BÀI THỰC HÀNH 3

I. Cài đặt cơ sở dữ liệu

Cơ sở dữ liệu **CardPaymentDB** được sử dụng để quản lý hoạt động bán thẻ trực tuyến. Tạo cơ sở dữ liệu và thiết kế các bảng dữ liệu theo mô tả dưới đây

1. Bảng **Customer:** Lưu thông tin về khách hàng (tức là người mua thẻ)

Field	DataType	Ghi chú
CustomerId	Int	PK, Identity
CustomerName	Nvarchar(100)	NOT NULL
RegisterTime	Datetime	NOT NULL DEFAULT GETDATE()
Email	Nvarchar(100)	NOT NULL, Unique Key
Password	Nvarchar(100)	NOT NULL
MobiNumber	Nvarchar(50)	NOT NULL
IsLocked	Bit	NOT NULL, DEFAULT (0)

2. Bảng **CardType**: Lưu danh mục các loại thẻ mà hệ thống có bán

Field	DataType	Ghi chú
CardTypeId	Int	PK
CardTypeName	Nvarchar(50)	NOT NULL
Descriptions	Nvarchar(255)	NOT NULL

3. Bảng **Invoice**: Lưu trữ danh sách các đơn hàng của khách hàng. Được lập khi khách hàng mua thẻ

Field	DataType	Ghi chú
InvoiceId	Bigint	PK, Identity
CustomerId	Int	NOT NULL
CreatedTime	Datetime	NOT NULL DEFAULT GETDATE()
InvoiceStatus	Int	NOT NULL DEFAULT (0)
FinishedTime	Datetime	NULL DEFAULT (NULL)

Trường **InvoiceStatus**: Cho biết trạng thái của đơn hàng, theo qui ước:

0 : Đơn hàng vừa mới được khởi tại và chưa hoàn tất

1 : Đơn hàng đã hoàn tất giao dịch thành công

< 0 : Giá trị nhỏ hơn 0 được sử dụng cho trạng thái đơn hàng trong trường hợp đơn hàng không giao dịch thành công (giá trị của trạng thái này cho biết lý do vì sao giao dịch không thành công)

4. Bảng **CardStore**: Lưu trữ danh sách các mã thẻ hiện có trong hệ thống (thẻ đã bán, chưa bán, thẻ bị hủy,…)

Field	DataType	Ghi chú
CardId	Bigint	PK, Identity
CardTypeId	Int	NOT NULL FOREIGN KEY
Serial	Nvarchar(50)	NOT NULL
PinNumber	Nvarchar(50)	NOT NULL
Amount	Money	NOT NULL
ExpiredTime	Datetime	NOT NULL
CardStatus	Int	NOT NULL
InvoiceId	Bigint	NULL DEFAULT (NULL) FOREIGN KEY

Unique Key (CardTypeId, Serial)

Trường **CardStatus**: cho biết trạng thái của 1 thẻ, theo qui ước:

1 : Thẻ chưa được bán (tức là có thể sử dụng để bán cho khách hàng)
0 : Thẻ đã bán cho khách hàng
-1 : Thẻ đã bị khóa

Sử dụng dữ liệu trong file CSV để import vào CSDL và đưa dữ liệu vào bảng CardStore.

II. Nhập dữ liệu

Nhập dữ liệu cho các bảng theo yêu cầu sau:

1. Bảng **CardType**:

CardTypeId	CardTypeName	Description
1	MOBILE	Thẻ MobiFone
2	VINA	Thẻ Vinaphone
3	VIETTEL	Thẻ Viettel

2. Bảng **Customer**: Import dữ liệu khách hàng từ file **BaiThucHanh_03 - Customer.csv**

3. Bảng **CardStore**: Import dữ liệu từ các file sau

- Thẻ MOBILE : file **BaiThucHanh_03 - MOBILE Cards.csv**
- Thẻ VINA : file **BaiThucHanh_03 - VINA Cards.csv**
- Thẻ VIETTEL : file **BaiThucHanh_03 - VIETTEL Cards.csv**

III. Cài đặt các hàm và thủ tục cho cơ sở dữ liệu

1. Hàm **fn_IsValidMobiNumber(@MobiNumber nvarchar(50)** trả về giá trị kiểu bit nhằm kiểm tra @MobiNumber có phải là số di động hợp lệ hay không. Trong đó qui định số di động được xem là hợp lệ nếu là chuỗi có độ dài từ 9 đến 12 ký tự và chỉ chứa các con số (`NOT LIKE '%[^0-9]%'`).

2. Thủ tục **proc_Customer_Register**

```
@CustomerName  nvarchar(100),
@Email nvarchar(100),
@Password nvarchar(100),
@MobiNumber nvarchar(50),
@CustomerId int OUTPUT
```

Có chức năng đăng ký mới một tài khoản khách hàng (bổ sung dữ liệu cho bảng **Customer**). Trong đó, tham số đầu ra @CustomerId trả về 1 giá trị theo qui ước:

- Nếu đăng ký tài khoản thành thành công: @CustomerId là mã của khách hàng vừa được đăng ký.
- Nếu đăng ký tài khoản không thành công: @CustomerId trả về 1 giá trị âm theo qui ước sau:
  - -1 : Nếu Email rỗng
  - -2 : Nếu Email trùng với Email của tài khoản đã tồn tại.
  - -3 : Nếu mật khẩu rỗng hoặc có độ dài nhỏ hơn 6 ký tự.
  - -4 : Nếu tên của khách hàng rỗng.
  - -5 : Nếu số di động không hợp lệ.

3. Thủ tục **proc_Customer_ChangePassword**

```
@Email nvarchar(100),
@OldPassword nvarchar(100),
@NewPassword nvarchar(100),
@Result int OUTPUT
```

thực hiện chức năng đổi mật khẩu tài khoản của khách hàng. Tham số @Result trả về kết quả theo qui ước:

- 1 : Đổi mật khẩu thành công
- 0 : Đổi mật khẩu không thành công

4. Thủ tục **proc_Customer_Authenticate**

```
@Email nvarchar(100),
@Password nvarchar(100)
```

có chức năng kiểm tra thông tin đăng nhập của khách hàng. Nếu thông tin đăng nhập là, thủ tục trả về thông tin của khách hàng bao gồm: mã khách hàng, tên khách hàng, email, di động và trạng thái của tài khoản; Ngược lại, thủ tục không trả về dữ liệu.

5. Thủ tục **proc_Customer_Update**

```
@CustomerId int,
@CustomerName nvarchar(100),
@Email nvarchar(100),
@MobiNumber nvarchar(50),
```

@Result int OUTPUT

thực hiện chức năng cập nhật thông tin tài khoản khách hàng có mã khách hàng là @CustomerId. Tham số đầu ra @Result cho biết kết quả cập nhật dữ liệu theo qui ước:

- 1 : Nếu cập nhật thành công
- 0 : Nếu mã khách hàng không tồn tại
- -1 : Nếu Email rỗng
- -2 : Nếu Email trùng với Email của tài khoản khác.
- -4 : Nếu tên của khách hàng rỗng.
- -5 : Nếu số di động không hợp lệ.

6. Thủ tục **proc_Customer_Select**

```
@Page int = 0,
@PageSize int = 0,
@SearchValue nvarchar(255) = N'',
@RowCount int OUTPUT,
@PageCount int OUTPUT
```

Có chức năng tìm kiếm khách hàng theo tên, địa chỉ email hoặc số di động (tìm kiếm tương đối). Dữ liệu trả về được hiển thị dưới dạng được phân trang, Trong đó, các tham số của thủ tục được mô tả như sau:

@Page	: Trang cần hiển thị dữ liệu
@PageSize	: Số dòng trên mỗi trang (nếu @PageSize = 0 thì hiển thị toàn bộ dữ liệu tìm được)
@SearchValue	: Giá trị tìm kiếm (trong trường hợp @SearchValue là chuỗi rỗng thì bỏ qua thao tác tìm kiếm mà lấy toàn bộ dữ liệu)
@RowCount	: Tham số đầu ra cho biết tổng số dòng dữ liệu tìm được
@PageCount	: Tham số đầu ra cho biết tổng số trang.

7. Thủ tục **proc_Invoice_Select**

```
@Page int = 1,
@PageSize int = 0,
@CustomerId int = 0,
@FromTime datetime = null,
@ToTime datetime = null,
@RowCount  OUTPUT,
@PageCount OUTPUT
```

Có chức năng tìm kiếm và hiển thị danh sách các hóa đơn hàng bán thẻ dưới dạng phân trang. Thông tin cần hiển thị bao gồm: mã đơn hàng, mã khách hàng, tên khách hàng, thời điểm tạo hóa đơn, trạng thái hóa đơn, thời điểm kết thúc và tổng trị giá (tiền) của đơn hàng. Trong đó các tham số của thủ tục được mô tả như sau:

@Page	: Trang cần hiển thị dữ liệu
@PageSize	: Số dòng trên mỗi trang (nếu @PageSize = 0 thì hiển thị toàn bộ dữ liệu tìm được)
@CustomerId	: Mã của khách hàng cần tìm kiếm đơn hàng. Nếu @CustomerId = 0 thì tìm kiếm theo tất cả các khách hàng.
@FromTime	: Thời điểm bắt đầu của khoảng thời gian cần tìm kiếm (tính theo

thời điểm tạo đơn hàng). Trường hợp @FromTime là NULL thì không hạn chế thời điểm bắt đầu khi tìm kiếm.

@ToTime : Thời điểm kết thúc của khoảng thời gian cần tìm kiếm (tính theo thời điểm tạo đơn hàng). Trường hợp @ToTime là NULL thì không hạn chế thời điểm kết thúc khi tìm kiếm.

@RowCount : Tham số đầu ra cho biết tổng số dòng dữ liệu tìm được

@PageCount : Tham số đầu ra cho biết tổng số trang.

8. Viết hàm **fn_GetInventories(CardTypeId int, @Amount money)** có chức năng tính số lượng thẻ đang còn tồn trong kho (tức là thẻ chưa bán) theo loại thẻ và mệnh giá.

9. Viết hàm **fn_GetCardStatistics()** có chức năng trả về một bảng nhằm thống kê hiện trạng số lượng thẻ trong kho theo mẫu sau:

Loại thẻ	Mệnh giá	Tổng số lượng thẻ	Số lượng thẻ đã bán	Số lượng thẻ còn tồn	Số lượng thẻ bị khóa hoặc hết hạn

10. Viết hàm **fn_GetRevenueByDate**

```
(
    @CardTypeId int,
    @FromTime datetime,
    @ToTime datetime
)
```

Trả về một bảng cho biết doanh thu (số tiền bán thẻ) hàng ngày trong khoảng thời gian từ ngày @FromTime đến ngày @ToTime theo mẫu sau:

Ngày	Tổng doanh thu

Kết quả trả về của hàm phải đầy đủ tất cả các ngày trong khoảng thời gian cần thống kê. Những ngày không có doanh thu thì hiển thị với Tổng doanh thu là 0.

(Lưu ý: Nếu tham số @CardTypeId = 0 thì thống kê doanh thu của tất cả loại thẻ)

11. Hiện trong kho hàng chỉ lưu trữ để bán các thẻ có mệnh giá là 50k, 100k, 200k và 500k.

Viết hàm **fn_GetRevenueByDateAndAmount**

```
(
    @CardTypeId int,
    @FromTime datetime,
    @ToTime datetime
)
```

Trả về một bảng cho biết doanh thu (số tiền bán thẻ) hàng ngày của từng mệnh giá thẻ trong khoảng thời gian từ ngày @FromTime đến ngày @ToTime theo mẫu sau:

Ngày	50k	100k	200k	500k	Tổng doanh thu

Kết quả trả về của hàm phải đầy đủ tất cả các ngày trong khoảng thời gian cần thống kê. Những ngày không có doanh thu thì hiển thị với doanh thu là 0.

(Lưu ý: Nếu tham số @CardTypeId = 0 thì thống kê doanh thu của tất cả loại thẻ)

12. Qui trình mua/bán thẻ qua hệ thống về cơ bản thông qua 3 bước như sau:

Mỗi một giao dịch mua thẻ có thể mua nhiều loại thẻ với các mệnh giá khác nhau (Ví dụ: Mua 10 thẻ Viettel mệnh giá 100000 đồng, 5 thẻ MOBILE với mệnh giá 200000 đồng, 20 thẻ MOBILE với mệnh giá 50,000 đồng). Qui trình xử lý khi có giao dịch mua thẻ phát sinh được thực hiện như sau:

- **Bước 1:** Khi cần mua thẻ, khách hàng sẽ tạo một yêu cầu mua hàng (tức là bắt đầu một giao dịch mua thẻ). Mỗi một giao dịch mua thẻ có thể mua nhiều loại thẻ với các mệnh giá khác nhau. Hệ thống sẽ căn cứ vào nhu cầu của khách hàng để khởi tạo đơn hàng cho khách hàng (nếu hợp lệ). Đơn hàng khi khởi tạo sẽ có InvoiceStatus = 0 (tức là "Đơn hàng đang tạm lập nhưng chưa hoàn tất") đồng thời phải tiến hành tạm giữ thẻ cho đơn hàng (thẻ tạm giữ là thẻ chưa chính thức được bán, nhưng không được sử dụng để bán cho đơn hàng khác).
- **Bước 2:** Sau khi khởi tạo đơn hàng xong, hệ thống sẽ chuyển khách hàng sang cổng thanh toán của ngân hàng và khách hàng sẽ tiến hành thanh toán tiền thông qua cổng thanh toán của ngân hàng.
- **Bước 3:** Sau khi khách hàng thanh toán xong, ngân hàng sẽ báo kết quả về cho hệ thống thông qua mã trạng thái (ResponseCode) theo qui định như sau:

ResponseCode	**Ý nghĩa**
1	Khách hàng thanh toán thành công
-1	Giao dịch không thành công do khách hàng hủy bỏ giao dịch
-2	Giao dịch không thành công do khách hàng không đủ tiền trong tài khoản
-3	Giao dịch không thành công do tài khoản ngân hàng của khách hàng bị khóa
-4	Giao dịch không thành công do vượt quá hạn mức chi tiêu trong ngày
-5	Giao dịch không thành công do dịch vụ thanh toán của ngân hàng đang bảo trì
-99	Giao dịch không thành công (lỗi không rõ lý do)

Căn cứ vào kết quả trả về từ ngân hàng, hệ thống sẽ tiến hành ghi nhận kết thúc đối với đơn hàng:

- Nếu thanh toán thành công: Ghi nhận trạng thái giao dịch thành công cho đơn hàng, đồng thời chuyển các thẻ đã tạm giữ sang trạng thái thẻ đã được bán.
- Nếu thanh toán không thành công: Ghi nhận trạng thái giao dịch không thành công cho đơn hàng, đồng thời trả các thẻ đã tạm giữ về trạng thái để có thể sử dụng để bán cho các đơn hàng khác.

Yêu cầu:

Tạo kiểu dữ liệu dạng bảng có tên là TypeInvoiceDetails, sử dụng đoạn lệnh dưới đây:

```
create type TypeInvoiceDetails as table
(
	CardTypeId int,
	Amount money,
	Quantity int,
	primary key (CardTypeId, Amount)
)
```

Viết thủ tục: **proc_Invoice_Init**

```
@CustomerId int,
@InvoiceDetails TypeInvoiceDetails readonly,
@InvoiceId bigint output
```

Có chức năng khởi tạo một đơn hàng bán thẻ. Trong đó, tham số đầu ra @InvoiceId có giá trị trả về theo qui ước sau:

- Nếu lập đơn hàng thành công, @InvoiceId là giá trị cho biết mã đơn hàng được tạo.
- Nếu đơn hàng lập không thành công, @InvoiceId là giá trị âm cho biết lý do không lập được đơn hàng, theo tắc ước:
  - -101 : Khách hàng không tồn tại
  - -102 : Tài khoản khách hàng bị khóa
  - -103 : Loại thẻ cần mua không tồn tại
  - -104 : Số lượng thẻ cần mua không hợp lệ
  - -105 : Số lượng thẻ không đủ để bán

Viết thủ tục **proc_Invoice_Finish**

```
@InvoiceId bigint,
@ResponseCode int
```

Có chức năng xử lý kết quả trả về từ phía ngân hàng để ghi nhận kết thúc đơn hàng. Trong trường hợp nếu kết quả trả về từ ngân hàng cho biết khách hàng đã thanh toán thành công thì thủ tục trả về danh sách các thẻ được xuất bán cho khách hàng với các thông tin: tên loại thẻ, Serial, số PIN, mệnh giá và thời điểm hết hạn thẻ.