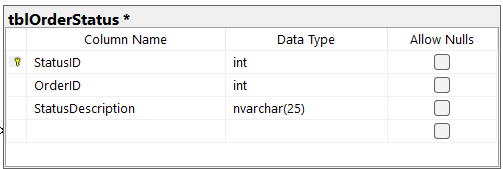
FOREIGN KEY ([BookID]) REFERENCES [tblBook](BookID)

);

***Example:***



### Create table **order status**



***Code:***

--create table order status

CREATE TABLE [tblOrderStatus](

[StatusID] INT PRIMARY KEY IDENTITY,

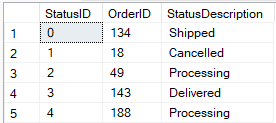
[OrderID] INT ,

StatusDescription NVARCHAR(25) CHECK (StatusDescription IN ('Cancelled', 'Delivered', 'Processing')),

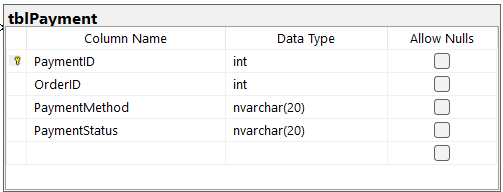
FOREIGN KEY ([OrderID]) REFERENCES [tblOrder]([OrderID])

);

***Example:***



### create table **payment**



***Code:***

--create table payment

CREATE TABLE [tblPayment] (

[PaymentID] INT PRIMARY KEY,

[OrderID] INT,

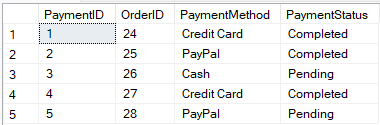
[PaymentMethod] nvarchar(20) ,

[PaymentStatus] NVARCHAR(20) CHECK (PaymentStatus IN ('Completed', 'Pending')),

FOREIGN KEY ([OrderID]) REFERENCES [tblOrder]([OrderID])

);

***Example:***



|  |  |
| --- | --- |
| full diagram |  |

## Create Trigger

### Create Trigger update book quantity through restock

-- Create trigger update book quantity

CREATE TRIGGER trgUpdateBookQuantity

ON tblRestock

AFTER INSERT

AS

BEGIN

UPDATE b

SET b.Quantity = b.Quantity + i.Quantity

FROM tblBook b

INNER JOIN tblRestock r ON b.BookID = r.BookID;

END;

### Create Trigger update book quantity through Order Status

-- Create trigger update book quantity through order status

CREATE TRIGGER trgUpdateBookQuantityOrder

ON tblOrderStatus

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF UPDATE(StatusDescription)

BEGIN

UPDATE b

SET b.Quantity = b.Quantity - od.Quantity

FROM tblBook AS b

INNER JOIN tblOrderDetail od ON b.BookID = od.BookID

INNER JOIN inserted i ON i.OrderID = od.OrderID

INNER JOIN deleted d ON i.OrderID = d.OrderID

WHERE i.StatusDescription = 'Delivered' AND d.StatusDescription != 'Delivered';

END

END;

### Create Trigger update Status description through payment status

-- Create trigger update status description through payment status

CREATE TRIGGER trgUpdateOrderStatus

ON tblPayment

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

UPDATE tblOrderStatus

SET StatusDescription = 'Processing'

WHERE OrderID = (SELECT i.OrderID FROM inserted i WHERE i.PaymentStatus = 'Completed');

END;

# sql command

## Sale report

***Code:***

***Result:***

## Number of books sold report

***Code:***

***Result:***

## number of buyer report

***Code:***

***Result:***

## Book with the highest sales volume

***Code:***

***Result:***

## top 5 buyer report

***Code:***

***Result:***