STORED PROCEDURES

Stored Procedures

- A stored procedure is prepared SQL code that we save so we can reuse the code over and over again. So if we think about a query that we write over and over again, instead of having to write that query each time we would save it as a stored procedure and then just call the stored procedure to execute the SQL code that we saved as part of the stored procedure.
- In addition to running the same SQL code over and over again we also have the ability to pass parameters to the stored procedure.

SYNTAX

CREATE PROCEDURE <procedure_EID>

AS

BEGIN

<SQL Statement>

END

EXECUTE <procedure_EID>

EXEC <procedure_EID>

 <procedure_EID>

Stored Procedures

Example 1 : Simple Procedure to get the details of Delhi employees

```
CREATE PROCEDURE SHDELEMP
AS
BEGIN
   SELECT * FROM EMP WHERE CITY = 'DELHI';
END;
```

Stored Procedures

Example 2 : **Parameterized** Procedure to get the details of employees of the specified city

```
CREATE PROCEDURE SHOWEMP @X VARCHAR(20)
AS
BEGIN
   SELECT * FROM EMP WHERE CITY = @X;
END;
```

Example 3 : **Parameterized** Procedure to get the contents of the specified table

```
CREATE PROCEDURE SHOW @Y VARCHAR(20)
AS
BEGIN
            EXEC('SELECT * FROM ' + @Y );
END;
```

Stored Procedures

Example 4 : **Parameterized** Procedure to insert the data in the emp_sal table

```
CREATE PROCEDURE IN_EMP_SAL
@ID VARCHAR(5), @A VARCHAR(20), @B VARCHAR(20), @X INT
AS
BEGIN
        SET NOCOUNT ON ;
        INSERT INTO EMP_SAL VALUES
        ( @ID, @A, @B, @X );

        SELECT * FROM EMP_SAL
        WHERE EID=@ID;
END;
```

Stored Procedures

A stored procedure with parameters:

SYNTAX

CREATE PROCEDURE <procedure_EID>

@ var1 datatype (size), var2 datatype (size)

AS

BEGIN

[SET NOCOUNT ON/OFF]

<SQL Statement>

END

EXECUTE <procedure_EID> var1 , var2

DROP PROCEDURE <procedure_EID>

ASSIGNMENT

ASSIGNMENT – 8

A-1 : CREATE BELOW PROCEDURES IN THE INVENTORY DATABASE AS SPECIFIED :

ADDSUPPLIER – SHOULD ADD THE SUPPLIER IN THE SUPLIER TABLE AND DISPLAY THE DETAILS OF THE NEW SUPPLIER ADDED.

ADDPRO – SHOULD ADD THE PRODUCT IN THE PRODUCT TABLE AND DISPLAY THE DETAILS OF THE NEW PRODUCT ADDED.

ADDCUST – SHOULD ADD THE CUSTOMER IN THE CUSTOMER TABLE AND DISPLAY THE DETAILS OF THE NEW CUSTOMER ADDED.

ADDORDER – SHOULD ADD THE ORDER IN THE ORDERS TABLE AND DISPLAY THE DETAILS OF THE ORDER. ORDER DATE SHOULD BE CURRENT DATE AND SHOULD COME AUTOMATICALLY.

TRANSACTIONS

Transactions

- A transaction is a unit of work that is performed against a database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing a transaction on the table.

Properties of Transactions

Transactions have the following four standard properties, usually referred to by the acronym ACID:

Atomicity: Ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure, and previous operations are rolled back to their former state.

Consistency: Ensures that the database properly changes state upon a successfully committed transaction.

Isolation: Enables transactions to operate independently of and transparent to each other.

Durability: Ensures that the result or effect of a committed transaction persists in case of a system failure

Transactions

There are following commands used to control transactions:

- **COMMIT**: To save the changes.
- **ROLLBACK**: To roll back the changes.
- **SAVEPOINT**: Creates points within groups of transactions in which to ROLLBACK.

AUTO INCREMENT FIELD

Auto Increment

Auto Increment allows a unique number to be generated automatically when a new record is added in to the table.

- Identity (START, INCREMENT)

Example :

```
create table emp2
(id int identity (1,1) primary key,
EID varchar (30),
age int);
```

SEQUENCES

Sequences

Sequences are the objects in SQL Server that is used to generate a number sequence. These are normally used to create a unique number.

- **Syntax**

```
CREATE SEQUENCE sequence_EID
[ AS datatype ]
[ START WITH value ]
[ INCREMENT BY value ]
[ MINVALUE value | NO MINVALUE ]
[ MAXVALUE value | NO MAXVALUE ]
[ CYCLE | NO CYCLE ]
[ CACHE value | NO CACHE ];
```

Sequences

- **Example 1:**

```
Create sequence MYSEQ
AS INT
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 1000
No CYCLE
CACHE 5;
```

- **Example 2:**

```
Create sequence MYSEQ
START WITH 1
INCREMENT BY 1
```

```
Drop Sequence MYSEQ;
```

NOTE: Sequences are the global objects, however, auto increment works on the table level