LEN - Проверява дължината като string

DATALENGTH - Проверява дължината като байтове

* Създаване на база данни:

CREATE DATABASE Minions

* Създаване на таблица:

CREATE TABLE People

(

ColumnName DataType Constraints

Id INT NOT NULL PRIMARY KEY IDENTITY (UNIQUE, DEFAULT, CHECK)

)

* Взимане на цялата информация от тамблица:

SELECT \* FROM People

* Взимане на част от информацията от таблица:

SELECT TOP (5) FirstName, LastName

FROM Minions

* Промяна на таблица:

ALTER TABLE Minions

* Добавяне на нова колона в таблица:

ALTER TABLE Minions

ADD Weight DECIMAL(10,2)

ADD ColumnName DataType

* Изтриване на колона от таблица:

ALTER TABLE Minions

DROP COLUMN Name

* Променяне на типа данни на колона:

ALTER TABLE Minions

ALTER COLUMN Weight int

ALTER COLUMN ColumnName NewDataType

* Добавяне на Primary Key към съществуваща колона:

ALTER TABLE Minions

ADD CONSTRAINT PK\_Id

PRIMARY KEY (ColumnName1, ColumnName2….)

* Добавяне на ограничение Unique:

ALTER TABLE Minions

ADD CONSTRAINT UQ\_Email

UNIQUE (ColName1, ColName2…)

* Задаване на дифоутна стойност:

ALTER TABLE People

ALTER COLUMN Balance

SET DEFAULT 0

* Добавяне на CHECK ограничение:

ALTER TABLE Minions

ADD CONSTRAINT CheckValue

CHECK (Age > 20) / CHECK (len(Name) > 5) / CHECK (Status in (‘true’, ‘false’))

* Изтриване на всички записи в таблица:

TRUNCATE TABLE Minions

* Изтриване на таблица:

DROP TABLE Minions

* Изтриване на базата данни:

DROP DATABASE Minions

* Изтриване на ограничение от таблица:

ALTER TABLE Minions

DROP CONSTRAINT PK\_Id

* Премахване на дифоутна стойност:

ALTER TABLE People

ALTER COLUMN Balance

DROP DEFAULT

* Избиране на всичко от таблица:

SELECT \* FROM TableName

* Избиране на определени колони от таблица:

SELECT ColumnName1, ColumnName2 … FROM TableName

* Преименуване на колона:

SELECT ColumnName AS NewColumnName FROM TableName

SELECT c.Duration,

c.ACG AS ‘Access Control Gateway’

FROM Calls AS c

* Свързване на колони или тест:

SELECT FirstName + ‘ ’ + LastName AS [Full Name],

EmployeeId as [No.]

FROM Employees

* Избиране само на уникалните стойности:

SELECT DISTINCT ColumnName FROM TableName

* Филтриране на редовете по даден критерий:

SELECT ColumnName1, ColumnName2 ….

FROM TableName

WHERE ColumnName1 =/</>… something

SELECT ColumnName FROM TableName

WHERE NOT (ColumnName = smth OR ColumnName = smth)

SELECT ColumnName1, ColumnName2 FROM TableName

WHERE ColumnName BETWEEN smth AND smth

SELECT ColumnName1, ColumnName 2 FROM TableName

WHERE ColumnName IN/NOT IN (smth, smth, smth)

WHERE ColumnName IS/IS NOT NULL

* Сортиране на данни:

SELECT ColumnName1, ColumnName2

FROM TableName

ORDER BY ColumnName ASC/DESC

ORDER BY NEWID() - генерира нови уникални id-та

* Създаване на VIEW:

CREATE VIEW v\_NAME AS

SELECT FirstName + ‘ ’ LastName AS [Full Name], Salary

FROM Employees ORDER BY DepartmentId

* Задаване на друг формат на дата:

FORMAT(Now(), ‘DD-MM-YYYY’) AS {Date}

* Писане на данни в таблица:

INSERT INTO TableName (ColumnName1, ColumnName2….)

VALUES (smth, smth….),

(smth, smth…..)

* Създаване на нова таблица от съществуваща таблица: !!!!!!!!!!

SELECT ColumnName1, ColumnName2, ColumnName3…

INTO NewTableName

FROM ExistingTableName

* Записване на данни от съществуваща в съществуваща таблица:

!!!!!!!!!!!!!

INSERT INTO TableName1(ColumnName1, ColumnNamme2 ….)

SELECT ColumnName1, ColumnName2….

FROM TableName2

* Създаване на SEQUENCE:

CREATE SEQUENCE seq\_TableName\_ColumnName

AS int

START WITH 1

INCREMENT BY 1

SELECT NEXT VALUE FOR seq\_TableName\_ColumnName

* Изтриване на определен ред от таблица:

DELETE FROM TableName WHERE Id = 1

* Ъпдейтване на информация:

UPDATE TableName

SET ColumnName = ‘something’

WHERE Id = 1

* Логическа конструкция CASE:

CASE

WHEN Name = ‘P’ THEN ‘D’

WHEN Name = ‘T’ THEN ‘S’

ELSE ‘K’

END

* Вградена функция ISNULL: !!!!!!!!!!!!!!!!!!

ISNULL(ColumnName, NewValue)

* Свързване на стрингове:

SELECT FirstName + ‘ ’ + LastName - Ако едно от тях е NULL,

AS [Full Name] всичко става NULL

FROM Employees

SELECT CONCAT(FirstName, ‘ ’, LastName) - Ако едно от тях е

AS [Full Name] NULL, го заменя с празен стринг

FROM Employees

* Взимане на част от стринг:

SUBSTRING(String, StartIndex, Length)

* Заменяне на един стринг с друг:

REPLACE(String, Pattern, Replacement)

SELECT REPLACE(Title, ‘blood’, ‘\*\*\*\*\*’)

AS Title

FROM Album

* Премахване на празните места от двете страни на стринг:

LTRIM(String)

RTRIM(String)

* Намиране дължината на стринг:

LEN(String)

* Намиране броя на използваните байтове:

DATALENGTH(String)

* Взимане на символи от началото или от края на стринг:

LEFT(String, Count)

RIGHT(String, Count)

* Превръщане на стринг с малки или с големи букви:

UPPER(String)

LOWER(String)

* Обръщане на стринг:

REVERSE(String)

* Повтаряне на стринг:

REPLICATE(String, Count)

* Къде се намира част от стринг:

CHARINDEX(Pattern, String, [StartIndex]) - StartIndex може да няма

* Вмъкване на събстинг на определена позиция:

STUFF(String, StartIndex, Length, Substring) - Length задава колко

букви да изтрием от

оригиналния стринг

* Числото Пи:

SELECT PI()

* Намиране на абсолютна стойност:

ABS(Value)

* Намиране на корен квадратен:

SQRT(Value)

* Повдигане на втора степен:

SQUARE(Value)

* Повдигане на степен:

POWER(Value, Exponent)

* Закръгляне на число:

ROUND(Value, Precision)

* Превръщане от един тип данни в дуг:

CAST(Data AS NewType)

CONVERT(NewType, Data)

FORMAT(DATETIME, 'yyyy-MM-dd')

* Връщане на най-близкото цяло число:

FLOOR(Value)

CEILING(Value)

* Проверяване знака на число:

SIGN(Value) - Връща 1/ -1 / 0

* Връщане на случайно число:

RAND() - връща число между 0 и 1

RAND(Seed) - връща една и съща стойност

RAND() \* 100 - връща число между 0 и 100

* Взимане на част от дата:

DATEPART(Part, Date)

(year, yyyy, yy, YEAR(Date), month, mm, m, MONTH(Date), day, dd, d, DAY(Date))

* Намиране на разликата между две дати:

DATEDIFF(Part, FirstDate, SecondDate)

* Връщане на датата като текст:

DATENAME(Part, Date)

* Извършване на аритметични операции с дати:

DATEADD(Part, Number, Date)

* Намиране на сегашната дата и час:

GETDATE()

* Заявки OFFSET и FETCH (работят само с ORDER BY):

SELECT Id, FirstName, LastName

FROM Empoyees

ORDER BY Id

OFFSET 10 ROWS

FETCH NEXT 5 ROWS ONLY

* Избиране на информация по частично съвпадение (Wildcards символи):

SELECT Id, FistName, LastName

FROM Employees

WHERE FirstName LIKE ‘Dim%’

WHERE Name LIKE ‘%max!%’ ESCAPE ‘!’

% - всякакъв стринг, вкл и с нулева дължина;

\_ - всеки отделен символ;

[...] - всеки символ в диапазона;

[^…] - всеки символ, който не е в диапазона.

* Групиране на колони:

SELECT e.DepartmentID

FROM Employees AS e

GROUP BY e.DepartmentID

SELECT DepartmentID, SUM(Salary) AS [Total Salary]

FROM Employees

GROUP BY DepatmentID

* Намиране на уникални стойности:

SELECT DISTINCT e.DepartmentID

FROM Employees AS e

* Агрегиращи функции:

COUNT()

SUM()

MAX()

MIN()

AVG()

* Клаузата HAVING:

SELECT DepartmentID,

SUM(Salary) AS [Total Salary]

FROM Employees

WHERE Salary > 40000

GROUP BY DepatmentID

HAVING SUM(Salary) >= 150000

* Каскадно изтриване и каскадно ъпдейтване:

CREATE TABLE Drivers

(

DriverID INT PRIMARY KEY,

DriverName VARCHAR(50)

)

CREATE TABLE Cars

(

CarID INT PRIMARY KEY,

DriverID INT,

CONSTRAINT FK\_Car\_Driver FOREIGN KEY (DriverID)

REFERENCES Drivers(DriverID)

ON DELETE CASCADE / ON UPDATE CASCADE

)

- ON DELETE/UPDATE NO ACTION - не разрешава промяна

- ON DELETE/UPDATE SET NULL

- IN DELETE/UPDATE SET DEFAULT

* INNER JOIN:

SELECT \* FROM TableA AS a

INNER JOIN TableB AS b

ON a.Id = b.Id

* LEFT/RIGHT/FULL OUTER JOIN:

SELECT \* FROM TableA AS a - връща всичко от първата таблица,

LEFT OUTER JOIN TableB AS b а от 2та само съвпаденията

ON a.Id = b.Id

RIGHT OUTER JOIN - връща всичко от 2та таблица..

FULL JOIN - ще върне всичко и от двете таблици

CROSS JOIN - връща всеки ред от първата таблица, комбиниран с

всички редове от втората таблица

* Ранкиране на данните:

SELECT

DENSE\_RANK() OVER (exp: ORDER BY AVG(Salary)) AS Ranking

FROM Employees

RANK() - при равни стойности прескача число

ROW\_NUMBER() - просто номерира редовете

NTILE() - разделя на N части редовете

PARTITION BY - разделя ги по нещо

LEAD(smth, 1) OVER (ORDER BY) - дава ст-та от следващия ред

* Common Table Expressions:

WITH CTE\_Name (ColumnA, ColumnB….)

AS

(

-- Insert subquery here

)

* Създаване на клъстеризирани и неклъстеризирани индекси:

CREATE NONCLUSTERED/CLUSTERED INDEX

IX\_Employees\_FirstName\_LastName

ON Employees(FirstName, LastName)

INCLUDE (Salary) - индексът ще съдържа инфо за заплатите, но те

няма да са част от индексацията

* Създаване и променяне на функция:

CREATE/ALTER FUNCTION

udf\_ProjectDurationWeeks (@StartDate DATETIME, @EndDate DATETIME)

RETURNS INT

AS

BEGIN

DECLARE @projectWeeks INT;

IF(@EndDate IS NULL)

BEGIN

SET @EndDate = GETDATE()

END

SET @projectWeeks = DATEDIFF(WEEK, @StartDate, @EndDate)

RETURN @projectWeeks;

END

* Използване на функция:

SELECT

dbo.udf\_ProjectDurationWeeks([StartDate], [EndDate])

AS ProjectWeeks

FROM [SoftUni].[dbo].[Projects]

* Създаване на транзакция:

BEGIN TRANSACTION

UPDATE Accounts

SET Balance = Balance - @withdrawAmount

WHERE Id = @account

IF @@ROWCOUNT <> 1 -- Affected rows are different than one

BEGIN

ROLLBACK

RAISERROR(‘Invalid account!’, 16, 1)

RETURN

END

COMMIT

* Създаване на процедура:

CREATE PROC dbo.usp\_SelectEmployeesBySeniority

AS

SELECT \* FROM Employees

WHERE DATEDIFF(Year, HireDate, GETDATE()) > 5

GO

* Изпълняване на процедура:

INSERT INTO Customers

EXEC usp\_SelectEmployeesBySeniority parameter1, parameter2

* Променяне на процедура:

ALTER PROC usp\_SelectEmployeesBySeniority

AS

Some code here….

GO

* Изтриване на процедура:

DROP PROC usp\_SelectEmployeesBySeniority

EXEC sp\_depends ‘usp\_SelectEmployeesBySeniority’

* Създаване на процедура с параметри:

CREATE PROCEDURE usp\_ProcedureName

[(@parameter1Name parameterType, @parameter2Name parameterType, …..)] AS

* Създаване на процедура, връщаща резултат:

CREATE PROCEDURE dbo.usp\_AddNumbers

@firstNumber smallint,

@secondNumber smallint,

@result int OUTPUT

AS

SET @result = @firstNumber + @secondNumber

GO

DECLARE @answer smallint

EXECUTE usp\_AddNumbers 5, 6, @answer OUTPUT

SELECT ‘The result is: ’, @answer

-- The result is: 11

* Създаване на After Trigger:

CREATE TRIGGER tr\_TownsUpdate

ON Towns FOR UPDATE/INSERT/DELETE

AS

BEGIN

IF (EXISTS(SELECT \* FROM inserted WHERE NAME IS NULL)

OR

EXISTS(SELECT \* FROM inserted WHERE LEN(Name) = 0))

BEGIN

RAISERROR(‘Town name cannot be empty.’, 16, 1)

ROLLBACK TRAN

RETURN

END

END

UPDATE Towns SET Name = ‘’ WHERE TownId = 1

* Създаване на Instead of Trigger:

CREATE TABLE Accounts

(

Username VARCHAR(10) NOT NULL PRIMARY KEY,

Password VARCHAR(20) NOT NULL,

Active VARCHAR(1) NOT NULL DEFAULT ‘Y’

)

CREATE TRIGGER tr\_AccountsDelete ON Accounts

INSTEAD OF DELETE

AS

BEGIN

UPDATE a SET Active = ‘N’

FROM Accounts AS a

JOIN DELETED AS d

ON d.Username = a.Username

WHERE a.Active = ‘Y’

END

Да гледам кои колони може да са NULL и да проверявам за това!!!!!!

Ако проверявам дали стринг започва или свършва с нещо, да използвам първо TRIM или направо WILDCARDS !!!!!!

Да внимавам цифрите да не превъртат типа данни!!!

GETDATE() връща дата И час!!!!!

Да ползвам ранкиране, когато трябва пр. на най-голямото нещо да намеря други параметри!!!!

Параметрите на процедурата се изброяват без скоби при изпълнението й!!!!