Installation of sql

First you need to install sql server express

Link <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

Installation is simple next next and close in the end

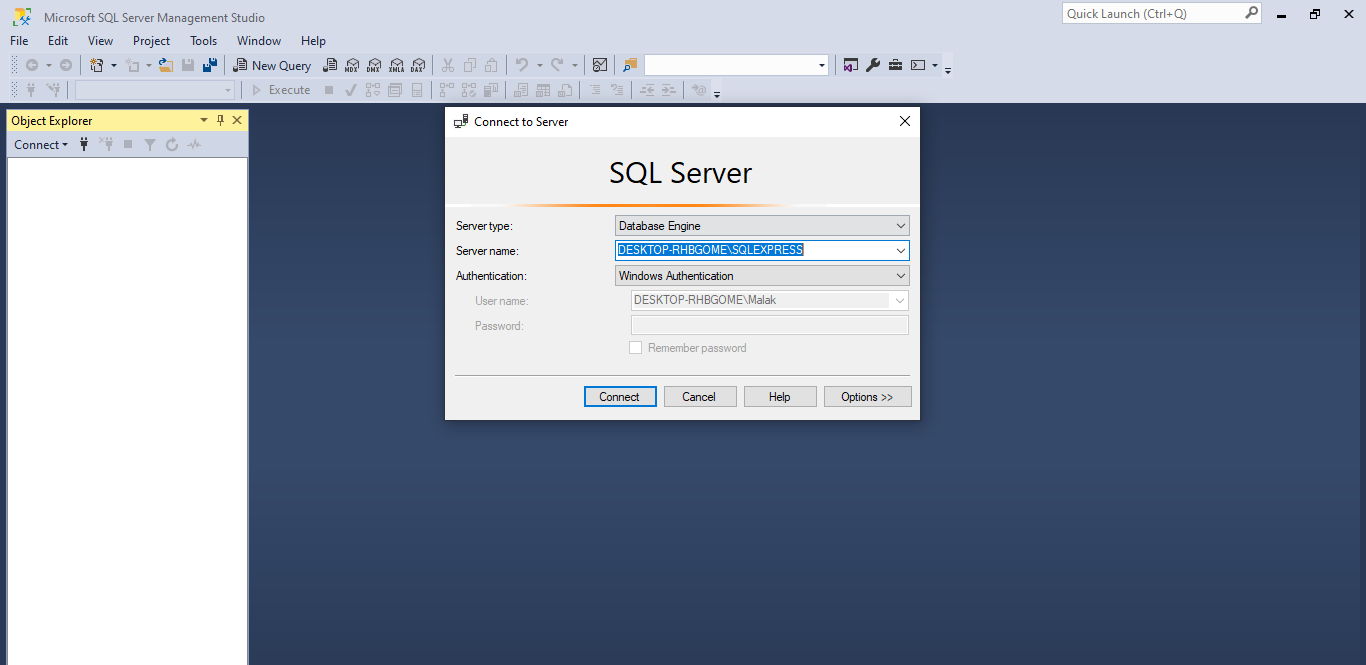
Then install sql server management

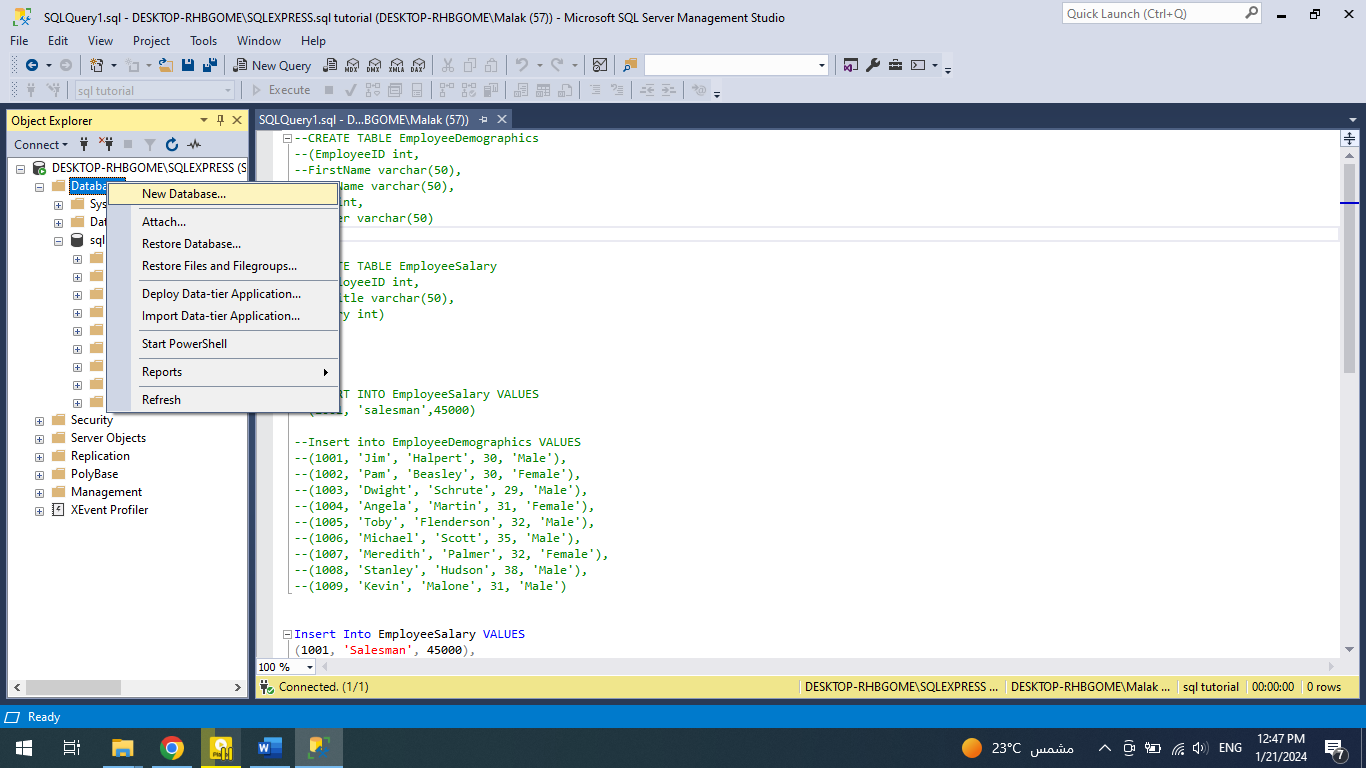
Link <https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>

It is simple installation

Note: I have these files in g/programs/ide/sql

Open Microsoft sql studio

connect

then select 

databases right click new database

enter database name then click ok

note the database you created appear in the object explorer under databases

the selected database appear in the toolbar in the left of execute button

after creating database we need to create tables we can create it using gui or using script

syntax: CREATE TABLE TABLENAME

(COULM1NAME TYPE, COULM2NAME TYPE)

EXAMPLE:

CREATE TABLE EmployeeDemographics

(EmployeeID int,

FirstName varchar(50),

LastName varchar(50),

Age int,

Gender varchar(50)

)

CREATE TABLE EmployeeSalary

(EmployeeID int,

JobTitle varchar(50),

Salary int)

Update table

Syntax: ALTER TABLE TABLENAME ADD COLUMNNAME DATATYPE

EXAMPLE:

ALTER TABLE EmployeeSalary ADD bones int;

REMOVE COLUMN

SYNTAX: ALTER TABLE TABLENAME DROP COLUMNNAME

EXAMPLE:

ALTER TABLE EmployeeSalary DROP bones

REMOVE TABLE

SYNTAX: DROP TABLE TABLENAME

EXAMPLE:

DROP TABLE EmployeeSalary;

INSERT INTO TABLES

Syntax: INSERT INTO TABLENAME VALUES

(ENTER DATA HERE AS COULMNS ODER)

EXAMPLE:

Insert into EmployeeDemographics VALUES

(1001, 'Jim', 'Halpert', 30, 'Male'),

(1002, 'Pam', 'Beasley', 30, 'Female'),

(1003, 'Dwight', 'Schrute', 29, 'Male'),

(1004, 'Angela', 'Martin', 31, 'Female'),

(1005, 'Toby', 'Flenderson', 32, 'Male'),

(1006, 'Michael', 'Scott', 35, 'Male'),

(1007, 'Meredith', 'Palmer', 32, 'Female'),

(1008, 'Stanley', 'Hudson', 38, 'Male'),

(1009, 'Kevin', 'Malone', 31, 'Male')

Insert Into EmployeeSalary VALUES

(1001, 'Salesman', 45000),

(1002, 'Receptionist', 36000),

(1003, 'Salesman', 63000),

(1004, 'Accountant', 47000),

(1005, 'HR', 50000),

(1006, 'Regional Manager', 65000),

(1007, 'Supplier Relations', 41000),

(1008, 'Salesman', 48000),

(1009, 'Accountant', 42000)

SQL Basics Tutorial For Beginners Installing SQL Server Management Studio and Create Tables 14

Select query

Select all data in table using \*

SELECT \*

FROM EmployeeDemographics

Select some coulmns like firstname and lastname

SELECT FirstName,LastName

FROM EmployeeDemographics

Select some rows using top and number of rows

SELECT top 5 FirstName,LastName

FROM EmployeeDemographics

distinct feature return unique values in specific coulmn

SELECT distinct(Gender)

FROM EmployeeDemographics

Count shows all the not null values in specific column

SELECT count(FirstName)

FROM EmployeeDemographics

Max select max value of column

SELECT max(salary)

FROM EmployeeSalary

Min is also the same as max and avg to take average

From key word

If you connect as master which is the top level database

You need to specify from feature more than normal(connect to our database sql tutorial)

SELECT avg(salary)

FROM [sql tutorial].dbo.EmployeeSalary

Where statement: used to add condition to query condition can be

|  |  |
| --- | --- |
| Condition | Meaning |
| = | Equal |
| <> | Not equal |
| > | Greater |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |

Note: comparison operator(<,>,<=,>=) work with number

Select \*

From EmployeeDemographics

Where FirstName = ‘Jim’

Not equal

Select \*

From EmployeeDemographics

Where FirstName <> ‘jim’

Select \*

From EmployeeDemographics

Where Age <= 32

We can combine more than one condition with these keyword

AND

OR

Select employee that there age greater than 32 AND Gender is Male

Select \*

From EmployeeDemographics

Where Age <= 32 AND Gender = ‘Male’

LIKE KEYWORD is used with expression

Example

Select employee that LastName start with S

Select \*

From EmployeeDemographics

Where LastName LIKE ‘S%’

% Mean any character

Null and Not Null

Null represent empty value so not null mean not empty

They are used with is keyword

Example

Select \*

From EmployeeDemographics

Where FirstName is null

Group by

It groups the data into groups

Examples

Group employee by gender and age

select Gender, Age

from Employee

group by Gender, Age;

we can improve our query to make it return count of each group

select Gender, Age, COUNT(Gender)

from Employee

group by Gender, Age;

why we use counter on gender and not on Age because it does not matter if you use any one of them

the group by feature it group the data using the two of them

if we have for example Male 30 and Male 30 and Female 30 in the data it groups them in to two group

group one Male and 30 count of employee in this group 2 and the second group contain 1 employee Female 30

order by used to order the data returned

its default is sort ASD (ascending) ASC, to sort descending DESC

select Gender, Age, COUNT(Gender)

from Employee

group by Gender, Age

order by Age DESC;

join : connect data of the two tables based on a value

if we need to connect tables, we use join, tables that are need to be connected must have column that are common on both tables this column is used to connect data of the two tables

there are different types of join

inner join: this gives us the intersection data, data that both have the connecting value (value in the connecting column)

example:

select \*

from EmployeeDemographics

inner join EmployeeSalary

on EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

this select all the columns that in the two table connecting the two table based on the Id in both of them,

full join: connect tables and the data that are not connected they have values of null

(it return the two tables combined and the data of any table that does not have data in the other table have values of null )

example:

select \*

from EmployeeDemographics

full outer join EmployeeSalary

on EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

left outer join: return the two tables combined and the date of the left table that does not have data in the right table have values of null

example:

select \*

from EmployeeDemographics

left outer join EmployeeSalary

on EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

right outer join: return the two tables combined and the data of the right table that does not have data in the left table have values of null

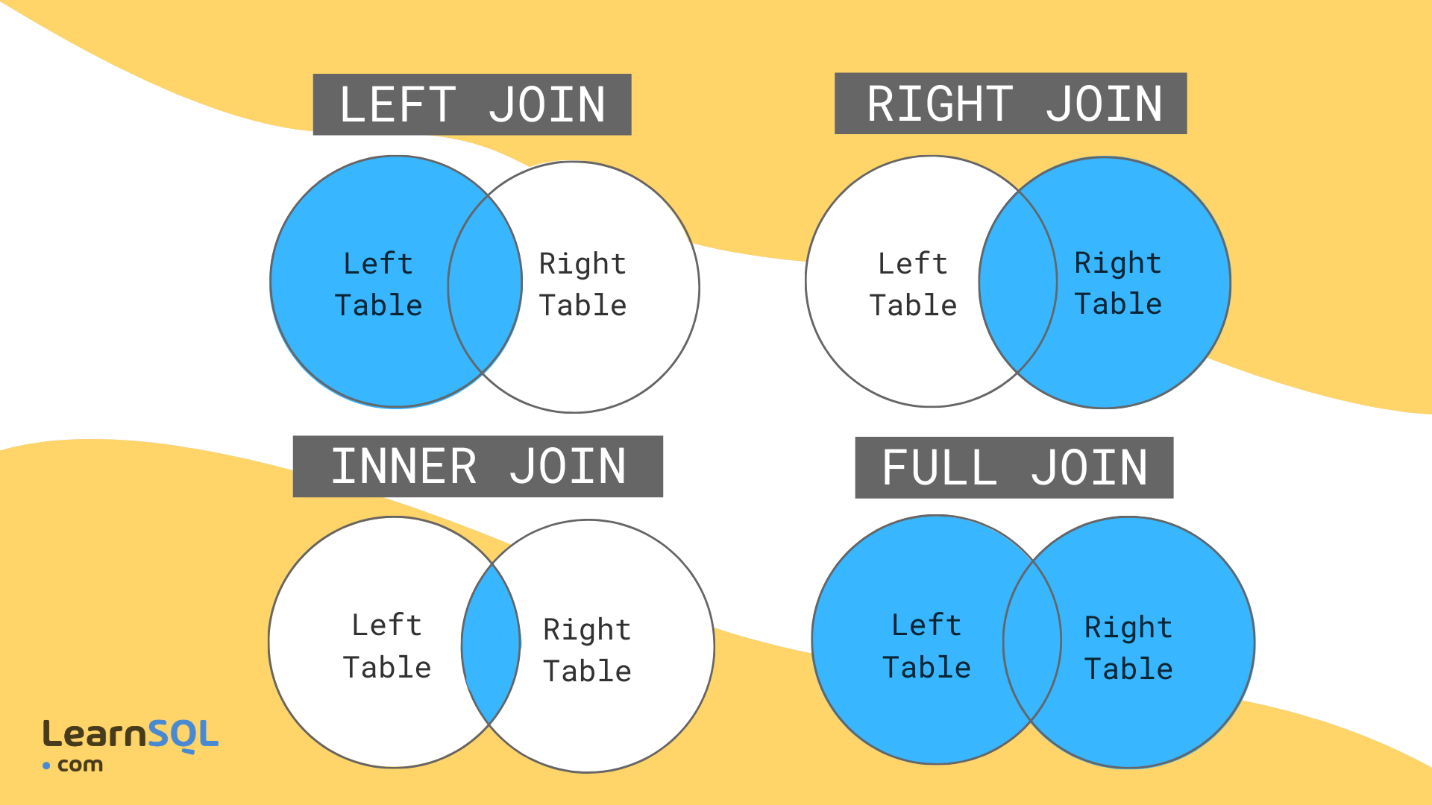
example:

select \*

from EmployeeDemographics

right outer join EmployeeSalary

on EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID



Union :is like join it combine tables to make one output it have some requirement

1. Number of columns in the two select must be the same in both
2. the columns in the two select should be the same (example first name , age should be in the two tables)

Union: combine tables without duplication

select EmployeeID

from EmployeeDemographics

union

select EmployeeID

from EmployeeSalary;

union all combine with duplication

select EmployeeID

from EmployeeDemographics

union all

select EmployeeID

from EmployeeSalary;

note: the query will not work if the column of first select is not of the same type as the column in the second select

note: the query will work but give bad output if the column in the two select are of the same type but not the same columns (for example if we select age and salary the output will be one column of name age but have salary in it)

case statement: it create new column in the output and the value are created based on some condition you are specify in when (it is like switch or if else in programming language) , it work from top to button which mean if the first condition are meet the other are ignored

select FirstName, LastName, Age,

case

when Age = 25 then 'malak'

when Age between 26 and 30 then 'young'

when Age > 30 then 'old'

else 'baby'

end

from EmployeeDemographics

where Age is not null

order by Age;

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions

Syntax:

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

*SQL aggregation is the task of collecting a set of values to return a single value. It is done with the help of aggregate functions, such as SUM, COUNT, and AVG.*

*EXAMPLE:*

select JobTitle,count(JobTitle)

from EmployeeSalary

group by JobTitle

HAVING avg(salary) > 50000

*Update query*

*Syntax:*

*Update tableName*

*Set columnName = new value*

*Where condition*

*Example:*

update EmployeeSalary

set Salary = Salary + (Salary \* 0.10)

where JobTitle = 'HR';

note: if we does not use where we will update all the records.

Note:if we need to change more than one value for a record we can specify all the columns in the set statement separated by and or ,

Delete query

Syntax:

Delete tableName

Where condition

Note condition used to specify the record that need to be deleted

Note if we does not use where we will delete our entire table

Note: before you use delete, use select \* to see what you are going to delete after you are ensure that nothing wrong modify query by replacing delete instead of select \*

*Aliases are the temporary names given to tables or columns for the purpose of a particular SQL query. It is used when the name of a column or table is used other than its original name, but the modified name is only temporary. Aliases are created to make table or column names more readable.*

*Examples:*

*We can use as keyword or space*

*select firstName as fname*

*from EmployeeDemographics;*

*use only space:*

*select firstName fname*

*from EmployeeDemographics;*

*select firstName + ' ' + LastName as fullname*

*from EmployeeDemographics;*

*select demo.FirstName, sal.JobTitle, sal.Salary*

*from EmployeeDemographics as demo*

*inner join EmployeeSalary as sal*

*on demo.EmployeeID = sal.EmployeeID*

*notes: we use aliases for more readability so we must not use alias like a, b, x and so on we need our alias to be short and meaningful*

*The “PARTITION BY” clause in SQL is used to divide a result set into partitions based on the values of one or more columns. These partitions can then be used to perform calculations and aggregate functions, such as running totals or cumulative sums, using window functions.*

*Example:*

*select distinct(JobTitle),max(Salary) over (partition by jobTitle)*

*from EmployeeSalary*

*note: this query show us the max salary in each job*

*note: it divide the table by jobtitle and for each partition (each job) it get max salary*

*note: we use distinct to return one record for each distinct job (remove repeatation)*

*cte common table expression:it is used to give a name to query , it is used to make the complex query more readable*

*example:this query select max salary for each job*

*select distinct(JobTitle),max(Salary) over (partition by jobTitle) as max\_salary from EmployeeSalary;*

*this is how we define cte*

*WITH CTE\_SA as*

*(select distinct(JobTitle),max(Salary) over (partition by jobTitle) as max\_salary from EmployeeSalary)*

*SELECT \* FROM CTE\_SA*

*Note: we do not need to write the query before cte definition*

*Note: because after the cte definition we must write select statement to view it, if the query give no named column we must give it a name*

*Temp tables: A temporary SQL table, also known as a temp table, is a table that is created and used within the context of a specific session or transaction in a database management system. It is designed to store temporary data that is needed for a short duration and does not require a permanent storage solution.*

*Example:*

*drop table if exists #temp\_jobMaxSalary*

*create table #temp\_jobMaxSalary*

*(jobTitle varchar(50),*

*jobMaxSalary int*

*)*

*insert into #temp\_jobMaxSalary*

*select distinct(JobTitle),max(Salary) over (partition by jobTitle) as max\_salary from EmployeeSalary;*

*select \**

*from #temp\_jobMaxSalary;*

*string functions: functions that apply to string*

*trim, ltrim, rtrim*

*trim: remove spaces in both sides*

*ltrim: remove spaces from the left*

*rtrim: remove spaces from the right*

*examples:*

*drop table if exists #tem\_salrayError*

*create table #tem\_salrayError*

*(id varchar(30),*

*salary varchar(30));*

*insert into #tem\_salrayError values*

*(' 1 ', '2000 ')*

*,('2',' 33200');*

*select \**

*from #tem\_salrayError;*

*select trim(id) as idtrim, trim(salary) as salarytrim*

*from #tem\_salrayError;*

*select id, LTRIM(id)*

*from #tem\_salrayError;*

*select salary, RTRIM(salary)*

*from #tem\_salrayError;*

*replace function: replace string syntax:replace(columnName,stringWeWantToRepalce, valueThatReplaceIt)*

*select salary, REPLACE(salary,'- malak','') as replacedsalary*

*from #tem\_salrayError*

*substring: return substring of string*

*syntax: substring(columnName,start,numberOfCharacter)*

*select salary, SUBSTRING(salary,1,7) as subsalary*

*from #tem\_salrayError*

*upper: return string in upper case*

*lower: return string in lower case*

*What is a****Stored Procedure****? A****stored procedure****is a prepared****SQL****code that you can save, so the code can be reused over and over again.*

*Syntax: to create stored procedure*

*Create procedure procedureName*

*As*

*Here is the procedure code, it can be any query*

*To execute the procedure: exec procedureName*

*Note: do not but exec procedureName in the procedure, the procedure create in empty query file.*

*Note: we can add parameters to procedure by add the @parameterName before as keyword and after create procedure, the parameter value can be used in anywhere in the procedure, all we need to do is to type @parameterName.*

*Also note: that if you add parameter in procedure you must pass it in exec*

*Example:*

*create procedure temp\_salary*

*as*

*create table #temp\_salary(*

*id int,*

*salary int)*

*insert into #temp\_salary values*

*(1,54123),*

*(2,3243),*

*(3,34532);*

*exec temp\_salary*

*subquery: are query in query they can be used in select, from, where*

*examples:*

*in select:*

*query: select id, jobtitle, salary,average salary of all the employee*

*select EmployeeID, JobTitle,salary, (select avg(Salary) from EmployeeSalary) as averageSalary*

*from EmployeeSalary*

*in from:*

*select \**

*from (select \* from EmployeeDemographics) as e*

*in where:*

*select EmployeeID,JobTitle,Salary*

*from EmployeeSalary*

*where EmployeeID in (*

*select EmployeeID*

*from EmployeeDemographics*

*where Age > 30*

*)*