

DDL

```
create table Authors
(
    AuthorID int auto_increment
        primary key,
    Name      varchar(255) not null
);

create table Publishers
(
    PublisherID int auto_increment
        primary key,
    Name        varchar(255) not null,
    Address     text         null,
    Phone       varchar(50)  null
);

create table Books
(
    BookID          int auto_increment
        primary key,
    ISBN            varchar(20)   null,
    Title           varchar(255)  not null,
    PublicationYear int          null,
    SellingPrice    decimal(10, 2) null,
    Category        varchar(100)  null,
    NumberOfBooks   int          null,
    MinimumQuantity int          null,
    PublisherID    int          not null,
    constraint ISBN
        unique (ISBN),
    constraint Books_ibfk_1
        foreign key (PublisherID) references Publishers (PublisherID)
);
```

```
create table BookAuthors
(
    BookID    int not null,
    AuthorID int not null,
    primary key (BookID, AuthorID),
    constraint BookAuthors_ibfk_1
        foreign key (BookID) references Books (BookID),
    constraint BookAuthors_ibfk_2
        foreign key (AuthorID) references Authors (AuthorID)
);

create index AuthorID
    on BookAuthors (AuthorID);

create index PublisherID
    on Books (PublisherID);

create table PublisherOrders
(
    PublisherOrderID int auto_increment
        primary key,
    Quantity      int      null,
    Status        varchar(50) null,
    BookID        int      not null,
    constraint PublisherOrders_ibfk_1
        foreign key (BookID) references Books (BookID)
);

create index BookID
    on PublisherOrders (BookID);

create table Users
(
    UserID        int auto_increment
        primary key,
    Username     varchar(255)      not null,
    Password     varchar(255)      not null,
    FirstName    varchar(100)       null,
    LastName     varchar(100)       null,
    Email        varchar(255)      not null,
    Phone        varchar(50)       null,
    ShippingAddress text           null,
    Role         enum ('ADMIN', 'CUSTOMER') not null,
    email_verified tinyint(1) default 0      null,
    Enabled      tinyint(1) default 0      null,
    created_at   datetime   default CURRENT_TIMESTAMP not null,
    constraint uq_users_email
        unique (Email)
);
```

```
create table BillingInfos
(
    BillingInfoID int auto_increment
        primary key,
    CardNumber      varchar(50) not null,
    ExpirationDate date        null,
    BillingAddress text        null,
    UserID          int         not null,
    CardHolderName varchar(50) null,
    constraint BillingInfos_ibfk_1
        foreign key (UserID) references Users (UserID)
);

create index UserID
    on BillingInfos (UserID);

create table Carts
(
    cart_id int auto_increment
        primary key,
    user_id int null,
    constraint Carts_ibfk_1
        foreign key (user_id) references Users (UserID)
);

create table CartItems
(
    cart_id   int      not null,
    book_id   int      not null,
    quantity  int default 1 null,
    primary key (cart_id, book_id),
    constraint CartItems_ibfk_1
        foreign key (cart_id) references Carts (cart_id)
            on delete cascade,
    constraint CartItems_ibfk_2
        foreign key (book_id) references Books (BookID)
            on delete cascade
);

create index user_id
    on Carts (user_id);
```

```
create table CustomerOrders
(
    CustomerOrderID int auto_increment
        primary key,
    OrderDate      datetime      default CURRENT_TIMESTAMP null,
    Status         varchar(50)           null,
    UserID          int             not null,
    TotalPrice     decimal(10, 2) default 0.00       not null,
    constraint CustomerOrders_ibfk_1
        foreign key (UserID) references Users (UserID)
);

create table CustomerOrderItems
(
    CustomerOrderID int                  not null,
    BookID          int                  not null,
    Quantity        int                  null,
    Price           decimal(10, 2)        null,
    TotalPrice      decimal(10, 2) default 0.00 not null,
    primary key (CustomerOrderID, BookID),
    constraint CustomerOrderItems_ibfk_1
        foreign key (CustomerOrderID) references CustomerOrders (CustomerOrderID),
    constraint CustomerOrderItems_ibfk_2
        foreign key (BookID) references Books (BookID)
);

create index BookID
    on CustomerOrderItems (BookID);

create index UserID
    on CustomerOrders (UserID);

create table EmailVerificationTokens
(
    TokenID      int auto_increment
        primary key,
    Token        varchar(255) not null,
    UserID        int          not null,
    ExpiryDate   datetime    not null,
    constraint Token
        unique (Token),
    constraint fk_email_verification_user
        foreign key (UserID) references Users (UserID)
            on delete cascade
);

```

```
create table ForgetPasswordTokens
(
    email      varchar(255) null,
    expiryDate datetime     null,
    OTP        varchar(255) null,
    constraint ChangePasswordTokens_pk
        unique (email),
    constraint ChangePasswordTokens_Users_Email_fk
        foreign key (email) references Users (Email)
            on update cascade on delete cascade
);

create table RefreshTokens
(
    RefreshTokenID int auto_increment
        primary key,
    Token          varchar(255) not null,
    UserID         int          not null,
    DeviceID      varchar(255) null,
    UserAgent     text         null,
    ExpiryDate    datetime     not null,
    constraint Token
        unique (Token),
    constraint fk_refresh_user
        foreign key (UserID) references Users (UserID)
            on delete cascade
);
```

Triggers

```
DELIMITER $$

CREATE TRIGGER after_order_confirm
AFTER UPDATE ON PublisherOrders
FOR EACH ROW
BEGIN
    IF NEW.Status = 'COMPLETED' AND OLD.Status <> 'COMPLETED' THEN
        UPDATE Books
        SET NumberOfBooks = NumberOfBooks + NEW.Quantity
        WHERE BookID = NEW.BookID;
    END IF;
END$$

CREATE TRIGGER after_book_update
AFTER UPDATE ON Books
FOR EACH ROW
BEGIN
    DECLARE order_quantity INT DEFAULT 10;
    IF OLD.NumberOfBooks >= OLD.MinimumQuantity
        AND NEW.NumberOfBooks < NEW.MinimumQuantity THEN
        INSERT INTO PublisherOrders (BookID, Quantity, Status)
        VALUES (NEW.BookID, order_quantity, 'PENDING');
    END IF;
END$$

CREATE PROCEDURE CompletePublisherOrder(IN p_order_id INT)
BEGIN
    UPDATE PublisherOrders
    SET Status = 'COMPLETED'
    WHERE PublisherOrderID = p_order_id;
END$$

CREATE PROCEDURE UpdateBookDetails(
    IN p_bookId INT,
    IN p_title VARCHAR(255),
    IN p_year INT,
    IN p_price DECIMAL(10,2),
    IN p_category VARCHAR(100)
)
BEGIN
    UPDATE Books
    SET Title = p_title,
        PublicationYear = p_year,
        SellingPrice = p_price,
        Category = p_category
    WHERE BookID = p_bookId;
END$$

DELIMITER ;
```

