Create an order REST API using .NET 6.0 and c#. Use Swagger for documentation and ease of testing. Use Entity Framework Core to access the database. For the database, you can use the attached sqlite database, “OrderApiDatabase.db”. The sqlite database already has the schema created along with test data. If you prefer MS SQL Server or MS SQL Server Express, you can use the attached scripts, “OrderApiScripts.sql”, to create the schema.

The API should have the following endpoints:

- A GET endpoint for retrieving a customer.
  - Parameters: customer ID
  - Response model:

```
{
  "customerId": 0,
  "name": "string",
  "balance": 0
}
```

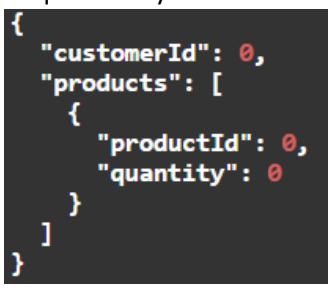
- A GET endpoint for retrieving a product.
  - Parameters: product ID
    Response model:

```
{
  "productId": 0,
  "name": "string",
  "price": 0,
  "quantity": 0
}
```

- A GET endpoint for retrieving orders.
  - Parameters: customer ID, start date, end date
  - Response model:

```
[
  {
    "orderId": 0,
    "customerId": 0,
    "orderDate": "2023-03-06T20:38:11.468Z",
    "total": 0,
    "lineItems": [
      {
        "productId": 0,
        "productName": "string",
        "quantityPurchased": 0,
        "totalCost": 0
      }
    ]
  }
]
```

- A POST endpoint for creating an order. The creation should only succeed if there are sufficient product quantities, and the customer has a sufficient balance to pay for all the products.
    - Request body:



```
{
  "customerId": 0,
  "products": [
    {
      "productId": 0,
      "quantity": 0
    }
  ]
}
```

    - Response model:

```
{
  "orderId": 0,
  "customerId": 0,
  "orderDate": "2023-03-06T20:43:23.675Z",
  "total": 0,
  "lineItems": [
    {
      "productId": 0,
      "productName": "string",
      "quantityPurchased": 0,
      "totalCost": 0
    }
  ]
}
```

*For this exercise you do not need to worry about authentication or authorization. You also do not need to worry about database concurrency issues.