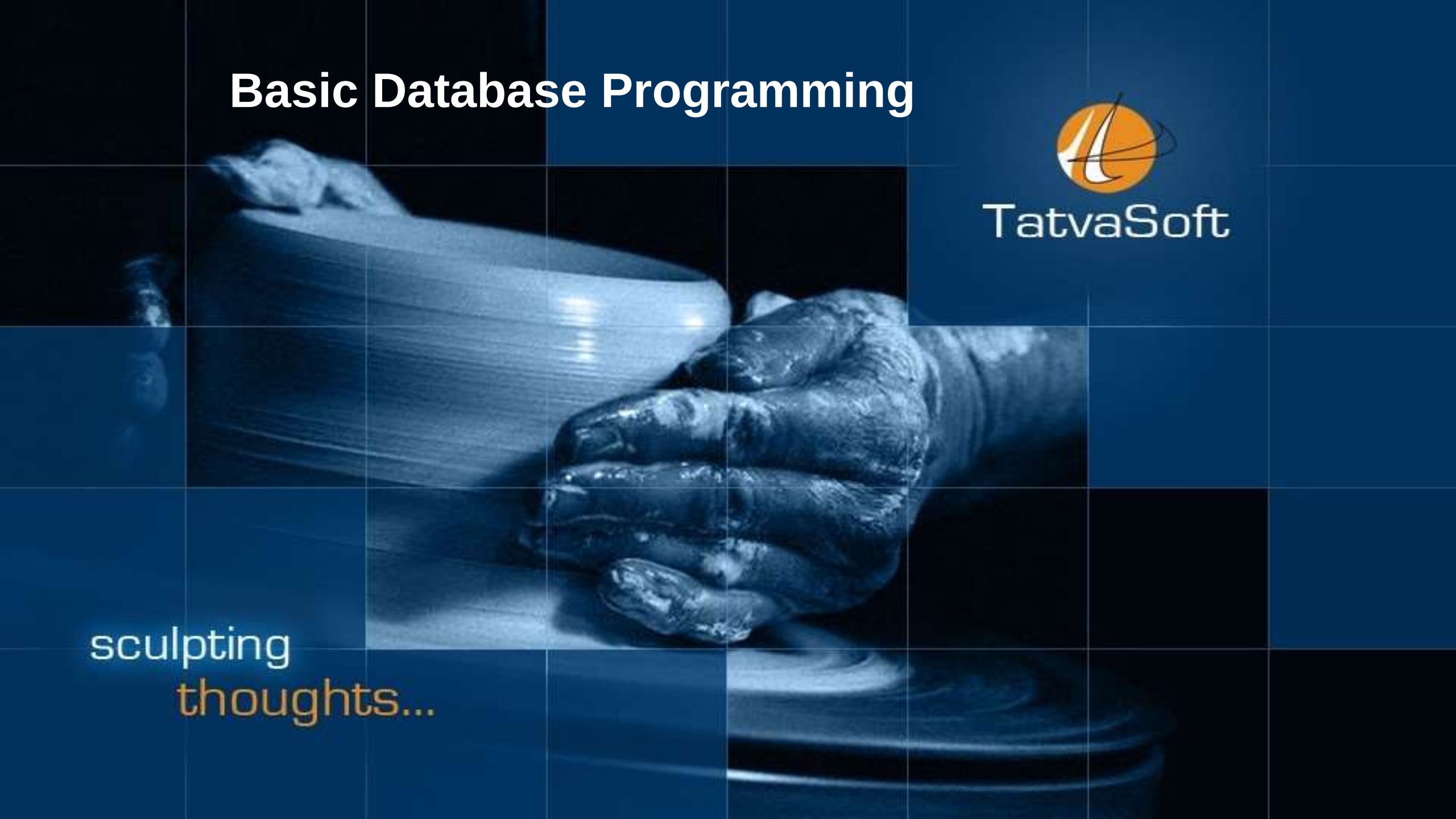
Basic Database Programming



Introduction of Database

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.

General Objectives:

1. Elimination of data redundancy
2. Simplify the use of data files
3. Lower the cost of storing and retrieving data
4. Improve accuracy and consistency
5. Provide data security from unauthorized use

Database Management System

A database management system is a suite of software applications that together make it possible for people or businesses to store, modify, and extract information from a database.

e.g,

Microsoft SQL Server Management Studio

SQL Statement

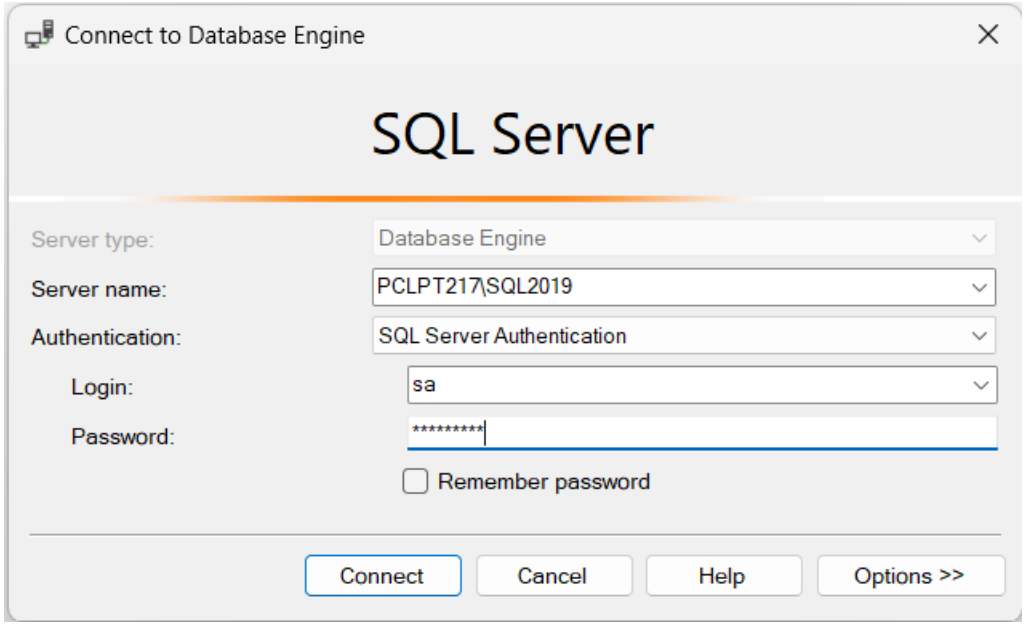
SQL is a structured query language for accessing and manipulating databases.

What can SQL do?:

1. SQL can execute queries against a database
2. SQL can retrieve data from a database
3. SQL can insert/update/delete records in/from a database
4. SQL can create new database
5. SQL can create new tables in database
6. SQL can create stored procedures in a database
7. SQL can create views in a database
8. SQL can set permissions on tables, procedures, and views

How to create a database?

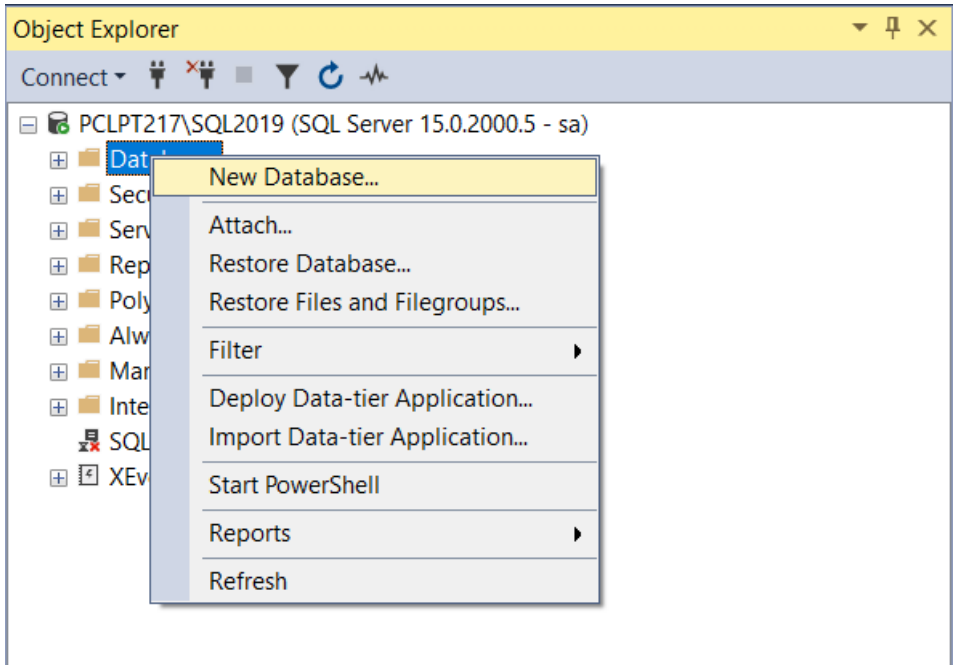
1. Connection sql server authentication



When you open management studio, a connection window with sql server will be opened as above figure shows. In this window you will enter login data for connection with sql server.

How to create a database?

2. Create new database

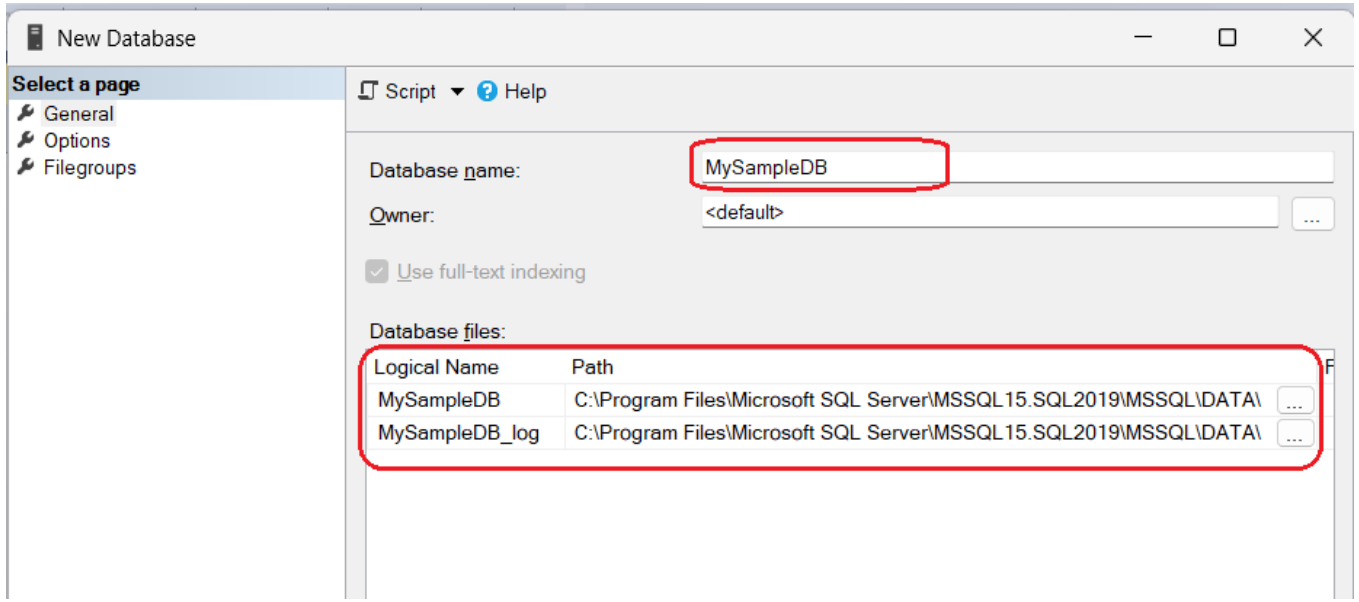


To create new database, right click on the “Databases” node in the left panel in the management studio.You will get couple of options in the context menu. Select the option 'New Database'.

TatvaSoft

How to create a database?

3. Name your database



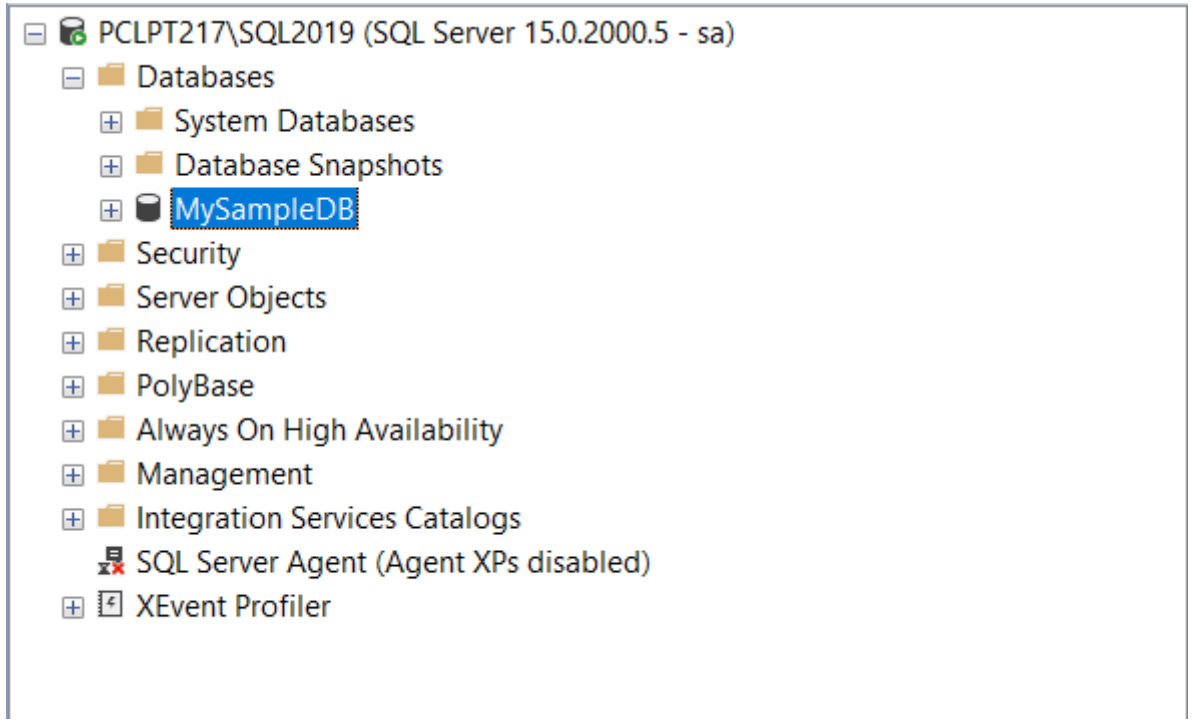
Once you select 'New Database' option, you will get another dialog box where you have to specify few details. The most important one is the database name. In the above screenshot, I have specified the new database name as “MySampleDatabase”.

How to create a database?

3.1 Name your database

You may notice two other circled fields called "logical name" and "path". When any database is created, there is at least 2 files created in the hard disk. One file is called "Data file" and it has an extension of .MDF. The other file is called "log file" and it has an extension of .LDF by default.

How to create a database?



Once you specify require detail click on Ok button to finish our database creation and you will see one database with specified name is created as above figure shows.

Database Table

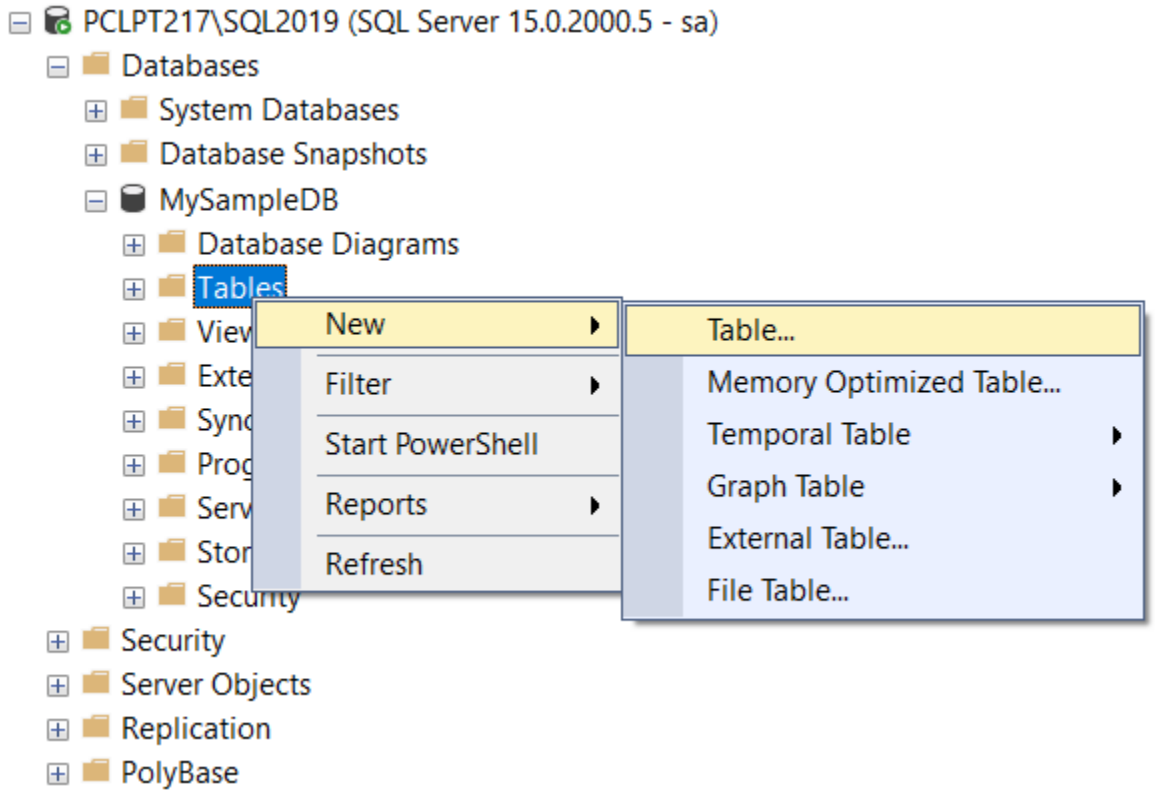
The foundation of every **R**elational **D**atabase **M**anagement **S**ystem is a database object called table. Every database consists of one or more than one table, which store the database's data or information or records. Each table has its own unique name and consists of columns and rows.

Way to create table:

1. Creating a table in data structure mode
2. Creating a table using SQL command

How to create a table?

1. Creating a table in data structure mode

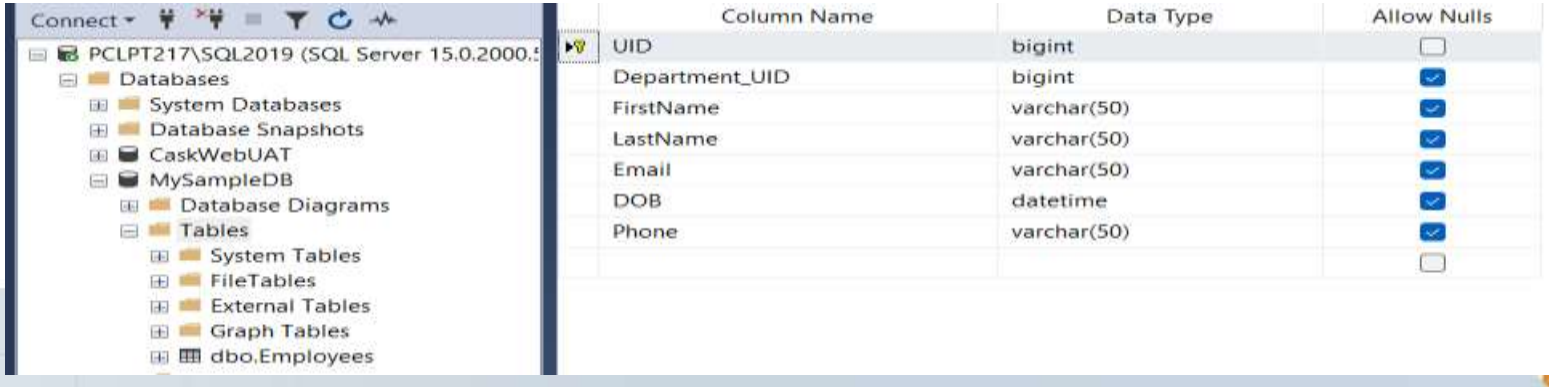


To create a table, right click mouse button over 'Tables' option under database and choose the 'New Table' option. The new table's data structure then opens composed of main two parts: spread sheet for creation of the columns and column's properties box as figure above screen.

How to create a table?

To insert a field, you must write the name of the column, choose the kind of data and examine if it will accept null values. After, with the new field selected, you will be able to set all its properties in the *Column Properties* box. Amongst the existing properties, we can highlight the Identity Specification option, where you can attribute the identity property and set the auto increment of the field, as shown in above figure.

To attribute a primary key to a field, just select it(field) and right click the *Set Primary Key* button located in the *Table Designer* toolbar. To finish the creation of the table, just click over the “X” in the right hand side of the table structure window or press Ctrl+S. The Management Studio asks if you wish to save this table and what name should be attributed to this new object. Once it is saved, the table begins to appear in the list of the *Tables* folder as shown in below screen. e.g Employee



TatvaSoft

How to create a table?

2. Creating a table using sql command

SQL Syntax:

```
CREATE TABLE "table_name"
    ("column 1" "data_type_for_column_1",
    "column 2" "data_type_for_column_2",
    ... )
```

Example:

```
CREATE TABLE Employee
    (UID bigint primary key identity(1,1),
    FirstName varchar(50) not null,
    LastName varchar(50) not null,
    Email varchar(250),
    DOB datetime not null,
    Phone varchar(20))
```

Data type

SQL data type is an attribute that specifies type of data of any object. Each column, variable and expression has related data type in SQL.

You would use these data types while creating your tables. You would choose a particular data type for a table column based on your requirement.

Reference :

http://www.tutorialspoint.com/sql/sql-data-types.htm

Primary key

Primary Key is used to uniquely identify each row in our table. An SQL Primary Key can consist of one or more fields on a table. When multiple fields are used as an primary key, they are called a composite primary key.

Primary keys can be specified either when the table is created (using SQL CREATE TABLE) or by changing the existing table structure (using SQL ALTER TABLE).

e.g. 1. Using SQL CREATE TABLE command

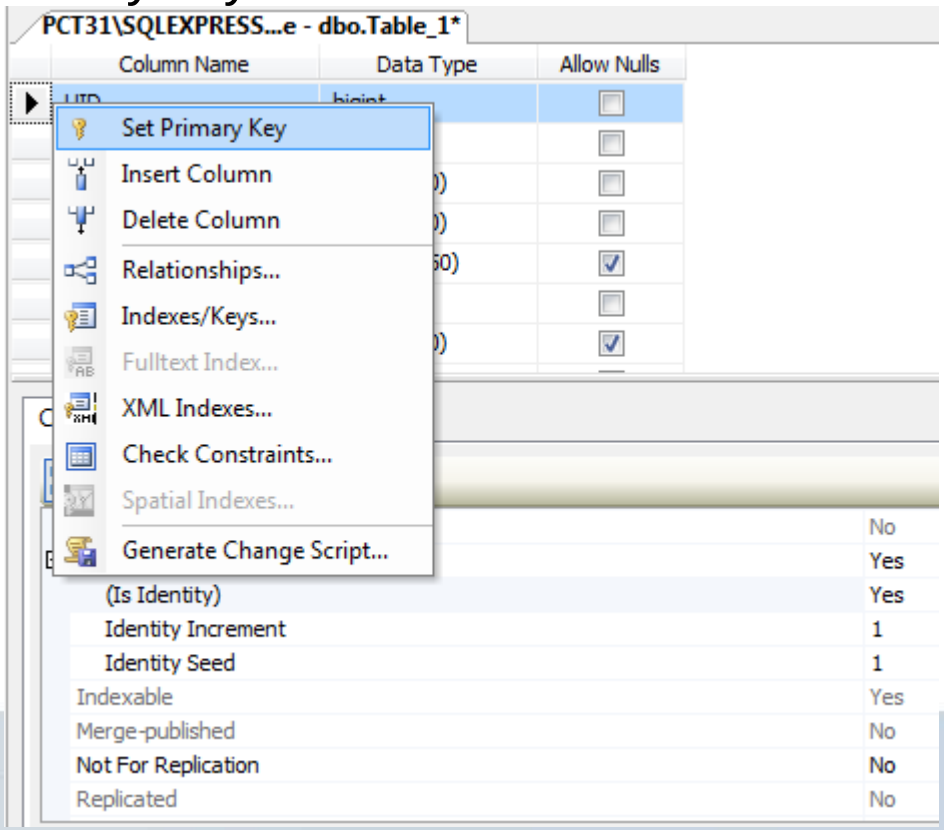
```
CREATE TABLE Employees(Emp_UID bigint primary  key,
    FirstName varchar(50), LastName varchar(50))
```

2. Using SQL ALTER TABLE command

```
ALTER TABLE Employees ADD PRIMARY KEY (Emp_UID);
```

Primary key

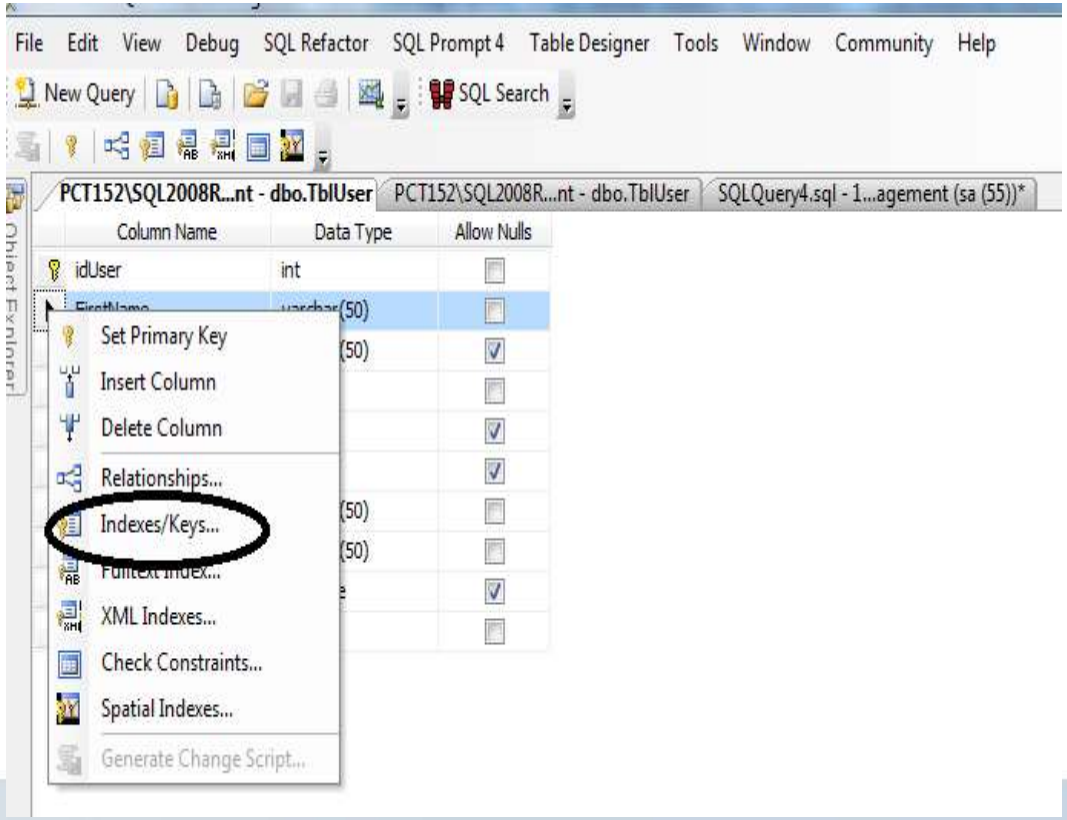
Primary keys can also be specified when designing the table. To attribute a primary key to a field, just select field and right click the *Set Primary Key* button located in the *Table Designer* toolbar as shown in below figure. Once you select it, you will see key image before field name that indicate that the field is marked as primary key field.



TatvaSoft

Unique key

Unique Key enforces uniqueness of the column on which they are defined. Unique Key creates a non-clustered index on the column. Unique Key allows only one NULL Value.



TatvaSoft

Foreign key

SQL Foreign Key Constraint, Foreign Key Constraint is a field or fields that points to the SQL Unique key constraint of another table. The purpose of the SQL Foreign key Constraint is to ensure referential integrity of the data. In other words, only those values that are supposed to appear in the database are permitted.

E.g, we have two table

1. Department

- UID bigint (PK)
- Name varchar(50)

2. Employee

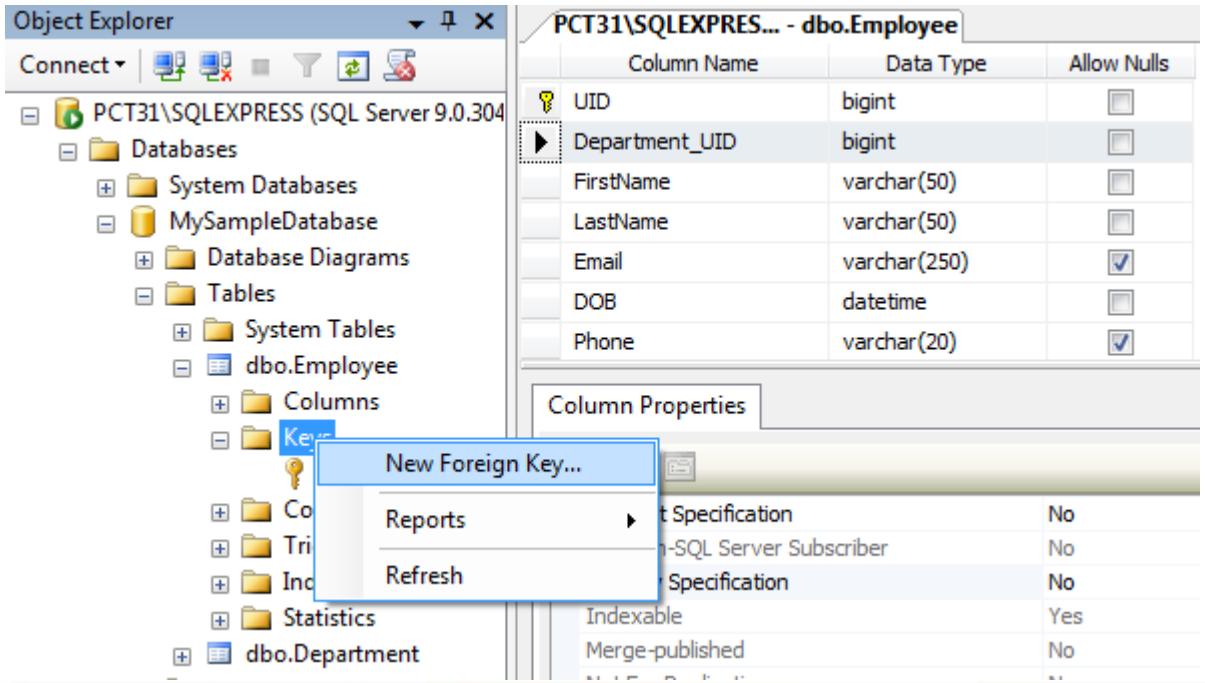
- UID bigint (PK)
- Department_UID bigint (FK)
- FirstName varchar(50)
- LastName varchar(50)

Here in Employee table's Department_UID field we can enter only those values that are appear in UID field in Department table.

Foreign key

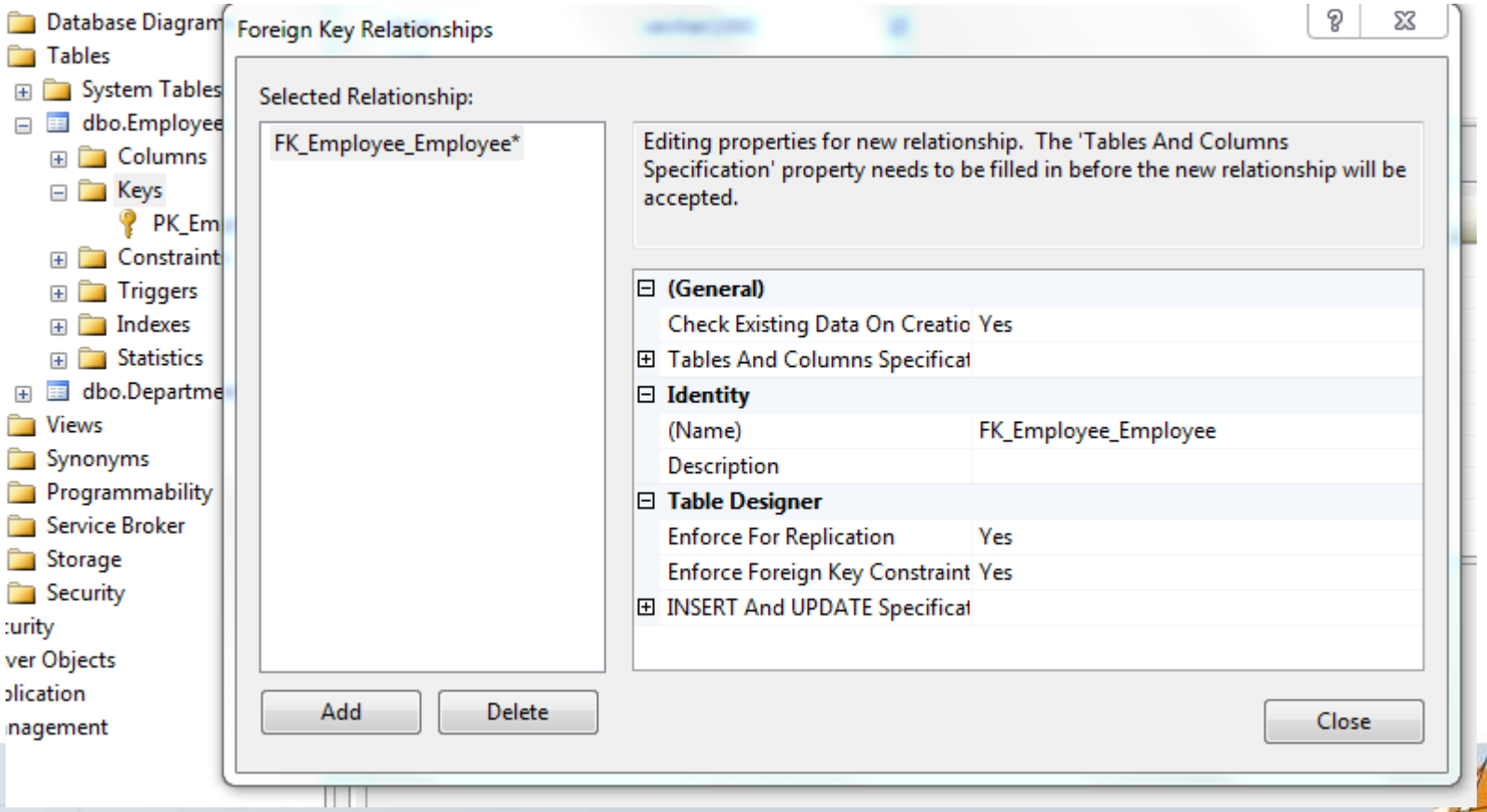
- Follow the below step to create foreign key.

  Step 1: Right click mouse button over 'Keys' option under database and choose the 'New Foreign Key...' option as figure below screen. You will get another dialog box name 'Foreign key Relationships'.



Foreign key

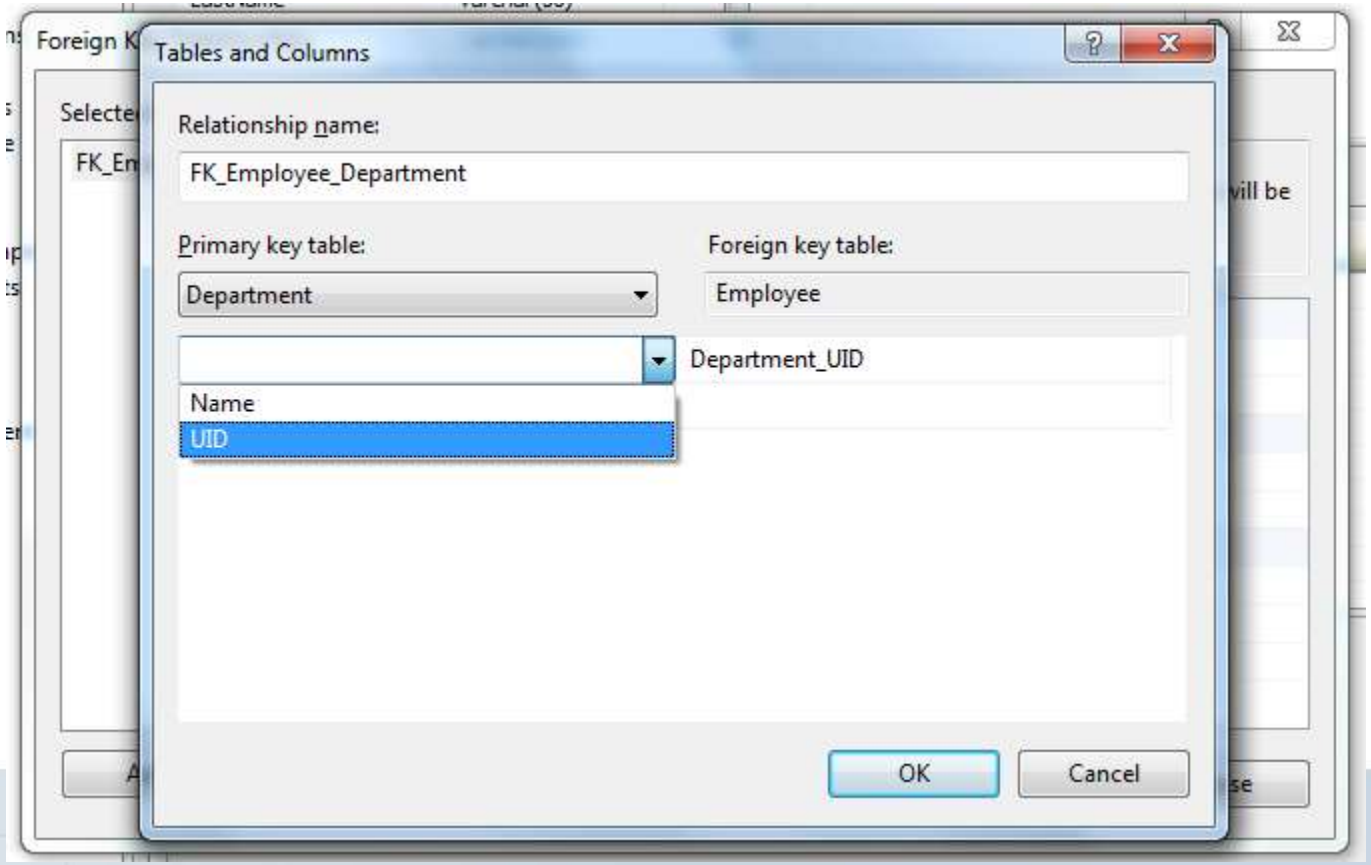
Step 2: Click on next column beside 'Tables and Columns Specification' option show in below figure. You will have a button click on it, you will get another dialog box name 'Tables and Columns'.



TatvaSoft

Foreign key

Step 3: In 'Tables and Columns' box you need to specify the Relationship name, Choose primary key table name(e.g, Department) and primary key field name(ie. UID) also select the foreign key field name(i.e Department_UID) below the Foreign key table name(e.g Employee) and click on ok button.

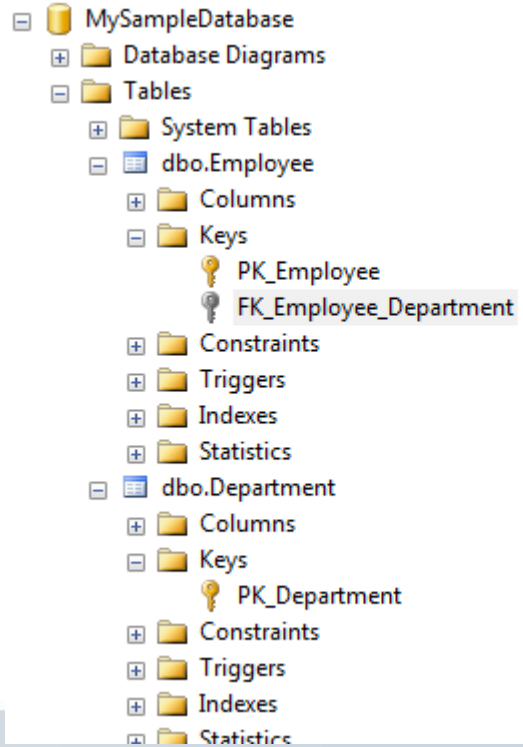


TatvaSoft

Foreign key

Step 4: Now close 'Foreign key Relationship' dialog box and press Ctrl + S to save table update and you will see new foreign key is created base on specified Foreign key name in 'Keys' folder as figure in below screen.

e.g, FK_Employee_Department



SQL Statement

SQL can be divided into two parts.

1. DML (Data Manipulation Language)
    1. SELECT – extract data from a database
    2. UPDATE – updates data in a database
    3. DELETE – delete data from a database
    4. INSERT INTO – inserts new data into a database
2. DDL (Data Definition Language)
    1. CREATE TABLE – creates a new table
    2. ALTER TABLE – modifies a table
    3. DROP TABLE – deletes a table

DML SQL Statement

1. SELECT

The SELECT statement is used to select data from a database.

Syntax :

```
SELECT column_name(s) FROM table_name
[WHERE  search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC | DESC]]
```

Example:

```
SELECT CategoryName, Description FROM categories
ORDER BY CategoryName;’
```

DML SQL Statement

1. UPDATE

   The UPDATE statement is used to update records in a table.

   Syntax :

   UPDATE table_name SET column_name1=value1,
   column_name2=value2…
   [WHERE search_condition]

   Example:

   UPDATE northwind.shippers_dup SET Email ='speedyexpress@gmail.com' WHERE ShipperID = '1';

DML SQL Statement

1. DELETE

The DELETE statement is used to delete records in a table.

Syntax :

```
DELETE FROM table_name
[WHERE  search_condition]
```

Example:

```
DELETE FROM Employee
WHERE FirstName='Ricky'
```

Note : If you omit the WHERE clause, all records will be deleted.

DML SQL Statement

1. INSERT INTO

   The INSERT INTO statement is used to insert new records in a table.

   Syntax :

   INSERT INTO table_name(column_name1,column_name2…)

   VALUES (value1,value2…)

   Example:

   INSERT INTO dbo.Employees

VALUES

(N'LastName',N'FirstName','Employees test','Ms.','1963-08-30 00:00:00.000','1993-05-03 00:00:00.000','14 Garrett Hill', 'Kirkland','WA','98052','USA','(206) 555-9482','452', NULL,'Michael is a graduate',5,'http://accweb/emmployees/peacock.bmp')

DDL SQL Statement

1. CREATE TABLE

   The CREATE TABLE statement is used to create a table in a database.

   Syntax :

```
CREATE TABLE table_name
(column_name1 data_type,
column_name2 data_type…)
```

   Example:

```
CREATE TABLE Employee
(UID bigint,
FirstName varchar(50),
LastName varchar(50),
DOB datetime)
```

DDL SQL Statement

1. ALTER TABLE

   The ALTER TABLE statement is use to add, delete or modify columns in an existing table.

   Syntax :

   1. To add new column

      ALTER TABLE table_name ADD column_name datatype

   2. To delete column

      ALTER TABLE table_name DROP column_name

   3. To change datatype of column in a table

      ALTER TABLE table_name

      ALTER COLUMN column_name datatype

DDL SQL Statement

1. ALTER TABLE

Example :

1. To add new column

```
ALTER TABLE Employee ADD Designation varchar(20)
```

2. To delete column

```
ALTER TABLE Employee DROP Designation
```

3. To change datatype of column in a table

```
ALTER TABLE Employee
ALTER COLUMN Designation varchar(50)
```

DDL SQL Statement

3. DROP TABLE

The DROP TABLE statement is used to delete a table.

Syntax:

```
DROP TABLE table_name
```

Example :

```
DROP TABLE Employee
```

Join

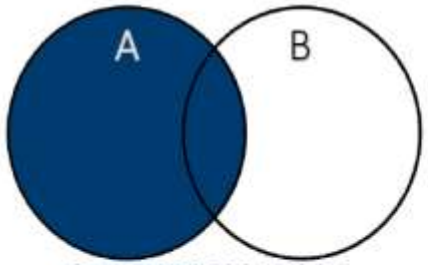
Joins in SQL Server allows the retrieval of data records from one or more tables having some relation between them. Logical operators can also be used to drill down the number of records to get the desired output from sql join queries.

Types of Join:

1. Inner Join

2. Outer Join
    1. Left Outer Join
    2. Right Outer Join
    3. Full Outer Join

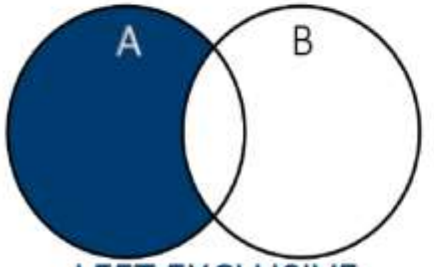
Join



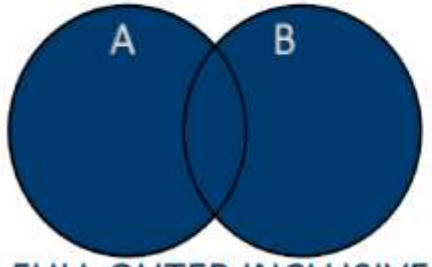
LEFT INCLUSIVE

SQL JOINS

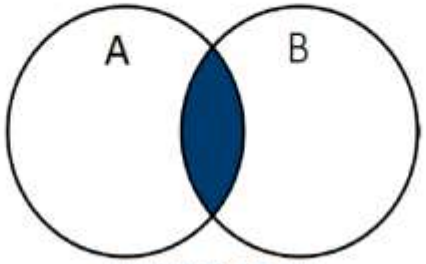
LEFT INCLUSIVE	RIGHT INCLUSIVE
SELECT *[Select List]* FROM TableA A LEFT OUTER JOIN TableB B ON A.Key= B.Key	SELECT *[Select List]* FROM TableA A RIGHT OUTER JOIN TableB B ON A.Key= B.Key
LEFT EXCLUSIVE	**RIGHT EXCLUSIVE**
SELECT *[Select List]* FROM TableA A LEFT OUTER JOIN TableB B ON A.Key= B.Key WHERE B.Key IS NULL	SELECT *[Select List]* FROM TableA A LEFT OUTER JOIN TableB B ON A.Key= B.Key WHERE A.Key IS NULL
FULL OUTER INCLUSIVE	**FULL OUTER EXCLUSIVE**
SELECT *[Select List]* FROM TableA A FULL OUTER JOIN TableB B ON A.Key = B.Key	SELECT *[Select List]* FROM TableA A FULL OUTER JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL OR B.Key IS NULL
INNER JOIN	
SELECT *[Select List]* FROM TableA A INNER JOIN TableB B ON A.Key = B.Key	



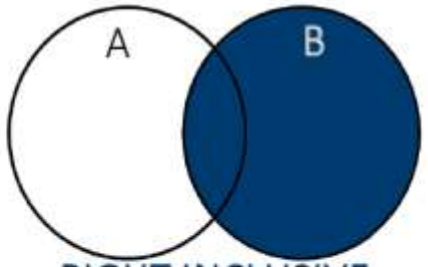
LEFT EXCLUSIVE



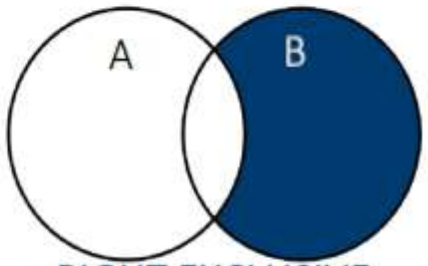
FULL OUTER INCLUSIVE



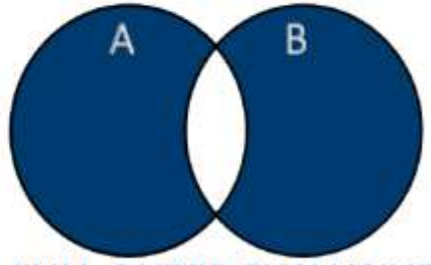
INNER JOIN



RIGHT INCLUSIVE



RIGHT EXCLUSIVE



FULL OUTER EXCLUSIVE

TatvaSoft

Type of Join

1. Inner Join:

Inner Join is a default type join of SQL Server. It uses logical operators such as =, <, > to match the records in two tables.

Example: (Inner Join with = operator)

```
SELECT s.SupplierID, p.ProductName, S.CompanyName
FROM northwind.suppliers s JOIN northwind.products p
    ON s.SupplierID = p.SupplierID
WHERE s.CompanyName IN ('Exotic Liquids','Specialty Biscuits, Ltd.','Escargots
    Nouveaux') ORDER BY s.SupplierID;
```

Type of Join

2.1 Left Outer Join:

Left Outer Join returns all the rows from the table specified first in the Left Outer Join Clause. If in the left table any row has no matching record in the right side table then that row returns null column values for that particular tuple.
Example:

```
SELECT * FROM dbo.Employees E
LEFT JOIN dbo.EmployeeTerritories  ET
    ON ET.EmployeeID = E.EmployeeID
```

2.2 Right Outer Join:

Right Outer Join is exactly the reverse method of Left Outer Join. It returns all the rows from right table and returns null values for the rows having no match in the left joined table.
Example:

```
SELECT * FROM dbo.EmployeeTerritories ET
RIGHT JOIN dbo.Employees E
    ON E.EmployeeID = ET.EmployeeID
```

Type of Join

2.3 Full Outer Join:

Full outer join returns all the rows from both left and right joined tables. If there is any match missing from the left table then it returns null column values for left side table and if there is any match missing from right table then it returns null value columns for the right side table.

Example:

```
SELECT * FROM dbo.EmployeeTerritories ET
FULL OUTER JOIN dbo.Employees E
    ON E.EmployeeID = ET.EmployeeID
```

Aggregate Functions: SUM(), COUNT(), AVG(), MIN(), Max()

- The COUNT() function returns the number of rows in a database table.
- The SUM() function returns the total sum of a numeric column.
- The AVG() function calculates the average of a set of values.
- The MIN() aggregate function returns the lowest value (minimum) in a set of non-NULL values.
- The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

What Is Group By in SQL?

- The Group By statement is used to group together any rows of a column with the same value stored in them, based on a function specified in the statement. Generally, these functions are one of the aggregate functions such as MAX() and SUM().

- ```
  SELECT column_1, function_name(column_2)
  FROM table_name
  WHERE [condition]
  GROUP BY column_name
  ORDER BY column_name;
  ```
  ```

Stored Procedure

Syntax:

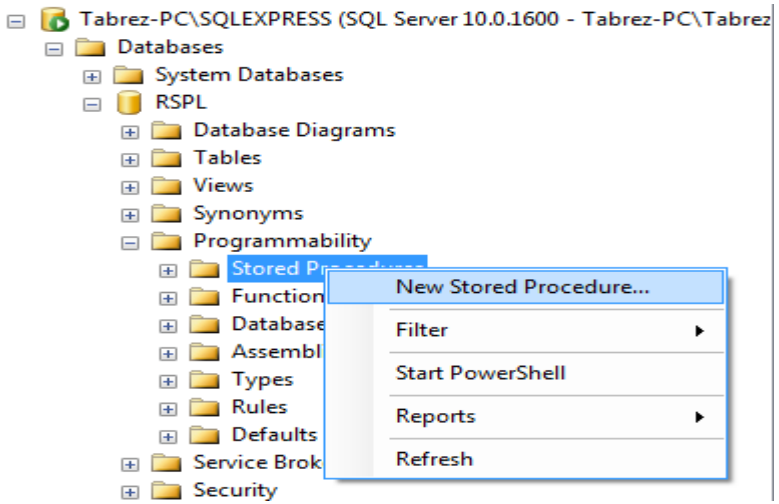
```
CREATE PROCEDURE <procedure_name>
    [@parameter1 data_type, @parameter2 data_type…]
    AS
    BEGIN
        /* SQL Statements. */
    END
    GO
```

Example:

```
CREATE PROCEDURE usp_SelectAllEmployee
    AS
    BEGIN
        SELECT * FROM Employee
    END
    GO
```

Stored Procedure

How to create Stored Procedure?



To create stored procedure right click on ‘Stored Procedure’ folder under ‘Programmability’ folder and choose ‘New Stored Procedure…’ option as shown in above figure.

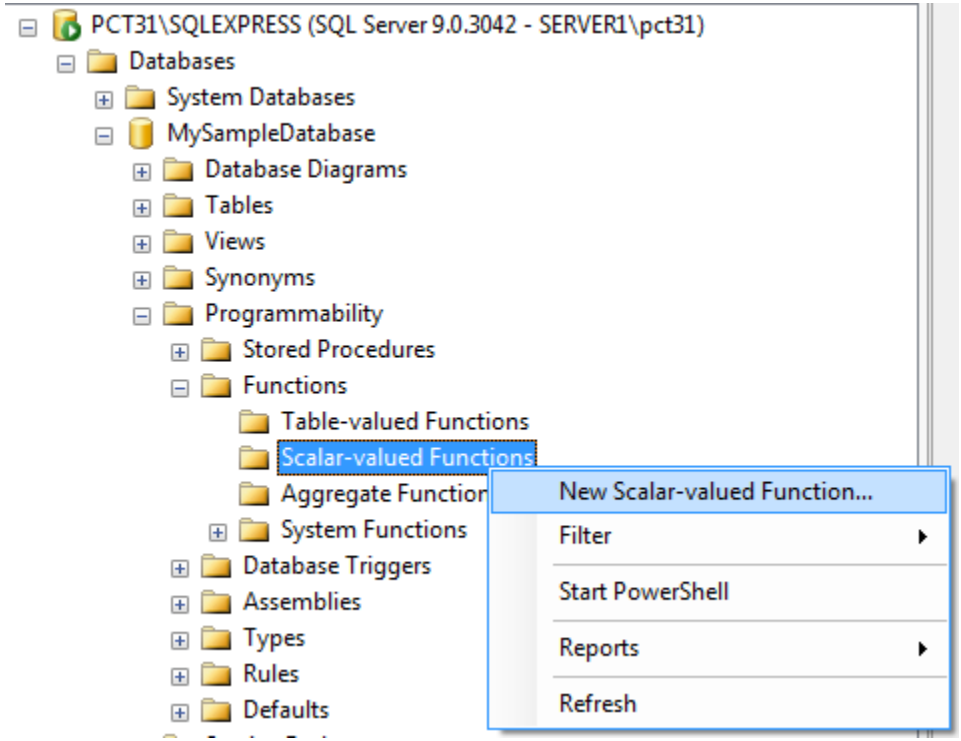
User-Defined Function

Types of UDF:

1. Scalar function
   - Return a single value.
2. In Line Table function
   - Return a single table variable that was created by a select statement.

Type of UDF

1. Scalar Function



To add new scalar function right click on 'Scalar-valued Functions' under 'Functions' folder and choose 'New Scalar-valued Function...' option as shown in above figure. You will see the scalar function's syntax in SQL pane beside object explorer, update it as per need and save it by pressing Ctrl+S and give function name when ask to save.

Type of UDF

1. Scalar Function

Example:

```
CREATE FUNCTION dbo.DateOnly(@InDateTime datetime)
    RETURNS varchar(10)
    AS
     BEGIN
        DECLARE @MyOutput varchar(10)
        SET @MyOutput = CONVERT(varchar(10),@InDateTime,101)
         RETURN @MyOutput
     END
```

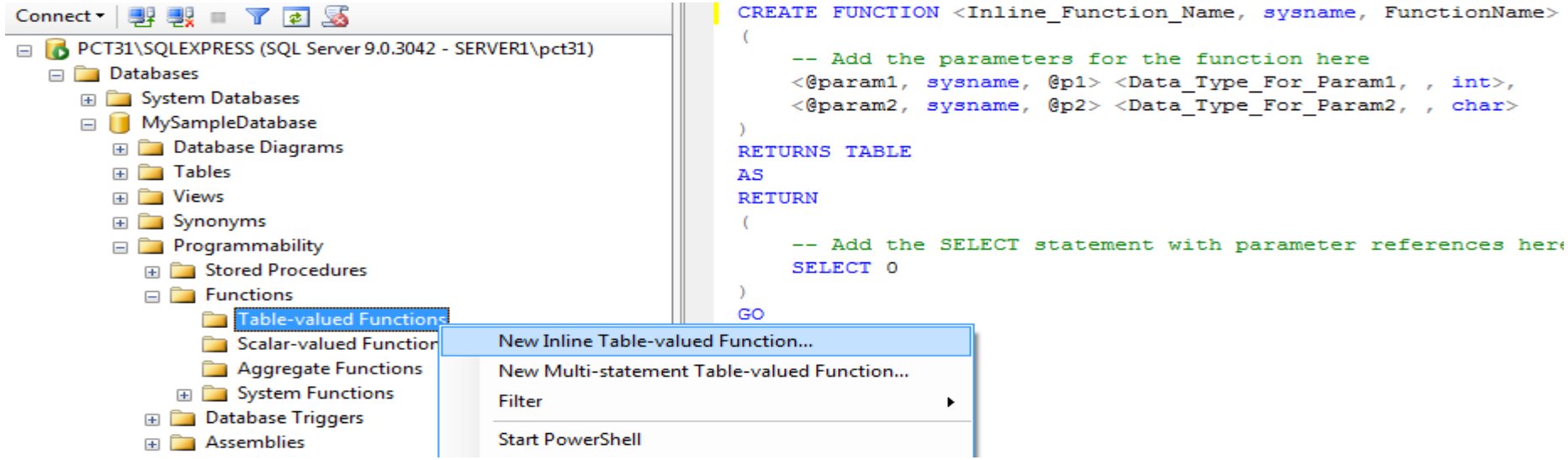
To call function:

```
SELECT dbo.DateOnly(GETDATE())
```

Above function takes full date and time as parameter and returns only date in varchar datatype.

Type of UDF

2. In-line Table Function



To add new In-line Table function right click on 'Table-valued Functions' under 'Functions' folder and choose 'New Inline Table-valued Function...' option as shown in above figure. You will see the Inline function's syntax in SQL pane beside object explorer, update it as per need and save. You will see the created function under 'Table-valued Functions' folder.

Type of UDF

2. Inline Table Function

Example:

```
CREATE FUNCTION dbo.LookByFName(@FirstLetter char(1))
    RETURNS TABLE
        AS
            BEGIN
            RETURN (SELECT *
                FROM employee
                WHERE LEFT(fname, 1) =  @FirstLetter)
        END
```

To call function:

```
SELECT * FROM dbo.LookByFName('A')
```

Above function returns all the employee whose first name start with a character 'A'.

View

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

Syntax:

```
CREATE [OR REPLACE] VIEW view_name AS
    SELECT column_name(s)
        FROM table_name
        WHERE condition
```

Example:

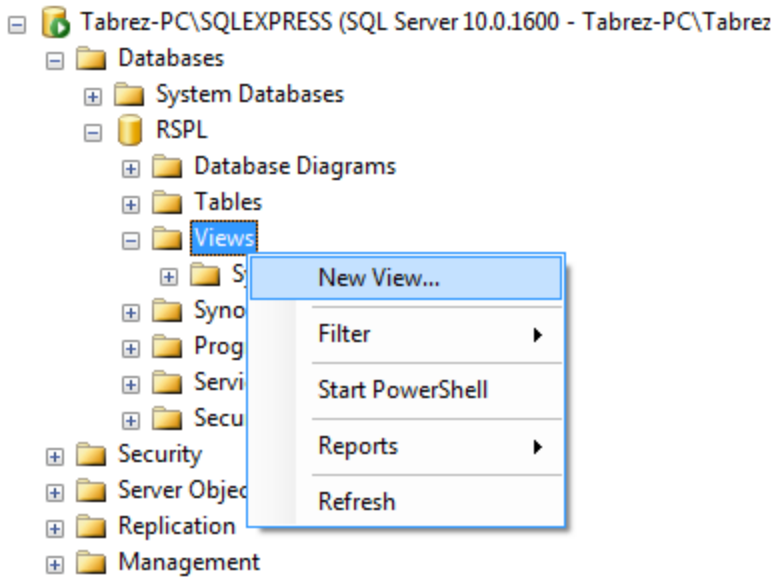
```
CREATE VIEW [Employee_List] AS
    SELECT EmployeeID,FirstName,LastName
        FROM Employee
        WHERE Salary > 5000
```

We can query the view above as follows:

```
SELECT * FROM [Employee_List]
```

How to create view?

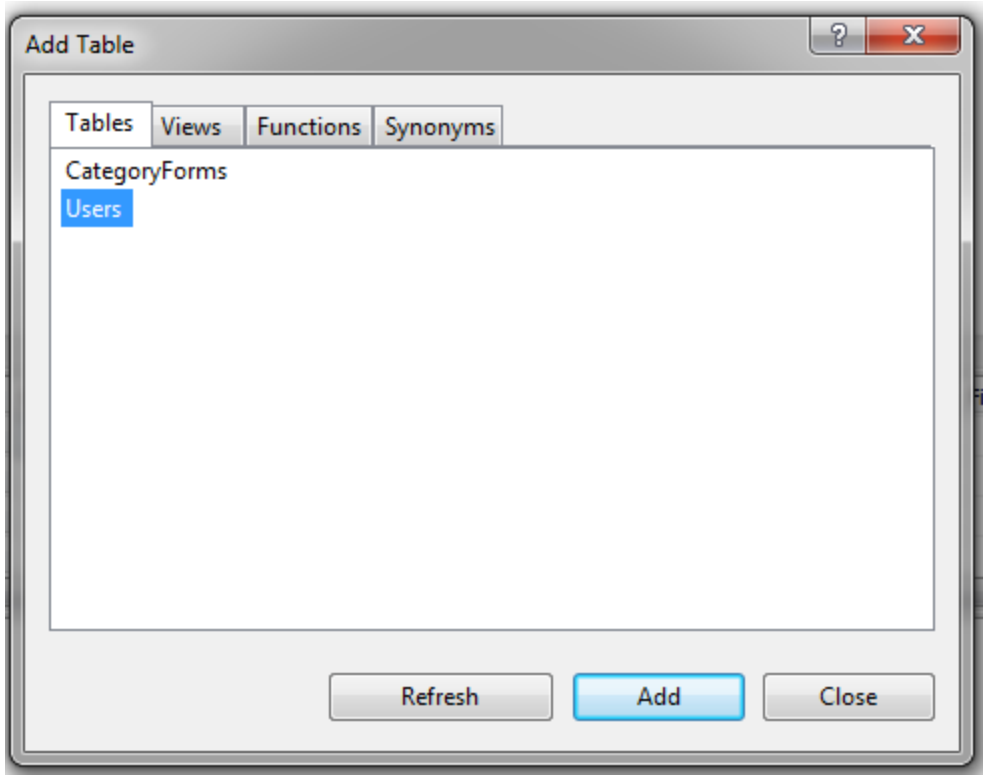
Step 1:



To create view right click on ‘Views’ folder under database and choose ‘New View…’ option as shown in above figure.

How to create view?

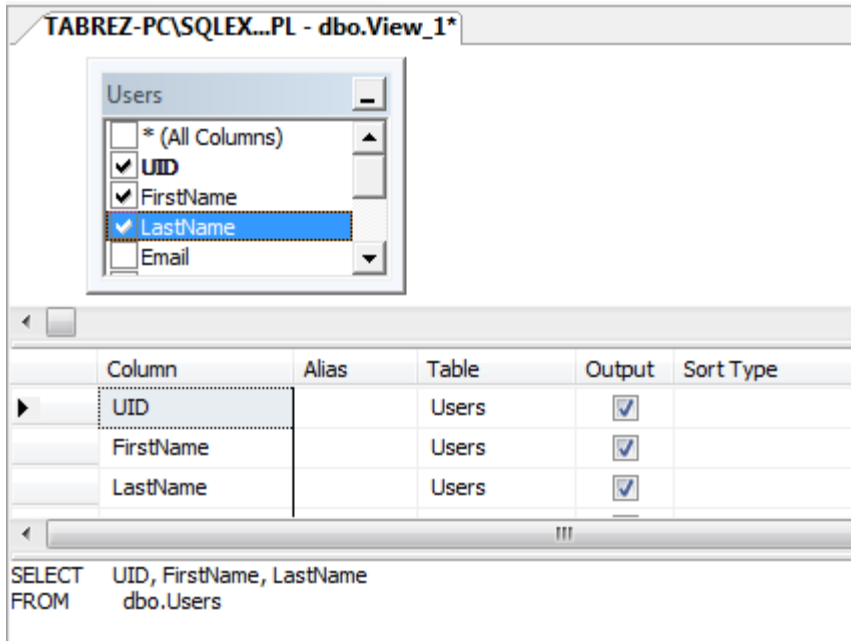
Step 2:



Once you click on ‘New View…’ option you will have a dialogbox with list of Tables, choose table for which you want to create view and add it in query builder by click on Add button and close it as shown in above figure.

How to create view?

Step 3:



Once you close ‘Add Table’ dialog box you will see your selected table added in query builder, corresponding query in SQL Pane (i.e. last row) and you can update it as shown in above figure.

After creating query to save view press Ctrl+S and give view name and save it you will see created view under ‘View’ folder.

Reference link

1. Database, Table, Sql Statement
    1. http://www.codingfusion.com/Post/75-Important-queries-in-SQL-Server-every-developer-should-know
    2. https://www.freecodecamp.org/news/basic-sql-commands/
    3. https://www.sqlservertutorial.net/
2. Function
    1. https://geeksarray.com/blog/sql-server-stored-procedure-vs-user-defined-function
    2. https://www.tutorialsteacher.com/sqlserver/user-defined-functions
3. Cursor
    1. https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-cursor/
4. Index
    https://www.tutorialsteacher.com/sqlserver/indexes
5. Join
    1. https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins
6. Back Up / Restore Database
    1. https://www.globo.tech/learning-center/how-to-backup-and-restore-mssql-database/

Advance Database Programming

THANK YOU

ANY QUESTIONS ?