**SQL:**

On Creating DB two files will get generate:

.MDF file - Data File (Contains actual data)  
.LDF file - Transaction Log file (Used to recover the database)

The primary purpose of an LDF file is to provide the ACID concept – Atomicity, Consistency, Isolation, and Durability

* atomicity: if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged
* consistency: any transaction brings the database from one valid state to another
* isolation: the execution of concurrent transactions brings the database to a state as if the transactions were executed serially, one by one
* durability: once committed, the transaction remain so, even in the case of errors, power loss, or crashes

**ALTER**  
ALTER DATABASE TESTUTCNW MODIFY NAME = TESTUTCNEW  
  
**ALTER: USING SP**  
EXECUTE SP\_RENAMEDB 'TESTUTCNEW', 'TESTUTCNEW1'

**DROP DATABASE:**

DROP DATABASE TESTUTCNEW1.: NOT WORK FOR CURRENT DB IN USE

Msg 3702, Level 16, State 3, Line 1

Cannot drop database "TESTUTCNEW1" because it is currently in use.

Alter Database TESTUTCNEW1 SET SINGLE\_USER WITH ROLLBACK IMMEDIATE

Put the database in single user mode and then drop the database.

With Rollback Immediate option: It will rollback all incomplete transactions and closes the connection to the database.

**Constraint**

**Primary key**

The PRIMARY KEY constraint uniquely identifies each record in a database table.

**Foreign key** constraint prevents invalid data form being inserted into the foreign key column.

**What is the difference between Primary key constraint and Unique key constraint?**   
**1.** A table can have only one primary key, but more than one unique key  
**2.** Primary key does not allow nulls, whereas unique key allows one null

**Unique:**

We use UNIQUE constraint to enforce uniqueness of a column i.e the column shouldn't allow any duplicate values

Alter Table Table\_Name  
Add Constraint Constraint\_Name Unique(Column\_Name) or (Col1, Col2)

**Default Constraint:**

ALTER TABLE tblPerson  
ADD CONSTRAINT DF\_tblPerson\_GenderId  
DEFAULT 1 FOR GenderId

To insert Default value for the column:

INSERT INTO TEST1 VALUES(DEFAULT)

**CHECK Constraint:**

**Limit the range of the values that can be entered for a column.**

**Limits the age between ZERO and 150.**  
ALTER TABLE tblPerson  
ADD CONSTRAINT CK\_tblPerson\_Age CHECK (Age > 0 AND Age < 150)

ALTER TABLE { TABLE\_NAME }  
ADD CONSTRAINT { CONSTRAINT\_NAME } CHECK ( BOOLEAN\_EXPRESSION )

If the BOOLEAN\_EXPRESSION returns true, then the CHECK constraint allows the value, otherwise it doesn't. Since, AGE is a nullable column; it's possible to pass null for this column, when inserting a row. When you pass NULL for the AGE column, the boolean expression evaluates to **UNKNOWN**, and allows the value.

**DROP Constraint:**

Before dropping the constraint, make sure all the foreign key references are dropped first.

alter table test1

drop constraint unq

**Cascading referential integrity constraint**

1. **No Action**: This is the default behaviour. No Action specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, an error is raised and the DELETE or UPDATE is rolled back.  
     
     
   **2. Cascade**: Specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, all rows containing those foreign keys are also deleted or updated.  
     
     
   **3. Set NULL**: Specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, all rows containing those foreign keys are set to NULL.    
     
     
   **4. Set Default**: Specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, all rows containing those foreign keys are set to default values.

Column:

**Identity column**

If a column is marked as an identity column, then the values for this column are automatically generated, when you insert a new row into the table.

Create Table tblPerson  
(  
PersonId int Identity(1,1) Primary Key,  
Name nvarchar(20)  
)

CREATE TABLE TESTR(ID INT IDENTITY(1,1))

INSERT INTO TESTR DEFAULT VALUES

To Insert values explicitly in default column

SET IDENTITY\_INSERT TESTR ON

INSERT INTO TESTR(ID)

VALUES(3)

SET IDENTITY\_INSERT TESTR OFF

Retrieving the identity values:

SELECT @@IDENTITY

@@IDENTITY returns the last identity value generated for any table in the current session, across all scopes.

SELECT SCOPE\_IDENTITY()  
SCOPE\_IDENTITY returns the last identity value generated for any table in the current session and the current scope.

SELECT IDENT\_CURRENT('TESTR')  
IDENT\_CURRENT returns the last identity value generated for a specific table in any session and any scope.

SCOPE\_IDENTITY() returns the last identity value that is created in the same session (Connection) and in the same scope (in the same Stored procedure, function, trigger). Let's say, I have 2 tables tblPerson1 and tblPerson2, and I have a trigger on tblPerson1 table, which will insert a record into tblPerson2 table. Now, when you insert a record into tblPerson1 table,  SCOPE\_IDENTITY() returns the idetentity value that is generated in tblPerson1 table, where as @@IDENTITY returns, the value that is generated in tblPerson2 table. So, @@IDENTITY returns the last identity value that is created in the same session without any consideration to the scope. IDENT\_CURRENT('tblPerson') returns the last identity value created for a specific table across any session and any scope.

**RESEED the identity value in the table.**  
DBCC CHECKIDENT(t6, RESEED, 5)

IDENT\_INCR('TABLE'):

Returns the increment value specified during the creation of an identity column in a table or view that has an identity column.

Returns NULL on error or if a caller table does not have identity column or table is not present.

IDENT\_SEED('TABLE')

Returns the original seed value specified during the creation of an identity column in a table or view that has an identity column.

Returns NULL on error or if a caller table does not have identity column or table is not present.

**Question:**

When error occurs the Identity value is skipped .  
So if u don't want to skip the Identity value use the below code..   
  
BEGIN TRANSACTION

INSERT INTO TBLPERSON VALUES('BOBAYYA','HBC',2)

IF(@@ERROR = 2627)

BEGIN

DECLARE @ID INT

SET @ID = @@IDENTITY

DBCC CHECKIDENT('TBLPERSON',RESEED, @ID)

END

ELSE

BEGIN

COMMIT TRANSACTION

END

Setting the Id to current Identity value. So when u insert the row next time the Id value is not skipped. it will continue consecutively.