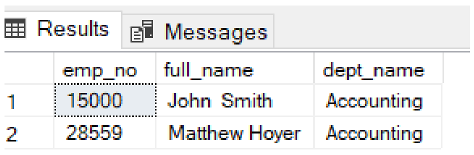
ASSIGNMENT 11

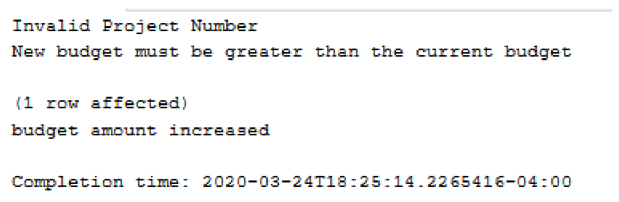
1.
   ```
CREATE PROCEDURE GetEmployeeInfo @dept_no char(4)
AS
BEGIN
 SELECT e.emp_no, e.emp_fname + ' ' + e.emp_lname as full_name,
                    d.dept_name
                    FROM employee e
                              join department d on e.dept_no = d.dept_no
                    where d.dept_no =@dept_no
END

exec GetEmployeeInfo 'd1'
```



	emp_no	full_name	dept_name
1	15000	John Smith	Accounting
2	28559	Matthew Hoyer	Accounting

2.
   ```
CREATE PROCEDURE IncreaseBudgetAmount @project_no char(4), @new_budget
float,
@message varchar(100) output
AS
BEGIN

 IF NOT EXISTS (
          SELECT 1 from dbo.project p where p.project_no = @project_no
                    )
           SET @message ='Invalid Project Number'

 ELSE IF EXISTS (
          SELECT 1 from dbo.project p where p.project_no = @project_no
                    and @new_budget> p.budget
                    )
          BEGIN
                     UPDATE dbo.project set budget = @new_budget where project_no =
@project_no;
                     SET @message ='budget amount increased'
```

```
            END;
            ELSE IF EXISTS (
            SELECT 1 from dbo.project p where p.project_no = @project_no
                    and @new_budget<= p.budget
                    )
            BEGIN
                     SET @message ='New budget must be greater than the current budget'

            END

    END

    declare @message varchar(100), @projec_no varchar(4), @new_budget float

    set @projec_no ='f1'
    set @new_budget =10
    exec IncreaseBudgetAmount @projec_no, @new_budget, @message output
    print @message

    set @projec_no ='p1'
    set @new_budget =10
    exec IncreaseBudgetAmount @projec_no, @new_budget, @message output
    print @message

    set @projec_no ='p1'
    set @new_budget =400000
    exec IncreaseBudgetAmount @projec_no, @new_budget, @message output
    print @message
```



```
Invalid Project Number
New budget must be greater than the current budget

(1 row affected)
budget amount increased

Completion time: 2020-03-24T18:25:14.2265416-04:00
```

3.
```
create function GetBudgetAmount (@project_name varchar(50) )
returns float
as
begin
return
(
select budget from project where project_name = @project_name
)
```

```
end

select dbo.GetBudgetAmount('CRM system')
```

4. A transaction is a logical unit of processing in a DBMS which entails one or more database access operation. A Transaction can either be implicit such as an INSERT or UPDATE or explicit when defined to BEGIN TRANSACTION …. COMMIT/ROLLBACK TRANSACTION.

5. Consistency is and ACID property which states that database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof. In this case the DDL above creates a table with PK constraint, non-nullable columns. All these rules must be valid according to ACID property(specifically Consistentency)
6. Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially.

7. It voliates all the ACID properties.
   a. Atomicity
   b. Consistency
   c. Isolation
   d. Durability

8. Concurrency is need because DBMS are designed to support multiple users and processes operating various set of tasks such as Inserting records, Updating records while other processes read from those records at the same time. If concurrency is uncontrolled, the following issues with be present: dirty reds, lost update problem, non-repeatable read, and the phantom read problem.

9. A local transaction occurs within a database. Distributed transaction involves two or more databases, and often time distributed across the network(s).

10. A save point marks a specified point within the transaction so that all updates that follow can be canceled without canceling the entire transaction

11. Locking means that the transaction marks the data that it accesses so that the DBMS knows not to allow other transactions to modify it until the first transaction succeeds or fails.

Row versioning provides each reading transaction the prior, unmodified version of data that is being modified by another active transaction. This allows readers to operate without acquiring locks, i.e., writing transactions do not block reading transactions, and readers do not block writers.

12. YES, by either changing the isolation level using the SET TRANSACTION ISOLATION LEVEL statement or by using hints. For example, for databases in "READ COMMITED" Isolation Level, NOLOCK hint allows the user to read non-committed data which essentially skips the waiting of exclusive placed by the writer process