به نام خدا

گزارش آز پایگاه داده	هفته ی ششم – بخش دوم
سید پوریا احمدی	۹۷۲۳۰۰۲
محمدمهدی هجرتی	۹۷۲۳۱۰۰

برای راستی آزمایی جواب های نشان داده شده داده های زیر insert شده را قرار داده شده است.

```
INSERT INTO BRANCH (branch_name, branch_city, assets)
VALUES ('one', 'Tehran', '12000'),
       ('two', 'Tehran', '13000'),
       ('three', 'mashhad', '15000');

INSERT INTO EMPLOYEE (employee_id, employee_name, dependent_name, telephone_number, starting_date, employment_length)
VALUES (1, 'Ali', 'Ali2', '123456', '2022-03-01', 2),
         (2, 'Hasan', 'Hasan2', '234567', '2022-05-08', 1),
         (3, 'Abbas', 'Abbas2', '345678', '2022-02-28', 4);

INSERT INTO CUSTOMER (customer_id, customer_name, customer_street, customer_city, employee_id)
VALUES (1, 'Ali', 'S12', 'Tehran', '1'),
         (2, 'Hasan', 'S32', 'Tehran', '1'),
         (3, 'Abbas', 'S32', 'Tehran', '2'),
         (4, 'Ali2', 'S32', 'Tehran', '3');

INSERT INTO LOAN (loan_number, amount, branch_name, customer_id)
VALUES (1, 100, 'one', 2),
         (2, 250, 'three', 1);

INSERT INTO PAYMENT (payment_number, payment_date, payment_amount, loan_number)
VALUES (1, '2022-05-08', 500 , 1),
         (2, '2022-06-8', 600, 1);

INSERT INTO ACCOUNT (account_number, balance, customer_id)
VALUES (1, 500, 1),
         (2, 200, 2),
         (3, 800, 3),
         (4, 700, 4);

INSERT INTO CHECKING_ACCOUNT (account_number, overdraft_amount)
VALUES (1, 100),
         (2, 200);

INSERT INTO SAVING_ACCOUNT (account_number, interest_rate)
VALUES (3, 500),
         (4, 600);

INSERT INTO DEPOSITOR (customer_id, account_number, access_date)
VALUES (1, 1, '2022-01-02'),
         (2, 2, '2022-02-22'),
         (4, 3, '2022-03-13');
```

۲ الف)

```
alter procedure bank_customeeer @name varchar(64), @bal int output, @acc_num int output
as
begin
select @bal = balance , @acc_num = account_number from ACCOUNT inner join
CUSTOMER on CUSTOMER.customer_id = ACCOUNT.customer_id where customer_name = @name;
end


declare @bala int ;
declare @acc_number int;
exec bank_customeeer  'Ali',  @bal = @bala output ,@acc_num = @acc_number output



print (@bala)
--print (@acc_number)
```

```
%
Messages
500

Completion time: 2022-05-14T20:20:58.5497381+04:30
```

در این روال ابتدا نام را دریافت کرده و با select ، balance, acc_num موارد خواسته شده سوال را به متغیر های خروجی میدهیم و با execute کردن آن جواب را پرینت میکنیم در اینجا اسم فرد علی هست و balance حسابش ۵۰۰ است.

۲ ب)

```
alter procedure payment_info  @pay int , @branch varchar(64) output
as
begin
select @branch = branch_name  from PAYMENT inner join
LOAN on LOAN.loan_number = PAYMENT.loan_number where PAYMENT.payment_number = @pay
end


declare @b_name varchar(64);

exec payment_info 1,@branch = @b_name output

print (@b_name)
```

```
%
Messages
one

Completion time: 2022-05-14T20:24:06.8705265+04:30
```

در این procedure ابتدا payment number به عنوان ورودی گرفته شده و خروجی نیز نام شعبه است.

حال با گرفتن شماره پرداخت داخل select با inner join از آنجا که داده ها در دو جدول متفوت هستند استفاده شده و نام شعبه برگردانده می شود که با execute کردن این روال نام شعبه در b_name ذخیره شده و پرینت میکنیم

۲ ج)

```
alter procedure customerr  @c_id  int ,@c_name VARCHAR(64) output,@c_street VARCHAR(64) output,@c_city VARCHAR(64) output
as
begin
select @c_name = customer_name ,@c_street = customer_street , @c_city = customer_city from CUSTOMER  where customer_id = @c_id;
waitfor delay '00:00:10';
end
declare @cus_name varchar(64) , @cus_street varchar(64),@cus_city varchar(64);
--declare @cus_street varchar(64);
--declare @cus_city varchar(64);
exec customerr 1, @c_name = @cus_name output , @c_street = @cus_street output , @c_city = @cus_city output


print (@cus_name)
print (@cus_street)
print (@cus_city)
```

```
115 %
Messages
Ali
S12
Tehran

Completion time: 2022-05-14T20:42:18.0487971+04:30
```

در این روال آیدی مشتری را گرفته و از جدول customer اطلاعاتش را گرفته و ۱۰ ثانیه تاخیر هم قرار میدهیم و سپس با execute کردن این روال خروجی ها را گرفته و پرینت میکنیم.

۳ الف)

```
alter function accnum ()
returns int
as
begin
    declare @accountNum int, @maxInterest int;
    select @maxInterest = max(interest_rate) from SAVING_ACCOUNT;

    select @accountNum = SAVING_ACCOUNT.account_number
    from SAVING_ACCOUNT where SAVING_ACCOUNT.interest_rate = @maxInterest;
    return @accountNum
end

declare @high int;

exec @high = accnum ;
print @high;
```

```
%
Messages
4

Completion time: 2022-05-14T20:30:09.0256653+04:30
```

در این تابع در select اول بیشترین سود را با تابع max گرفته و در maxinterest میریزیم

حال از شماره حسابی که maxinterest را دارد را درaccountnum میریزیم و بر میگردانیم و پرینت میکنیم.

۳ ب)

```
alter function employeeeee (@idd int)
returns varchar(64)
as
begin
    declare @num varchar(64);
    select @num = dependent_name  from EMPLOYEE where employee_id= @idd
    return @num;
end

declare @return varchar (64);
exec @return =employeeee   @idd = 2;

print @return
```

```
%
Messages
Hasan2

Completion time: 2022-05-14T20:32:57.6389782+04:30
```

در این تابع به عنوان ورودی آیدی کارمند را داده و نام دپارتمان این شخص را که در جدول employee است را با شرط where پیدا کرده و مقدار برگردانده شده را نیز پرینت میکنیم.

۴ الف)

```sql
CREATE TABLE logTable
(
    Id INT PRIMARY KEY IDENTITY(1,1),
    ChangeDate DATETIME DEFAULT GETDATE(),
    Command NCHAR(6),
    OldPaymentNumber INT NULL,
    NewPaymentNumber INT NULL,
    OldPaymentDate Date NULL,
    NewPaymentDate Date NULL,
    OldPaymentAmount INT NULL,
    NewPaymentAmount INT NULL,
    OldLoanNumber INT NULL,
    NewLoanNumber INT NULL,
)
GO

CREATE TRIGGER paymentLogger
ON PAYMENT
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @operation CHAR(6)
        SET @operation = CASE
                WHEN EXISTS(SELECT * FROM inserted) AND EXISTS(SELECT * FROM deleted)
                    THEN 'Update'
                WHEN EXISTS(SELECT * FROM inserted)
                    THEN 'Insert'
                WHEN EXISTS(SELECT * FROM deleted)
                    THEN 'Delete'
                ELSE NULL
        END
```

```sql
            ELSE NULL
        END
    IF @operation = 'Delete'
            INSERT INTO logTable (Command, ChangeDate, OldPaymentNumber, OldPaymentDate, OldPaymentAmount, OldLoanNumber)
            SELECT @operation, GETDATE(),  d.payment_number, d.payment_date, d.loan_number, d.loan_number
            FROM deleted d

    IF @operation = 'Insert'
            INSERT INTO logTable (Command, ChangeDate, NewPaymentNumber,  NewPaymentDate, NewPaymentAmount, NewLoanNumber)
            SELECT @operation, GETDATE(), i.payment_number, i.payment_date, i.payment_amount, i.loan_number
            FROM inserted i

    IF @operation = 'Update'
            INSERT INTO logTable (Command, ChangeDate, NewPaymentNumber, OldPaymentNumber, NewPaymentDate,OldPaymentDate, NewPaymentAmount, OldPaymentAmount, NewLoanNumber, Old
            SELECT @operation, GETDATE(), i.payment_number, d.payment_number, i.payment_date, d.payment_date, i.payment_amount, d.payment_amount, i.loan_number, d.loan_number
            FROM deleted d, inserted i
END
GO


INSERT INTO PAYMENT (payment_number, payment_date, payment_amount, loan_number)
VALUES (6, '2022-01-08', 100 , 2);

select * from logTable
-- disable paymentLogger on logTable;
```

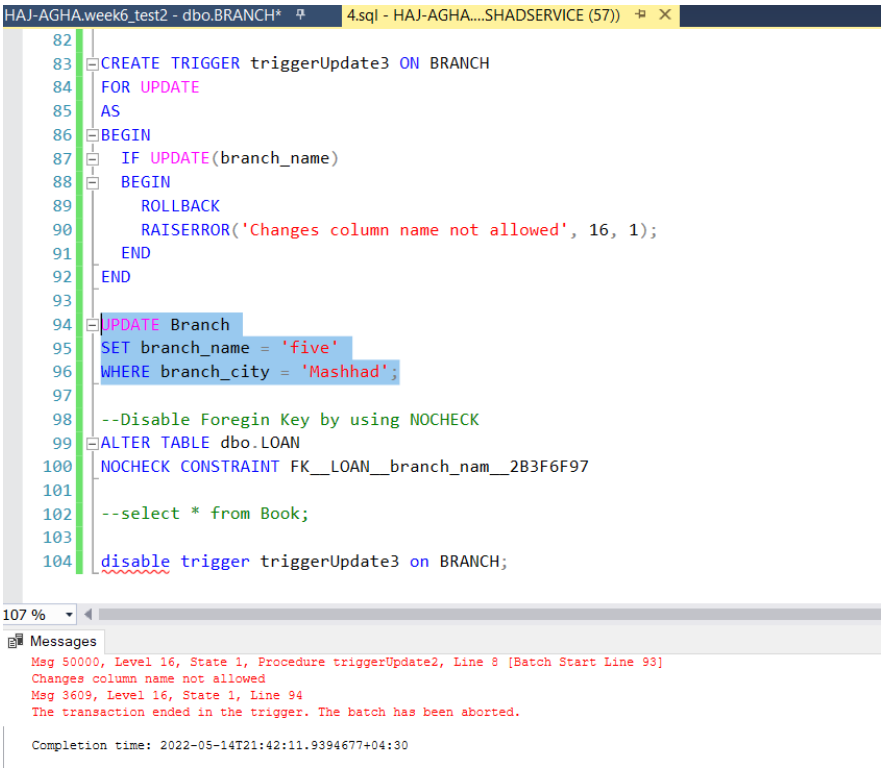
Id	ChangeDate	Command	OldPaymentNumber	NewPaymentNumber	OldPaymentDate	NewPaymentDate	OldPaymentAmount	NewPaymentAmount	OldLoanNumber	NewLoanNumber
1	2022-05-13 21:05:20.983	Insert	NULL	6	NULL	2022-01-08	NULL	100	NULL	2

در این سوال متناظر با جدول payment یک جدول برای ثبت لاگ ساخته شده است. که دارای تمام فیلدهای جدول اصلی payment به دو صورت old و new می باشد. به علاوه فیلدهایی برای نوع کامند وارد شده و تاریخ آن افزوده شده است.

در trigger ابتدا با توجه به دیتای موجود در جدول های inserted و deleted نوع کامند وارد شده را تشخیص میدهیم.

در ادامه با توجه به نوع کامند اطلاعات را وارد جدول میکنیم.

۴ ب)



Trigger نوشته شده در این سوال به ازای update شدن نوشته شده است. در خط ۸۷ می بینیم که در صورتی که فیلد branch_name بخواهد update شود این trigger اجرا می شود. که در آن این تغییر roollback شده و ارور نمایش داده می شود.