

BEGINNER:

--------------

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,

)

INSERT INTO EmployeeDemographics VALUES

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

INSERT INTO EmployeeDemographics VALUES

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

(1003, 'Cam','Halpt', 41, 'Male'),

(1004, 'Hem','Haert', 30, 'Female'),

(1005, 'Leam','Halrt', 30, 'Male'),

(1006, 'Tom','Halpe', 29, 'Male'),

(1007, 'Bom','Halt', 30, 'Female'),

(1008, 'Kevin','Hal', 30, 'Male'),

(1009, 'Tem','Hulprt', 32, 'Female')

INSERT INTO EmployeeSalary VALUES

(1001, 'Salesman', 45000),

(1002, 'Salesman', 55000),

(1003, 'Salesman', 25000),

(1004, 'Salesman', 35000),

(1005, 'HR', 40000),

(1006, 'Salesman', 60000),

(1007, 'Acountant', 42000),

(1008, 'Salesman', 41000),

(1009, 'Salesman', 43000)

/\* where Statement

=, <>, <, > And, Or, Like, Null, Not Null, In

\*/

SELECT \*

FROM EmployeeDemographics

--WHERE Age <= 32 OR Gender = 'Female'

WHERE LastName LIKE 'H%A%E' -- S% S İLE BAŞLAYANLARI BULUR %S% İÇİNDE S GEÇENLERİ BULUR

-- %S S İLE BİTENLERİ BULUR

-- H%A%E H İLE BAŞLAYIP İÇİNDE A VE E OLANLARI BULUR

SELECT \*

FROM EmployeeDemographics -- IF YOU USE NULL : TABLONDA NULL VARSA ONLARI BULUR

WHERE FirstName is NOT NULL --IF YOU USE NOT NULL: TABLONDA NULL YOKSA ONLARI BULUR

SELECT \*

FROM EmployeeDemographics

WHERE FirstName= 'Jim' AND FirstName='Micel' -- yerine IN kullanarak çünkü = multiple things için geçerli

WHERE FirstName IN ('jim', 'Micel' )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT Gender, Age, COUNT(Gender) -- SELECT DISTINCT ( Gender) yazılırsa olan cinsiyetlerin çeşidi kadar gösterilir

FROM EmployeeDemographics -- SELECT Gender yazılırsa olan cinsiyetleri tablo sırasına göre tek tek gösterir

GROUP BY Gender, Age -- SELECT Gender, COUNT (Gender) yazılırsa hangi cinsiyetten ne kadar olduğunu gösterir

-- GROUP BY DA YAZILANLARIN TABLOSU GÖSTERİLİR COUNT SADECE ÇAĞIRICIDIR.

-- YANİ COUNT UN İÇİNE YAZDIĞIMIZ ÖZELLİK ASLINDA ÇAĞIRILAN ÖZELLİK DEĞİLDİR GÖSTERGEDİR.

-- BU KODUN AYNISINI COUNT (GENDER) YERİNE COUNT(AGE) İLE YAPTIPIMIZDA DA AYNI TABLO ELDE EDİLİR ( DENENDİ- ONAYLANDI)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Group By, Order By

\*/

SELECT \*

FROM EmployeeDemographics

SELECT Gender, COUNT(Gender) AS CountGender

FROM EmployeeDemographics

WHERE Age > 31

GROUP BY Gender

ORDER BY CountGender DESC

SELECT \*

FROM EmployeeDemographics

ORDER BY Age DESC, Gender DESC -- order by age kullanımında yaşa göre sıralamada küçük yaştan büyüğe doğru sıralanma olacak

-- order by age desc kullanımındaysa büyükte küçüğe sıralama yapacaktır.

//

SELECT Age, COUNT (Age) AS CountAge

FROM EmployeeDemographics

WHERE Age >30 ===> Yaşları 30 dan büyük olanların küçükten büyüğe sıralaması ve kaç kişinin bu gruplarda olduğunun tablosunu verir ancak cinsiyetler yok

GROUP BY Age Cinsiyetlerin de olduğu bir tablo için;

ORDER BY CountAge desc

//

SELECT \*

FROM EmployeeDemographics

ORDER BY 4 DESC, 5 DESC -- BU KONUM SATIRI SIRALAMAYI DEĞİŞTİRİR.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INTERMIDIATE:

-------------

Joıns

Unıons

Case Statements

Updating/Deleting Data

Partition By

Data Types

Aliasing

Creating Views

Having vs Group By Statement

GETDATE()

Primary Key vs Foreign Key

/\*

Inner Joins, Full/left/right outher joins

\*/

SELECT \*

FROM SQLTUTORIAL.dbo.EmployeeDemographics

SELECT \*

FROM SQLTUTORIAL.dbo.EmployeeSalary

SELECT \* -- \*'ın anlamı her şey demek -- bu kod iki tabloyu birleştirerek beraber gösterir

FROM SQLTUTORIAL.dbo.EmployeeDemographics -- kesişimleri /benzer olanları gösterir.

Inner Join SQLTUTORIAL.dbo.EmployeeSalary

ON EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

\*\*\*\*\*\*\*\*

SELECT \*

FROM SQLTUTORIAL.dbo.EmployeeDemographics

SELECT \*

FROM SQLTUTORIAL.dbo.EmployeeSalary

SELECT \*

FROM SQLTUTORIAL.dbo.EmployeeDemographics -- employeıd ye göre sıralanmış olarak her şeyi gösterir

Full outher Joins SQLTUTORIAL.dbo.EmployeeSalary

ON EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

Right outher joins ==> Right table ı alacak kesişimleri de dösterecek ama sadece Left table da olup right tableda olanı göstermeyecek

Left outher joins ==> left table ı alacak kesişimleri de dösterecek ama sadece right table da olup left tableda olanı göstermeyecek

SELECT EmplooyeSalary, EmployeeID, FirstName, LastName, JobTitle, Salary

FROM SQLTUTORIAL.dbo.EmployeeDemographics

Inner join SQLTUTORIAL.dbo.EmployeeSalary -- ınner join bize ortak olanları gösterir

ON EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

-- ınner join yerine right outer join olsaydı sadece right table da olanları gösterir bu sefer kesişimi de göstermezdi

--> left outer join olsaydı right için yaptığını left için yapıyor olurdu

SELECT EmployeeDemographics, EmployeeID, FirstName, LastName, Salary

FROM SQLTUTORIAL.dbo.EmployeeDemographics

Inner join SQLTUTORIAL.dbo.EmployeeSalary

ON EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

WHERE FirstNAME <> 'Micheal' -- Micheal isimli kişiyi listeden çıkarır.

ORDER BY Salary DESC -- it starts with top salary an show them on the table

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT JobTitle, AVG (Salary)

FROM SQLTUTORIAL.dbo.EmployeeDemographics

Inner join SQLTUTORIAL.dbo.EmployeeSalary

ON EmployeeDemographics.EmployeeID = EmployeeSalary.EmployeeID

WHERE JobTitle='Salesman' -- iş tanımı salesman olanlar için ortalama maaş gösterimi yaptı

GROUP BY jobtitle

/\* -- join de union a iki tabloyu birleştirim kombine bir çıktı oluşturuyor

Union, Union All -- Union bu çıktıların verilerini ayrı ayrı yapmak yerine aynı tablo üzerinde listeleyerek gösterir.

\*/ -- join bu çıktıları birleştirmesine rağmen çıktıyı yan yana iki tablo olarak gösteriyordu

-- Bu durum join ve union kullanım ayrımını veriri

SELECT \*

FROM SQLTUTORIAL.dbo.EmployeeDemographics -- UNION kullanımda tabloda kesişim (aynı olan çıktılar) tek bir çıktı olarak listede gösterilir.

UNION

SELECT \*

FROM SQLTUTORIAL.dbo.WareHouseEmployeeDemographics

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT \* -- UNION ALL kullanımında ise kesişimleri bir göstermek yerine

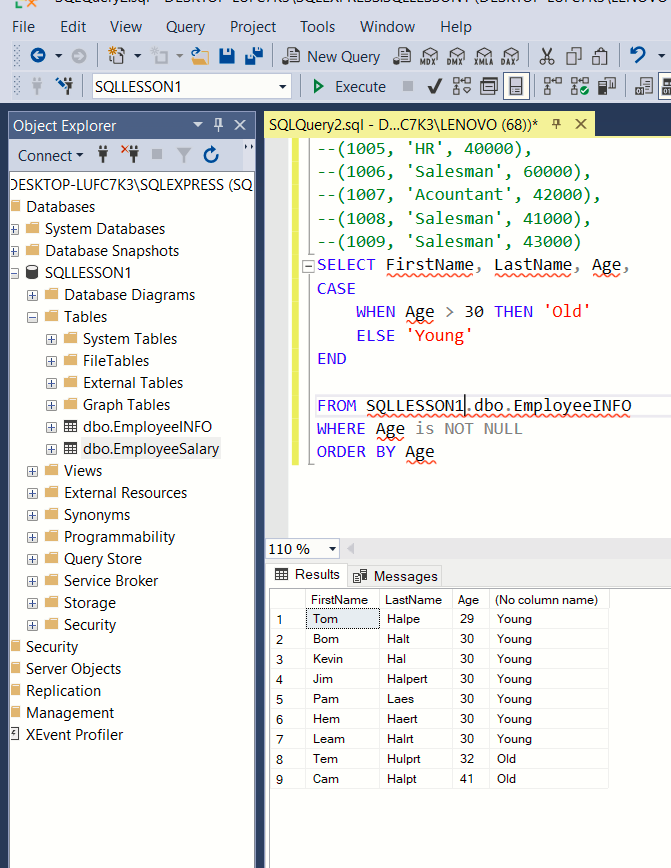
FROM SQLTUTORIAL.dbo.EmployeeDemographics --her ne kadar aynı olsa da çıktıda hepsini aynı anda gösterir.

UNION ALL

SELECT \*

FROM SQLTUTORIAL.dbo.WareHouseEmployeeDemographics

ORDER BY EmployeeID



/\*

Case Statement

\*/

SELECT FirstName, LastName, Age

CASE

WHEN Age > 30 THEN 'Old'

ELSE 'Young'

END

FROM SQLTUTORIAL.dbo.EmployeeDemographics

WHERE Age is NOT NULL

ORDER BY Age

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT FirstName, LastName, Age,

CASE

WHEN Age = 41 THEN 'Stanley'

WHEN Age > 30 THEN 'Old'

ELSE 'Baby'

END

***///////////BURDAN İTİBAREN SQLTUTORIAL YERİNE SQLLESSON1 OLUŞTURULMUŞ AMA İKİSİNDE DE AYNI VERİLER KULLANILMIŞTIR. ////////////////////////////////////////////////////***

FROM SQLLESSON1.dbo.EmployeeINFO

WHERE Age is NOT NULL

ORDER BY Age

\*\*\*\*\*\*\*\*\*\*

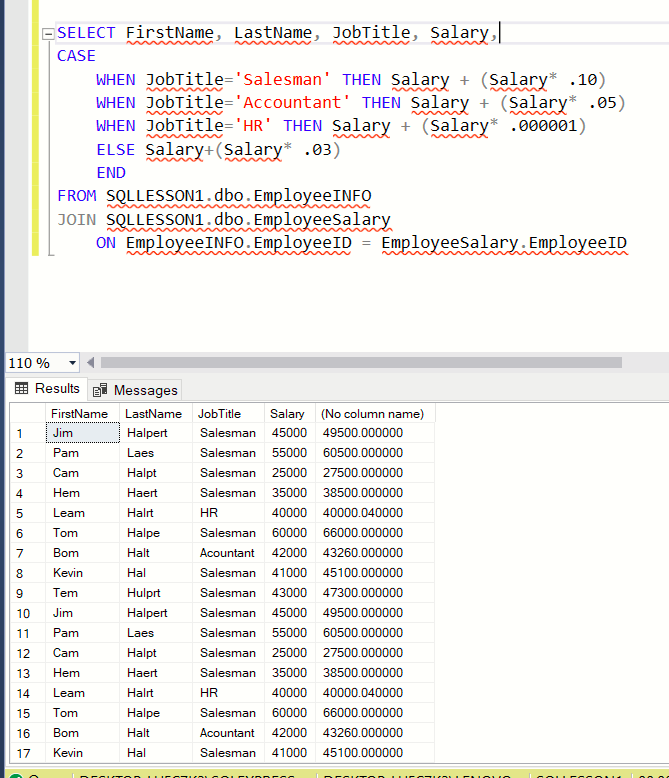
SELECT FirstName, LastName, JobTitle, Salary

FROM SQLLESSON1.dbo.EmployeeINFO

JOIN SQLLESSON1.dbo.EmployeeSalary

ON EmployeeINFO.EmployeeID = EmployeeSalary.EmployeeID

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 🡺 🡺 MAAŞLARA ZAM YAPILDI

SELECT FirstName, LastName, JobTitle, Salary,

CASE

WHEN JobTitle='Salesman' THEN Salary + (Salary\* .10)

WHEN JobTitle='Accountant' THEN Salary + (Salary\* .05)

WHEN JobTitle='HR' THEN Salary + (Salary\* .000001)

ELSE Salary+(Salary\* .03)

END AS SalaryAfterRaise

FROM SQLLESSON1.dbo.EmployeeINFO

JOIN SQLLESSON1.dbo.EmployeeSalary

ON EmployeeINFO.EmployeeID = EmployeeSalary.EmployeeID

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Having Clause

\*/

SELECT JobTitle, COUNT(JobTitle)

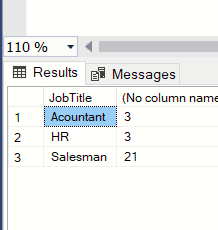
FROM SQLLESSON1.dbo.EmployeeINFO

JOIN SQLLESSON1.dbo.EmployeeSalary

ON EmployeeINFO.EmployeeID = EmployeeSalary.EmployeeID

GROUP BY JobTitle

🡺 ÇIKTI



SELECT JobTitle, COUNT(JobTitle)

FROM SQLLESSON1.dbo.EmployeeINFO

JOIN SQLLESSON1.dbo.EmployeeSalary

ON EmployeeINFO.EmployeeID = EmployeeSalary.EmployeeID

GROUP BY JobTitle

HAVING AVG(Salary) > 45000

ORDER BY AVG(Salary)

/\*

Updating / Deleting Data -- YENİ ŞEYLER EKLEMEM ÖNCEDEN OLUŞMUŞU YENİLEMEK VE ÇIKARILMAK

\*/ -- İSTENENLERİN YAPILDIĞI KODLAMADIR

SELECT \*

FROM SQLLESSON1.dbo.EmployeeINFO

UPDATE SQLLESSON1.dbo.EmployeeINFO

SET EmployeeID =1008

WHERE FirstName = 'Jim' AND LastName = 'Halpert'

SELECT \*

FROM SQLLESSON1.dbo.EmployeeINFO

UPDATE SQLLESSON1.dbo.EmployeeINFO

SET Age= 31, Gender='Female'

WHERE FirstName = 'Jim' AND LastName = 'Halpert'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT \*

FROM SQLLESSON1.dbo.EmployeeINFO

UPDATE SQLLESSON1.dbo.EmployeeINFO

SET Age= 31, Gender='Female'

WHERE EmployeeID=1005

DELETE FROM SQLLESSON1.dbo.EmployeeINFO

WHERE EmployeeID=1007

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Aliasing -- özellikle takım çalışmalarında , anlaşılmanın kolay olması nerden neyin geldiğinin anlaşılmasının kolaylığını sağlar

-- karmaşık tablolarda daha organize ve daha kolay okuma sağlar

\*/

SELECT FirstName + ' ' + LastName AS FullName

FROM [SQLLESSON1].[dbo].[EmployeeINFO]

\*\*\*\*\*\*\*\*\*

SELECT AVG(Age) AS AvgAge

FROM [SQLLESSON1].[dbo].[EmployeeINFO]

\*\*\*\*\*\*\*\*\*

SELECT Demo.EmployeeID, Sal.Salary

FROM [SQLLESSON1].[dbo].[EmployeeINFO] AS Demo

join [SQLLESSON1].[dbo].[EmployeeSalary] AS Sal

on Demo.EmployeeID = Sal.EmployeeID

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECT INFO.EmployeeID, INFO.FirstName, INFO.FirstName,

Sal.JobTitle, Ware.Age -- bu kodda warehouse kısmında ve ware age kısmında sıkıntı çıktı

SELECT INFO.EmployeeID, Sal.Salary

FROM [SQLLESSON1].[dbo].EmployeeINFO a

left join [SQLLESSON1].[dbo].EmployeeSalary b

on a.EmployeeID=b.EmployeeID

left join [SQLLESSON1].[dbo].WareHouseEmployeeINFO c

on a.EmployeeID = c.EmployeeID

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Partition By

\*/

SELECT FirstName, LastName, Gender, Salary,

COUNT(Gender) OVER ( PARTITION BY Gender) as TotalGender

FROM SQLLESSON1.dbo.EmployeeINFO INFO

join SQLLESSON1.dbo.EmployeeSalary sal

on INFO.EmployeeID = sal.EmployeeID

\*\* AYNISININ PARTITION BY YERİNE GROUP BY KODU

SELECT FirstName, LastName, Gender, Salary,COUNT(Gender)

FROM SQLLESSON1.dbo.EmployeeINFO INFO

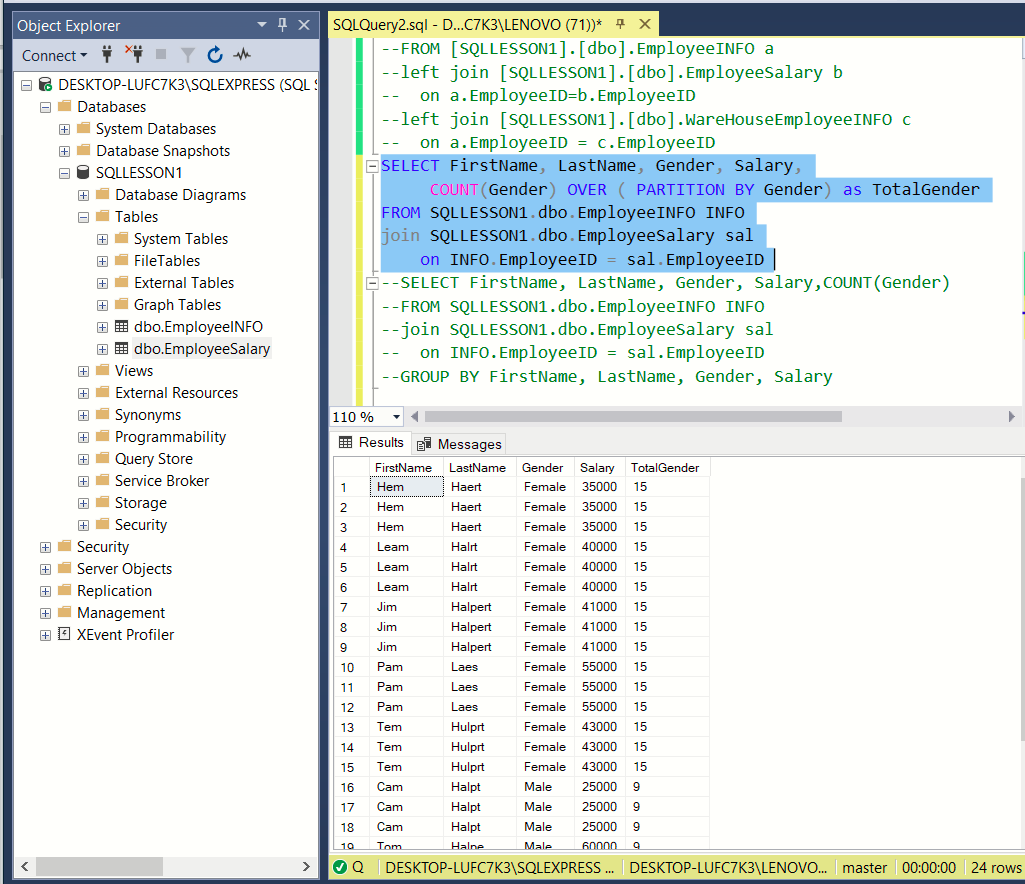
join SQLLESSON1.dbo.EmployeeSalary sal

on INFO.EmployeeID = sal.EmployeeID

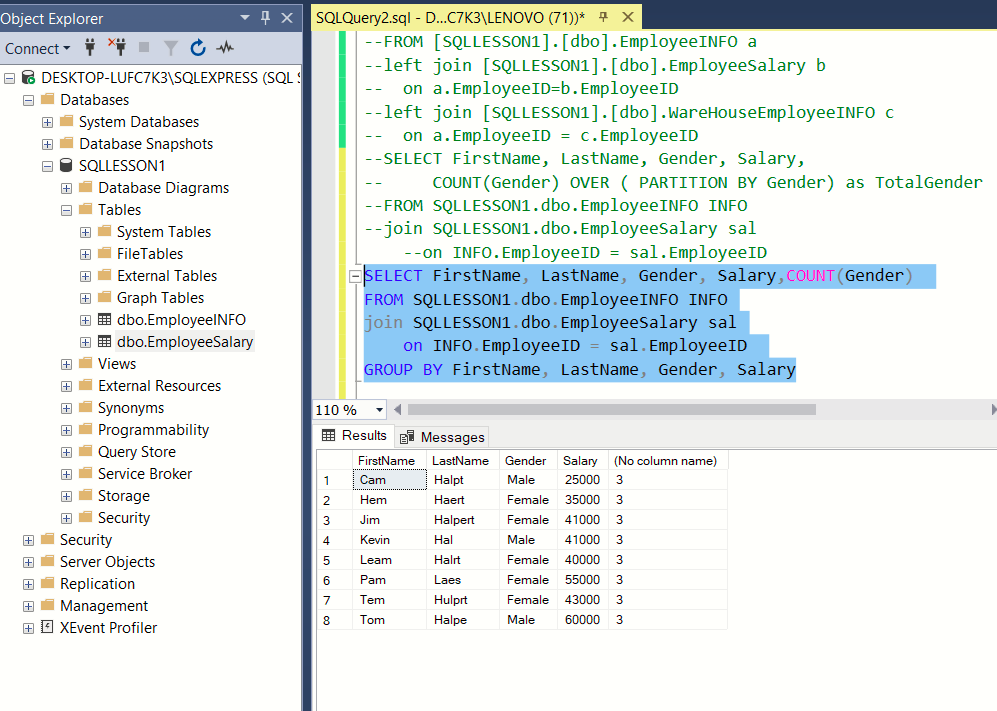
GROUP BY FirstName, LastName, Gender, Salary

--------------------------------ÇIKTI KARŞILAŞTIRMALARI------------------------------------------------------------

* PARTITION BY KULLANIMI



* GROUP BY KULLANIMI



ADVENCED:

----------

CTEs

SYS Tables

Subqueries

Temp Tables

String Functions (TRIM, LTRIM, RTRIM, Replace, Substring, Upper, Lower)

Regular Expression

Stored Procedures

Importing Data from different File Types/Sources

Exporting Data to different File Types

/\*

CTEs

\*/

WITH CTE\_Employee as

(SELECT FirstName, LastName, Gender, Salary,

COUNT (Gender) OVER (PARTITION BY Gender) as TotalGender,

AVG (Salary) OVER (PARTITION BY Gender) as AvgSalary

FROM SQLLESSON1.dbo.EmployeeINFO emp

join SQLLESSON1.dbo.EmployeeSalary sal

ON emp.EmployeeID = sal.EmployeeID

WHERE Salary >'40000'

)

SELECT \*

FROM CTE\_Employee

/\*

Temp Tables --VAR OLAN TABLODAN YENİ TABLO OLUŞTURMAYI SAĞLAR.

-- EĞER OLUŞAN TABLODA OLUŞTURULMAK İSTENEN DEĞER VE TABLO ZATEN VARSA ONU SİLER VE YERİNE YENİSİNİ OLUŞTURUR.

\*/

CRATE TABLE #temp\_Employee

(

employeeID int,

JobTitle varchar(100),

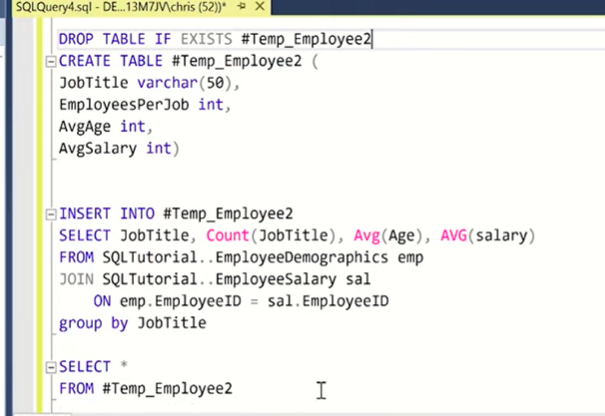
Salary int

)

INSERT INTO #temp\_Employee VALUES

SELECT\*

FROM SQLLESSON1..employeeSalary



/\*

Today's Topic: String Functions - TRIM, LTRIM, RTRIM, Replace, Substring, Upper, Lower

\*/

--Drop Table EmployeeErrors;

CREATE TABLE EmployeeErrors (

EmployeeID varchar(50)

,FirstName varchar(50)

,LastName varchar(50)

)

Insert into EmployeeErrors Values

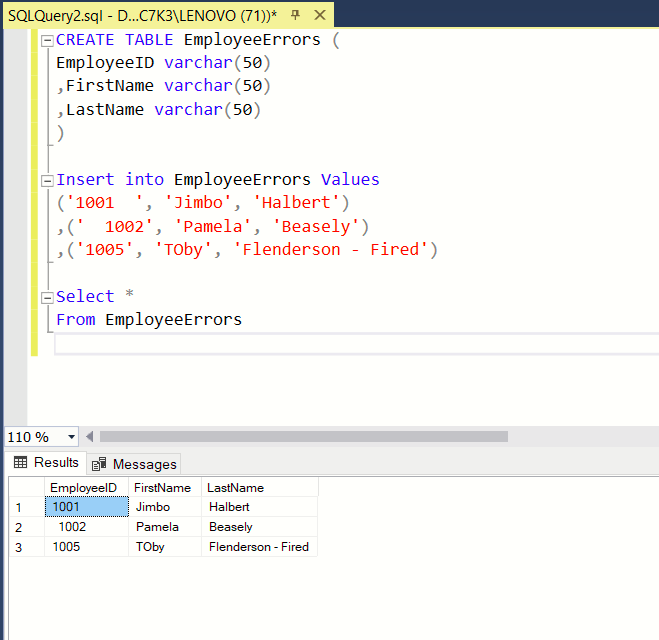
('1001 ', 'Jimbo', 'Halbert')

,(' 1002', 'Pamela', 'Beasely')

,('1005', 'TOby', 'Flenderson - Fired')

Select \*

From EmployeeErrors



Select EmployeeID, TRIM(employeeID) AS IDTRIM

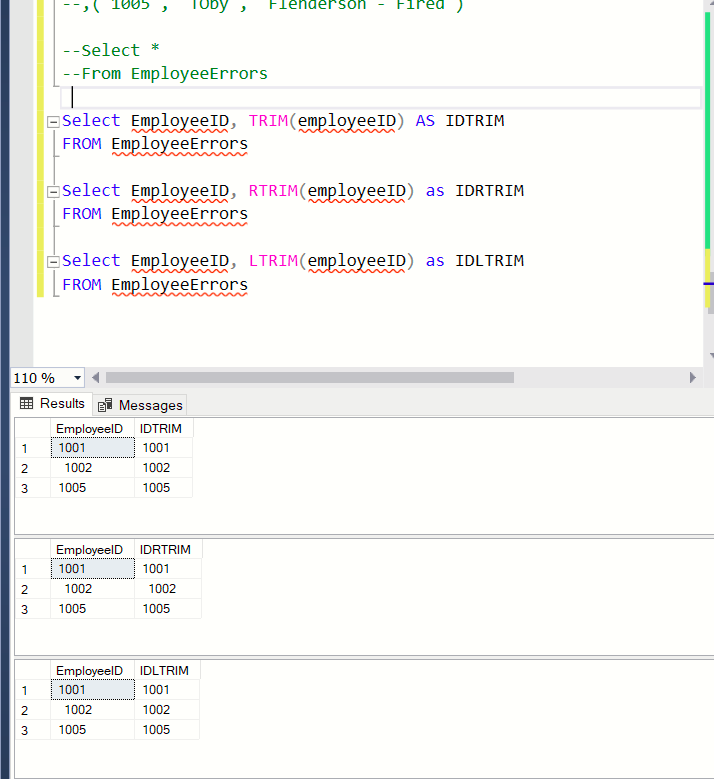
FROM EmployeeErrors

Select EmployeeID, RTRIM(employeeID) as IDRTRIM

FROM EmployeeErrors

Select EmployeeID, LTRIM(employeeID) as IDLTRIM

FROM EmployeeErrors



-- Using Replace

Select LastName, REPLACE(LastName, '- Fired', '') as LastNameFixed

FROM EmployeeErrors

-- Using Substring

Select Substring(err.FirstName,1,3), Substring(dem.FirstName,1,3), Substring(err.LastName,1,3), Substring(dem.LastName,1,3)

FROM EmployeeErrors err

JOIN EmployeeDemographics dem

on Substring(err.FirstName,1,3) = Substring(dem.FirstName,1,3)

and Substring(err.LastName,1,3) = Substring(dem.LastName,1,3)

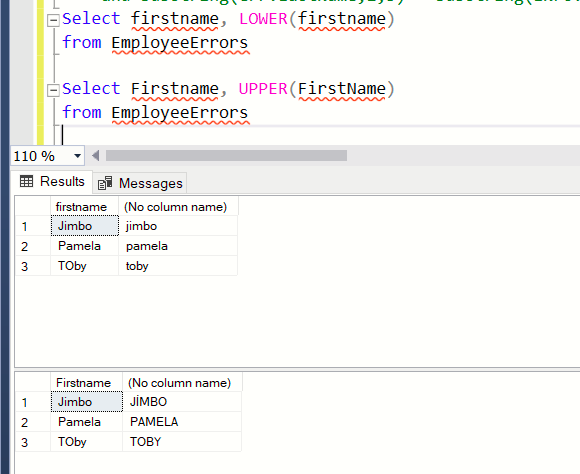
-- Using UPPER and lower

Select firstname, LOWER(firstname)

from EmployeeErrors

Select Firstname, UPPER(FirstName)

from EmployeeErrors



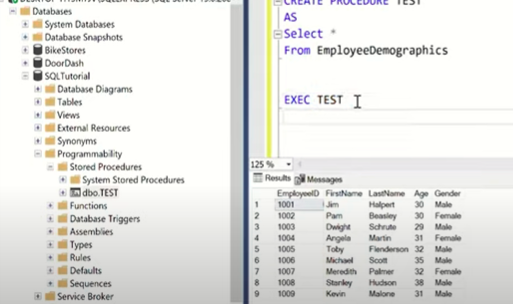
CREATE PROCEDURE TEST

AS

SELECT \* -- BU KODU YAZDIKTAN SONRA EXECUTE YAPIP

FROM EmployeeINFO -- SQLLESSON1 İÇİNDEKİ "PROGRAMMABILITY İÇİNDE STORED PROCEDURES

-- DOSYASINI REFRESH YAPMAK VE BÖYLE CE DOSYA OLUŞMUŞ OLACAKTIR.



/\*

Today's Topic: Stored Procedures

\*/

CREATE PROCEDURE TEST

AS

SELECT \* -- BU KODU YAZDIKTAN SONRA EXECUTE YAPIP

FROM EmployeeINFO -- SQLLESSON1 İÇİNDEKİ "PROGRAMMABILITY İÇİNDE STORED PROCEDURES

-- DOSYASINI REFRESH YAPMAK VE BÖYLE CE DOSYA OLUŞMUŞ OLACAKTIR.

EXEC TEST

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CREATE PROCEDURE Temp\_Employee

AS

DROP TABLE IF EXISTS #temp\_employee

Create table #temp\_employee (

JobTitle varchar(100),

EmployeesPerJob int ,

AvgAge int,

AvgSalary int

)

Insert into #temp\_employee

SELECT JobTitle, Count(JobTitle), Avg(Age), AVG(salary)

FROM SQLTutorial..EmployeeDemographics emp

JOIN SQLTutorial..EmployeeSalary sal

ON emp.EmployeeID = sal.EmployeeID

group by JobTitle

Select \*

From #temp\_employee

GO;

--ALTER PROCEDURE

CREATE PROCEDURE Temp\_Employee2

@JobTitle nvarchar(100)

AS

DROP TABLE IF EXISTS #temp\_employee3

Create table #temp\_employee3 (

JobTitle varchar(100),

EmployeesPerJob int ,

AvgAge int,

AvgSalary int

)

Insert into #temp\_employee3

SELECT JobTitle, Count(JobTitle), Avg(Age), AVG(salary)

FROM SQLTutorial..EmployeeDemographics emp

JOIN SQLTutorial..EmployeeSalary sal

ON emp.EmployeeID = sal.EmployeeID

where JobTitle = @JobTitle --- make sure to change this in this script from original above

group by JobTitle

Select \*

From #temp\_employee3

GO;

exec Temp\_Employee2 @jobtitle = 'Salesman'

exec Temp\_Employee2 @jobtitle = 'Accountant'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Today's Topic: Subqueries (in the Select, From, and Where Statement)

\*/

Select EmployeeID, JobTitle, Salary

From EmployeeSalary

-- Subquery in Select

Select EmployeeID, Salary, (Select AVG(Salary) From EmployeeSalary) as AllAvgSalary

From EmployeeSalary

-- How to do it with Partition By

Select EmployeeID, Salary, AVG(Salary) over () as AllAvgSalary

From EmployeeSalary

-- Why Group By doesn't work

Select EmployeeID, Salary, AVG(Salary) as AllAvgSalary

From EmployeeSalary

Group By EmployeeID, Salary

order by EmployeeID

-- Subquery in From

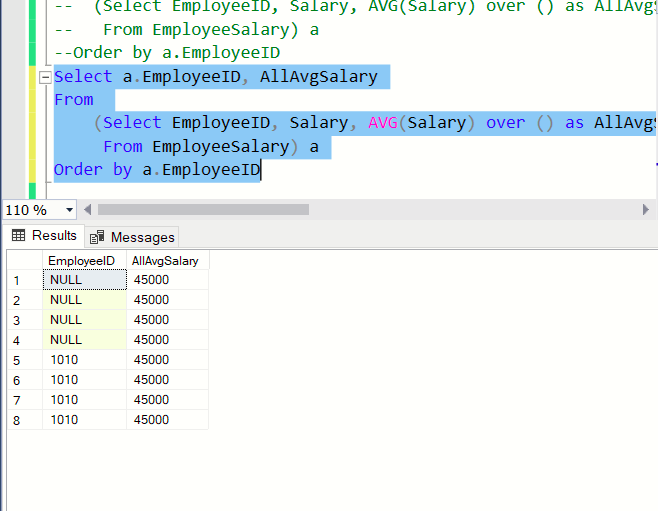
Select a.EmployeeID, AllAvgSalary

From

(Select EmployeeID, Salary, AVG(Salary) over () as AllAvgSalary

From EmployeeSalary) a

Order by a.EmployeeID



-- Subquery in Where

Select EmployeeID, JobTitle, Salary

From EmployeeSalary

where EmployeeID in (

Select EmployeeID

From EmployeeDemographics

where Age > 30)