SELECT \*

FROM Orders ord

JOIN [Order Details] od ON ord.OrderID=od.OrderID

JOIN Products pr ON od.ProductID=pr.ProductID

WHERE pr.ProductName LIKE'%a%' AND ShipCountry Like '%a%'

SELECT \*

FROM Orders ord

JOIN [Order Details] od ON ord.OrderID=od.OrderID

JOIN Products pr ON od.ProductID=pr.ProductID

WHERE pr.ProductName LIKE'%a%' AND ShipCountry Like 'Canada'

CREATE TABLE [Сотрудники](

[Табельный номер] int,

[ФИО] nvarchar(30),

[Дата рождения] date,

[E-mail] nvarchar(30),

[Должность] nvarchar(30),

[Отдел] nvarchar(30)

)

DROP TABLE [Сотрудники]

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30) NOT NULL,

Birthday date,

Email nvarchar(30),

Position nvarchar(30),

Department nvarchar(30),

CONSTRAINT PK\_Employees PRIMARY KEY(ID) -- описываем PK после всех полей, как ограничение

)

ALTER TABLE Employees ALTER COLUMN Name nvarchar(30) NOT NULL

ALTER TABLE Employees ALTER COLUMN ID int NOT NULL

ALTER TABLE Employees3 DROP CONSTRAINT PK\_Employees3

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30) NOT NULL,

Birthday date,

Email nvarchar(30),

Possition nvarchar(30),

Department nvarchar(30)

)

INSERT Employees(ID,Position,Department,Name) VALUES

(1000,N'Директор',N'Администрация',N'Иванов И.И.'),

(1001,N'Программист',N'ИТ',N'Петров П.П.'),

(1002,N'Бухгалтер',N'Бухгалтерия',N'Сидоров С.С.'),

(1003,N'Старший программист',N'ИТ',N'Андреев А.А.')

ALTER TABLE Employees ALTER COLUMN Name nvarchar(30) NULL

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30) NOT NULL,

Birthday date,

Email nvarchar(30),

Possition nvarchar(30),

Department nvarchar(30),

CONSTRAINT PK\_Employees PRIMARY KEY(ID) -- описываем РК после всех полей как ограничение

)

CREATE TABLE Employees2(

ID int NOT NULL CONSTRAINT PK\_Employees2 PRIMARY KEY, -- указываем как характеристику поля

Name nvarchar(30) NOT NULL,

Birthday date,

Email nvarchar(30),

Position nvarchar(30),

Department nvarchar(30)

)

­­

CREATE TABLE #Temp(

ID int,

Name nvarchar(30)

)

­­­

CREATE TABLE Positions(

ID int IDENTITY(1,1) NOT NULL CONSTRAINT PK\_Positions PRIMARY KEY,

Name nvarchar(30) NOT NULL

)

CREATE TABLE Departments(

ID int IDENTITY(1,1) NOT NULL CONSTRAINT PK\_Departments PRIMARY KEY,

Name nvarchar(30) NOT NULL

)

-- заполняем поле Name таблицы Positions, уникальными значениями из поля Positions таблицы Employees

INSERT Positions(Name)

SELECT DISTINCT Position

FROM Employees

WHERE Position IS NOT NULL -- отбрасываем записи у которых позиция не указана

INSERT Departments(Name)

SELECT DISTINCT Department

FROM Employees

WHERE Department IS NOT NULL

SELECT \*

FROM Positions

-- добавляем поле для ID должности

ALTER TABLE Employees ADD PositionID int

-- добавляем поле для ID отдела

ALTER TABLE Employees ADD DepartmentID int

ALTER TABLE Employees ADD CONSTRAINT FK\_Employees\_PositionID

FOREIGN KEY(PositionID) REFERENCES Positions(ID)

ALTER TABLE Employees ADD CONSTRAINT FK\_Employees\_DepartmentID

FOREIGN KEY(DepartmentID) REFERENCES Departments(ID)

UPDATE e

SET

PositionID=(SELECT ID FROM Positions WHERE Name=e.Position),

DepartmentID=(SELECT ID FROM Departments WHERE Name=e.Department)

FROM Employees e

ALTER TABLE Employees DROP COLUMN Position,Department

SELECT e.ID,e.Name,p.Name PositionName,d.Name DepartmentName

FROM Employees e

LEFT JOIN Departments d ON d.ID=e.DepartmentID

LEFT JOIN Positions p ON p.ID=e.PositionID

ALTER TABLE Employees ADD ManagerID int

ALTER TABLE Employees ADD CONSTRAINT FK\_Employees\_ManagerID

FOREIGN KEY (ManagerID) REFERENCES Employees(ID)

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30),

Birthday date,

Email nvarchar(30),

PositionID int,

DepartmentID int,

ManagerID int,

CONSTRAINT PK\_Employees PRIMARY KEY (ID),

CONSTRAINT FK\_Employees\_DepartmentID FOREIGN KEY(DepartmentID) REFERENCES Departments(ID)

ON DELETE CASCADE,

CONSTRAINT FK\_Employees\_PositionID FOREIGN KEY(PositionID) REFERENCES Positions(ID),

CONSTRAINT FK\_Employees\_ManagerID FOREIGN KEY (ManagerID) REFERENCES Employees(ID)

)

INSERT Employees (ID,Name,Birthday,PositionID,DepartmentID,ManagerID)VALUES

(1000,N'Иванов И.И.','19550219',2,1,NULL),

(1001,N'Петров П.П.','19831203',3,3,1003),

(1002,N'Сидоров С.С.','19760607',1,2,1000),

(1003,N'Андреев А.А.','19820417',4,3,1000)

-- даем разрешение на добавление/изменение IDENTIFY значения

SET IDENTITY\_INSERT Departments ON

INSERT Departments(ID,Name) VALUES(3,N'ИТ')

-- запрещаем добавление/изменения IDENTITY значения

SET IDENTITY\_INSERT Departments OFF

-- даем разрешение на добавление/изменение IDENTIFY значения

SET IDENTITY\_INSERT Departments ON

INSERT Departments(ID,Name) VALUES(3,N'ИТ')

-- запрещаем добавление/изменения IDENTITY значения

SET IDENTITY\_INSERT Departments OFF

­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­UPDATE Employees SET Email='i.ivanov@test.tt' WHERE ID=1000

UPDATE Employees SET Email='p.petrov@test,tt' WHERE ID=1001

UPDATE Employees SET Email='s.sidorov@test.tt' WHERE ID=1002

UPDATE Employees SET Email='a.andreev@test.tt' WHERE ID=1003

­­­­­­­­­­­­­­­­­­­­

ALTER TABLE Employees ADD CONSTRAINT UQ\_Employees\_Email UNIQUE(Email)

ALTER TABLE Employees ADD HireDate date NOT NULL DEFAULT SYSDATETIME()

ALTER TABLE Employee ADD DEFAULT SYSDATETIME() FOR HireDate

­­­­­­­­­­­­­­

ALTER TABLE Employees ADD CONSTRAINT DF\_Employees\_HireDate DEFAULT SYSDATETIME() FOR HireDate

INSERT Employees(ID,Name,Email)VALUES(1004,N'Сергеев С.С.','s.sergeev@test.tt')

ALTER TABLE Employees ADD CONSTRAINT CK\_Employees\_ID CHECK(ID BETWEEN 1000 AND 1999)

INSERT Employees(ID,Email) VALUES(2000,'test@test.tt')

INSERT Employees(ID,Email) VALUES(1500,'test@test.tt')

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30),

Birthday date,

Email nvarchar(30),

PositionID int,

DepartmentID int,

HireDate date NOT NULL DEFAULT SYSDATETIME(), -- для DEFAULT я сделаю исключение

CONSTRAINT PK\_Employees PRIMARY KEY (ID),

CONSTRAINT FK\_Employees\_DepartmentID FOREIGN KEY(DepartmentID) REFERENCES Departments(ID),

CONSTRAINT FK\_Employees\_PositionID FOREIGN KEY(PositionID) REFERENCES Positions(ID),

CONSTRAINT UQ\_Employees\_Email UNIQUE (Email),

CONSTRAINT CK\_Employees\_ID CHECK (ID BETWEEN 1000 AND 1999)

)

ALTER TABLE Employees DROP CONSTRAINT PK\_Employees

ALTER TABLE Employees DROP CONSTRAINT UQ\_Employees\_Email

ALTER TABLE Employees ADD CONSTRAINT PK\_Employees PRIMARY KEY NONCLUSTERED (ID)

ALTER TABLE Employees ADD CONSTRAINT QU\_Employees\_Email UNIQUE CLUSTERED (Email)

DROP INDEX IDX\_Employees\_Name ON Employees

CREATE UNIQUE NONCLUSTERED INDEX UQ\_Employees\_EmailDesc ON Employees(Email DESC)

­­­­­

CREATE INDEX IDX\_Employees\_Name ON Employees(Name)

CREATE TABLE Employees(

ID int NOT NULL,

Name nvarchar(30),

Birthday date,

Email nvarchar(30),

PositionID int,

DepartmentID int,

HireDate date NOT NULL CONSTRAINT DF\_Employees\_HireDate DEFAULT SYSDATETIME(),

ManagerID int,

CONSTRAINT PK\_Employees PRIMARY KEY (ID),

CONSTRAINT FK\_Employees\_DepartmentID FOREIGN KEY(DepartmentID) REFERENCES Departments(ID),

CONSTRAINT FK\_Employees\_PositionID FOREIGN KEY(PositionID) REFERENCES Positions(ID),

CONSTRAINT FK\_Employees\_ManagerID FOREIGN KEY (ManagerID) REFERENCES Employees(ID),

CONSTRAINT UQ\_Employees\_Email UNIQUE(Email),

CONSTRAINT CK\_Employees\_ID CHECK(ID BETWEEN 1000 AND 1999),

INDEX IDX\_Employees\_Name(Name)

)

INSERT Employees (ID,Name,Birthday,Email,PositionID,DepartmentID,ManagerID)VALUES

(1000,N'Иванов И.И.','19550219','i.ivanov@test.tt',2,1,NULL),

(1001,N'Петров П.П.','19831203','p.petrov@test.tt',3,3,1003),

(1002,N'Сидоров С.С.','19760607','s.sidorov@test.tt',1,2,1000),

(1003,N'Андреев А.А.','19820417','a.andreev@test.tt',4,3,1000)

SELECT

5550/100\*15,

SYSDATETIME(), -- получение системной даты БД

SIN(0)+COS(0)

SELECT

ID,

ID/100, -- здесь произойдет целочисленное деление

CAST(ID AS float)/100, -- используем функцию CAST для преобразование в тип float

CONVERT(float,ID)/100, -- используем функцию CONVERT для преобразование в тип float

ID/100. -- используем преобразование за счет указания что знаменатель вещественное число

FROM Employees

ORDER BY ID­­­­­­­­­­­­­­­­

SELECT \*

FROM dbo.Employees -- dbo - имя схемы

SELECT emp.ID,emp.Name

FROM Employees AS emp

­SELECT Employees.ID,Employees.Name

FROM Employees

SELECT emp.ID,emp.Name

FROM Employees emp -- ключевое слово AS можно отпустить (я предпочитаю такой вариант)

-- создадим для демонстрации временную таблицу

CREATE TABLE #Trash(

ID int NOT NULL PRIMARY KEY,

Col1 varchar(10),

Col2 varchar(10),

Col3 varchar(10)

)

-- наполним данную таблицу всяким мусором

INSERT #Trash(ID,Col1,Col2,Col3)VALUES

(1,'A','A','A'), (2,'A','B','C'), (3,'C','A','B'), (4,'A','A','B'),

(5,'B','B','B'), (6,'A','A','B'), (7,'A','A','A'), (8,'C','A','B'),

(9,'C','A','B'), (10,'A','A','B'), (11,'A',NULL,'B'), (12,'A',NULL,'B')

-- посмотрим что возвращает запрос без опции DISTINCT

SELECT Col1,Col2,Col3

FROM #Trash

-- удалим временную таблицу

DROP TABLE #Trash

SELECT DISTINCT DepartmentID

FROM Employees

SELECT DISTINCT DepartmentID,PositionID

FROM Employees

-- создаем новые колонки

ALTER TABLE Employees ADD

LastName nvarchar(30), -- фамилия

FirstName nvarchar(30), -- имя

MiddleName nvarchar(30), -- отчество

Salary float, -- и конечно же ЗП в каких-то УЕ

BonusPercent float -- процент для вычисления бонуса от оклада

-- наполняем их данными (некоторые данные намеренно пропущены)

UPDATE Employees

SET

LastName=N'Иванов',FirstName=N'Иван',MiddleName=N'Иванович',

Salary=5000,BonusPercent= 50

WHERE ID=1000 -- Иванов И.И.

UPDATE Employees

SET

LastName=N'Петров',FirstName=N'Петр',MiddleName=N'Петрович',

Salary=1500,BonusPercent= 15

WHERE ID=1001 -- Петров П.П.

UPDATE Employees

SET

LastName=N'Сидоров',FirstName=N'Сидор',MiddleName=NULL,

Salary=2500,BonusPercent=NULL

WHERE ID=1002 -- Сидоров С.С.

UPDATE Employees

SET

LastName=N'Андреев',FirstName=N'Адрей',MiddleName=NULL,

WHERE ID=1003 -- Андреев А.А.

SELECT

-- даем имя вычисляемому столбцу

LastName+' '+FirstName+' '+MiddleName AS ФИО,

-- использование двойных кавычек, т.к. используеться пробел

HireDate AS "Дата приема",

-- использование квадратных скобок, т.к. используется пробел

Birthday AS [Дата рождения],

-- слово AS не обязательно

Salary ZP

FROM Employees

SELECT

LastName+' '+FirstName+' '+MiddleName FullName1,

-- 2 варианта для замены NULL пустыми строками '' (получаем поведение как в ORACLE)

ISNULL(LastName,'')+' '+ISNULL(FirstName,'')+' '+ISNULL(MiddleName,'')FULLName2,

CONCAT(LastName,' ',FirstName,' ',MiddleName) FullName3

FROM Employees

­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­

SELECT

'Дата приема'=HireDate, -- помимо "..." и [...] можно использовать '...'

[Дата рождения]=Birthday,

ZP=Salary

FROM Employees

SELECT

ID,Name,

Salary/100\*BonusPercent AS Result1, -- без обработки NULL значений

Salary/100\*ISNULL(BonusPercent,0) AS Result2, -- используем функцию ISNULL

Salary/100\*COALESCE(BonusPercent,0) AS Result3 -- используем функцию COALESCE

FROM Employees

SELECT ID,Name

FROM Employees

WHERE ID%2=0 -- остаток от деления на 2 равен 0

SELECT

LastName,

FirstName,

Salary

FROM Employees

ORDER BY LastName,FirstName -- упорядочить результаты по 2-м стоблцам - по фамилии, по имени

SELECT LastName,FirstName,Salary

FROM Employees

ORDER BY -- упорядочить в порядке

Salary DESC, -- 1. убывания Заработной Платы

LastName, -- 2. по Фамилии

FirstName -- 3. по Имени

SELECT TOP 3 -- вернуть только первые 3 записи из всего результата

ID,LastName,FirstName

FROM Employees

ORDER BY Salary DESC -- сортируем результат по убыванию Заработноай Платы

SELECT TOP 3 -- вернуть только первые 3 записи из всего результата

ID,LastName,FirstName

FROM Employees

ORDER BY

Salary DESC, -- 1. сортируем результат по убыванию Заработной Платы

Birthday ASC, -- 2. потом по Дате рождения

ID DESC -- 3. и для полной однозначности результата добавляем сортировку по ID

SELECT LastName,FirstName

FROM Employees

ORDER BY CONCAT(LastName,' ',FirstName) -- используем выражение

SELECT CONCAT(LastName,' ',FirstName) fi

FROM Employees

ORDER BY fi -- используем псевдоним

SELECT BonusPercent FROM Employees ORDER BY BonusPercent

SELECT BonusPercent FROM Employees ORDER BY BonusPercent DESC

SELECT BonusPercent FROM Employees ORDER BY ISNULL(BonusPercent,100)

SELECT TOP 2

\*

FROM Employees

SELECT TOP 25 PERCENT

\*

FROM Employees

INSERT Employees(ID,Name,Email,PositionID,DepartmentID,ManagerID,Salary)

VALUES(1004,N'Николаев Н.Н.','n.nikolayev@test

INSERT Employees(ID,Name,Email,PositionID,DepartmentID,ManagerID,Salary)

VALUES(1005,N'Александров А.А.','a.alexanlrov@test.tt',NULL,NULL,1000,2000)

SELECT TOP 3 WITH TIES

ID,Name,Salary

FROM Employees

ORDER BY Salary

SELECT DISTINCT TOP 2

Salary

FROM Employees

ORDER BY Salary

SELECT ID,LastName,FirstName,Salary

FROM Employees

WHERE DepartmentID=3 -- ИТ

ORDER BY LastName,FirstName

SELECT DISTINCT TOP 1

Salary

FROM Employees

WHERE DepartmentID=3

ORDER BY Salary

SELECT ID,Name

FROM Employees

WHERE DepartmentID IS NULL

SELECT ID,Name,Salary/100\*BonusPercent AS Bonus

FROM Employees

WHERE BonusPercent IS NOT NULL

SELECT ID,Name,Salary/100\*BonusPercent AS Bonus

FROM Employees

WHERE NOT(BonusPercent<=0 OR BonusPercent IS NULL)

SELECT ID,Salary/100\*BonusPercent AS Bonus

FROM Employees

WHERE BonusPercent>0 AND BonusPercent IS NOT NULL

SELECT ID,Name,Salary/100\*BonusPercent AS Bonus

FROM Employees

WHERE ISNULL(BonusPercent,0)>0

SELECT ID,Name,Salary

FROM Employees

WHERE Salary BETWEEN 2000 AND 3000 -- у кого зарплата в диапозоне 2000-3000

SELECT ID,Name,Salary

FROM Employees

WHERE Salary>=2000 AND Salary<=3000 -- все у кого ЗП в диапозоне 2000-3000

SELECT ID,Name,Salary

FROM Employees

WHERE Salary<=2000 OR Salary>=3000 -- все у кого ЗП в диапозоне 2000-3000

SELECT ID,Name,Salary

FROM Employees

WHERE Salary NOT BETWEEN 2000 AND 3000

-- аналогично выражения NOT(Salary>=2000 AND Salary<=3000)

SELECT ID,Name,Salary

FROM Employees

WHERE Salary BETWEEN 2000 AND 3000 -- у кого ЗП в диапозоне 2000-3000

AND DepartmentID=3 -- учитывать сотрудников только отдела 3

SELECT ID,Name,Salary

FROM Employees

WHERE PositionID IN(3,4) -- у кого должность равна 3 или 4

SELECT ID,Name,Salary

FROM Employees

WHERE PositionID=3 OR PositionID=4 -- у кого должность равна 3 или 4

SELECT ID,Name,Salary

FROM Employees

WHERE PositionID NOT IN(3,4) -- аналогично выражению NOT(PositionID=3 OR Position ID=4)

SELECT ID,Name,Salary

FROM Employees

WHERE PositionID<>3 AND PositionID<>4 -- равносильно PositionID NOT IN(3,4)

SELECT ID,Name,DepartmentID

FROM Employees

WHERE DepartmentID IN(1,2,NULL) -- NULL записи не войдут в результат

SELECT ID,Name,DepartmentID

FROM Employees

WHERE DepartmentID IN(1,2) -- 1 или 2

OR DepartmentID IS NULL -- или NULL

SELECT ID,Name,DepartmentID

FROM Employees

WHERE ISNULL(DepartmentID,-1) IN(1,2,-1)

-- если уверены, что в нет и не будет департамента с ID=-1

SELECT ID,Name,DepartmentID

FROM Employees

WHERE DepartmentID NOT IN(1,NULL)

SELECT ID,Name,DepartmentID

FROM Employees

WHERE DepartmentID<>1

AND DepartmentID<>NULL -- проблема из-за этой проверки на NULL - это условие всегда вернет NULL

SELECT ID,Name,DepartmentID

FROM Employees

WHERE DepartmentID NOT IN(1) -- или в данном случае просто DepartmentID<>1

AND DepartmentID IS NOT NULL -- и отдельно проверяем на NOT NULL

SELECT ID,Name

FROM Employees

WHERE Name LIKE 'Пет%' -- у кого имя начинается с букв "Пет"

SELECT ID,LastName

FROM Employees

WHERE LastName LIKE '%ов' -- у кого фамилия оканчивается на "ов"

SELECT ID,LastName

FROM Employees

WHERE LastName LIKE '%ре%' -- у кого фамилия содержит сочетание "ре"

SELECT ID,LastName

FROM Employees

WHERE LastName LIKE '\_етров' -- у кого фамилия состоит из первого символа и последующих букв "етров"

SELECT ID,LastName

FROM Employees

WHERE LastName LIKE '\_\_\_\_ов' -- у кого фамилия состоит из четырех любых символов и последующих букв "ов"

UPDATE Employees

SET

FirstName='Это\_мусор, содержащий %'

WHERE ID=1005

SELECT \*

FROM Employees

WHERE FirstName LIKE '%!%%' ESCAPE '!' -- строка содержит знак "%"

SELECT \*

FROM Employees

WHERE FirstName LIKE '%!\_%' ESCAPE '!' -- строка содержит знак "\_"

SELECT ID,Name

FROM Employees

WHERE Name LIKE N'Пет%'

SELECT ID,LastName

FROM Employees

WHERE LastName=N'Петров'

SELECT ID,Name

FROM Employees

WHERE UPPER(Name) LIKE UPPER(N'Пет%') -- или LOWER(Name) LIKE LOWER(N'Пет%')

SELECT ID,LastName

FROM Employees

WHERE UPPER(LastName)=UPPER(N'Петров') -- или LOWER(LastName)=LOWER(N'Петров')

SELECT ID,Name,Birthday

FROM Employees

WHERE Birthday BETWEEN '19800101' AND '19891231' -- сотрудники 80-х годов

ORDER BY Birthday

SELECT ID,Name,Birthday

FROM Employees

WHERE Birthday BETWEEN DATEFROMPARTS(1980,1,1) AND DATEFROMPARTS(1989,12,31)

ORDER BY Birthday

SELECT

CONVERT(date,'12.03.2015',104),

CONVERT(datetime,'2014-11-30 17:20:15',120)

DECLARE @min\_int int SET @min\_int=-2147483648

DECLARE @max\_int int SET @max\_int=2147483647

SELECT

-- (-2147483648)

@min\_int,CAST(@min\_int AS float),CONVERT(float,@min\_int),

-- 2147483647

@max\_int,CAST(@max\_int AS float),CONVERT(float,@max\_int),

-- numeric(16,6)

@min\_int/1., -- (-2147483648.000000)

@max\_int/1. -- 2147483647.000000

DECLARE @int int SET @int=123

SELECT

@int\*1., -- numeric(12,0) - 0 знаков после запятой

@int\*1.0, -- numeric(13,1) - 1 знак

@int\*1.00 -- numeric(14,2) - 2 знака

-- хотя порой лучше сделать явное преобразование

-- поведение при преобразовании money в varchar

DECLARE @money money

SET @money = 1025.123456789

-- произойдет явное преобразование в 1025.1235, т.к. тип money хранит только 4 числа после запятой

SELECT

@money, -- 1025.1235

-- по умолчанию CAST и CONVERT ведут себя одинаково (т.е. грубо говоря применяется стиль 0)

CAST(@money as varchar(20)), -- 1025.12

CONVERT(varchar(20), @money), -- 1025.12

CONVERT(varchar(20), @money,0),

-- 1025.12 (стиль 0 - без разделителя тысячных и 2 цифры после запятой (формат по умолчанию))

CONVERT(varchar(20), @money, 1),

-- 1,025.12 (стиль 1 - используется разделитель тысячных и 2 цифры после запятой)

CONVERT(varchar(20), @money, 2)

-- 1025.1235 (стиль 2 - без разделителя и 4 цифры после запятой)

-- поведение при преобразовании float в varchar

DECLARE @float1 float SET @float1 = 1025.123456789

DECLARE @float2 float SET @float2 = 1231025.123456789

SELECT

@float1, -- 1025.1234567890

@float2, -- 1231025.123456789

-- по умолчанию CAST и CONVERT ведут себя одинаково (т.е. грубо говоря применяют стиль 0)

-- стиль 0 - Не более 6 разрядов. По необходимости используются экспоненциальное представление чисел

-- при преобразовании в varchar здесь творяться действительно страшные вещи

CAST(@float1 as varchar(20)), -- 1025.12

CONVERT(varchar(20), @float1), -- 1025.12

CONVERT(varchar(20), @float1, 0), -- 1025.12

CAST(@float2 as varchar(20)), -- 1.23103e+006

CONVERT(varchar(20), @float2), -- 1.23103e+006

CONVERT(varchar(20), @float2, 0), -- 1.23103e+006

-- стиль 1 - Всегда 8 разрядов. Всегда используем экспоненциальное представление чисел.

-- этот стиль для float тоже не очень точен

CONVERT(varchar(20), @float1, 1), -- 1.0251235e+003

CONVERT(varchar(20), @float2, 1), -- 1,2310251e+006

-- стиль 2 - Всегда 16 разрядов. Всегда используется экспоненциальное представление чисел.

-- здесь с точностью уже получше

CONVERT(varchar(30), @float1, 2), -- 1.025123456789000e+003 - OK

CONVERT(varchar(30), @float2, 2) -- 1.231025123456789e+006 - OK

-- decimal u numeric

DECLARE @money money SET @money = 1025.123456789 -- 1025.1235

DECLARE @float1 float SET @float1 = 1025.123456789

DECLARE @float2 float SET @float2 = 1231025.123456789

DECLARE @numeric numeric(28,9) SET @numeric = 1025.123456789

SELECT

CAST(@numeric as varchar(20)), -- 1025.123456789

CONVERT(varchar(20), @numeric), -- 1025.123456789

CAST(@money as numeric(28,9)), --1025.123500000

CAST(@float1 as numeric(28,9)), -- 1025.123456789

CAST(@float2 as numeric(28,9)) -- 1231025.123456789

DECLARE @money money

SET @money = 1025.123456789

DECLARE @money money = 1025.123456789

SELECT

ID,Name,Salary,

CASE

WHEN Salary>=3000 THEN 'ЗП >=3000'

WHEN Salary>=2000 THEN '2000 <= ЗП < 3000'

ELSE 'ЗП < 2000'

END SalaryTypeWithELSE,

CASE

WHEN Salary>=3000 THEN 'ЗП >= 3000'

WHEN Salary>=2000 THEN '2000 <= ЗП < 3000'

END SalaryTypeWithoutELSE

FROM Employees

SELECT

ID,Name,Salary,DepartmentID,

-- для наглядности выведем процент в виде строки

CASE DepartmentID -- проверяемое значение

WHEN 2 THEN '10%' -- 10% от ЗП выдать Бухгалтерам

WHEN 3 THEN '15%' -- 15% от ЗП выдать ИТ-шникам

ELSE '5%' -- всем остальным по 5%

END NewYearBonusPercent,

-- построим выражение с использованием CASE, чтобы увидить сумму бонуса

Salary/100\*

CASE DepartmentID

WHEN 2 THEN 10 -- 10% от ЗП выдать Бухгалтерам

WHEN 3 THEN 15 -- 15% от ЗП выдать ИТ-шникам

ELSE 5 -- всем остальным по 5%

END BonusAmount

FROM Employees

SELECT

ID,Name,Salary,DepartmentID,

CASE

WHEN DepartmentID=2 THEN '10%' -- 10% от ЗП выдать Бухгалтерам

WHEN DepartmentID=3 THEN '15%' -- 15% от ЗП выдавать ИТ-шникам

ELSE '5%' -- всем остальным по 5%

END NewYearBonusPercent,

-- построил выражение с использованием CASE, что бы увидеть сумму бонуса

Salary/100\*

CASE

WHEN DepartmentID=2 THEN 10 -- 10% от ЗП выдать Бухгалтерам

WHEN DepartmentID=3 THEN 15 -- 15% от ЗП выдать ИТ-шникам

ELSE 5 -- всем остальным по 5%

END BonusAmount

FROM Employees

SELECT

ID,Name,Salary,

CASE

WHEN DepartmentID IN(1,2) THEN 'A'

WHEN DepartmentID=3 THEN

CASE PositionID -- вложенный CASE

WHEN 3 THEN 'B-1'

WHEN 4 THEN 'B-2'

END

ELSE 'C'

END Demo1,

IIF(DepartmentID IN(1,2),'A',

IIF(DepartmentID=3,CASE PositionID WHEN 3 THEN 'B-1' WHEN 4 THEN 'B-2' END, 'C')) Demo2

FROM Employees

SELECT

ID,Name,Salary

FROM Employees

ORDER BY

CASE WHEN Salary>=2500 THEN 1 ELSE 0 END, -- выдать ЗП сначала тем у кого она ниже 2500

Name -- дальше упорядочить список в порядке ФИО

SELECT

ID,Name,Salary

FROM Employees

WHERE CASE WHEN Salary>=2500 THEN 1 ELSE 0 END=1 -- все записи у которых выражение равно 1

SELECT

ID,Name,Salary,DepartmentID,

CASE

WHEN DepartmentID=2 THEN '10%' -- 10% от ЗП выдать Бухгалтерам

WHEN DepartmentID=3 THEN '15%' -- 15% от ЗП выдать ИТ-шникам

WHEN DepartmentID IS NULL THEN '-' -- внештатникам бонусов не даем (используем IS NULL)

ELSE '5%' -- всем остальным по 5%

END NewyearBonusPercent1,

-- а так проверять на NULL нельзя, вспоминаем что говорилось про NULL во второй части

CASE DepartmentID -- проверяемое значение

WHEN 2 THEN '10%'

WHEN 3 THEN '15%'

WHEN NULL THEN '-' -- !!! в данном случае использование второй формы не подходит

ELSE '5%'

END NewYearBonusPercent2

FROM Employees

SELECT

ID,Name,Salary,DepartmentID,

CASE ISNULL(DepartmentID,-1) -- использую замену в случае NULL на -1

WHEN 2 THEN '10%'

WHEN 3 THEN '15%'

WHEN -1 THEN '-' -- если мы уверены, что отдела в ID равным (-1) нет и не будет

ELSE '5%'

END NewYearBonusPercent3

FROM Employees

SELECT

ID,Name,LastName,

ISNULL(LastName,'Не указано') DemoISNULL,

CASE WHEN LastName IS NULL THEN 'Не указано' ELSE LastName END DemoCASE,

IIF(LastName IS NULL,'Не указано',LastName) DemoIIF

FROM Employees

SELECT

COUNT(\*) [Общее кол-во сотрудников],

COUNT(DISTINCT DepartmentID) [Число уникальных отделов],

COUNT(DISTINCT PositionID) [Число уникальных должностей],

COUNT(BonusPercent) [Кол-во сотрудников у которых указан % бонуса],

MAX(BonusPercent) [Максимальный процент бонуса],

MIN(BonusPercent) [Минимальный процент бонуса],

SUM(Salary/100\*BonusPercent) [Сумма всех бонусов],

AVG(Salary/100\*BonusPercent) [Средний размер бонуса],

AVG(Salary) [Средний размер ЗП]

FROM Employees

SELECT

DepartmentID,

PositionID,

BonusPercent,

Salary/100\*BonusPercent [Salary/100\*BonusPercent],

Salary

FROM Employees

SELECT DISTINCT DepartmentID -- 2. берем только уникальные значения

FROM Employees

WHERE DepartmentID IS NOT NULL -- 1. отбрасываем NULL

SELECT BonusPercent -- 2. берем все значения

FROM Employees

WHERE BonusPercent IS NOT NULL -- 1. отбрасываем NULL значения

SELECT

COUNT(\*), -- 6

COUNT(DISTINCT DepartmentID), -- 3

COUNT(DepartmentID) -- 5

FROM Employees

SELECT TOP 1 BonusPercent

FROM Employees

WHERE BonusPercent IS NOT NULL

ORDER BY BonusPercent DESC -- сортируем по убыванию

SELECT TOP 1 BonusPercent

FROM Employees

WHERE BonusPercent IS NOT NULL

ORDER BY BonusPercent -- сортируем по возрастанию

SELECT Salary/100\*BonusPercent

FROM Employees

WHERE Salary/100\*BonusPercent IS NOT NULL

SELECT

AVG(Salary/100\*BonusPercent), -- 1108.3333333333

SUM(Salary/100\*BonusPercent)/COUNT(Salary/100\*BonusPercent), -- 1108.33333333333

SUM(Salary/100\*BonusPercent)/COUNT(\*) -- 554.166666666667

FROM Employees

SELECT

AVG(ISNULL(Salary/100\*BonusPercent,0)), -- 554.166666666667

SUM(Salary/100\*BonusPercent)/COUNT(\*) -- 554.166666666667

FROM Employees

SELECT

COUNT(\*) [Общее кол-во сотрудников],

COUNT(DISTINCT DepartmentID) [Число уникальных отделов],

COUNT(DISTINCT PositionID) [Число уникальных должностей],

COUNT(BonusPercent) [Кол-во сотрудников у которых указан % бонуса],

MAX(BonusPercent) [Максимальных процент бонуса],

MIN(BonusPercent) [Минимальных процент бонуса],

SUM(Salary/100\*BonusPercent) [Сумма всех бонусов],

AVG(Salary/100\*BonusPercent) [Средний размер бонуса],

AVG(Salary) [Средний размер ЗП]

FROM Employees

WHERE DepartmentID=3 -- учесть только ИТ- отдел

SELECT

DepartmentID,

PositionID,

BonusPercent,

Salary/100\*BonusPercent [Salary/100\*BonusPercent],

Salary

FROM Employees

WHERE DepartmentID=3 -- учесть только ИТ-отдел

SELECT

DepartmentID,

PositionID,

BonusPercent,

Salary/100\*BonusPercent [Salary/100\*BonusPercent],

Salary

FROM Employees

WHERE DepartmentID=3 -- учесть только ИТ-отдел

SELECT

SUM(Salary),

AVG(Salary),

-- обрабатывается итог при помощи ISNULL

ISNULL(SUM(Salary),0),

ISNULL(AVG(Salary),0)

FROM Employees

WHERE DepartmentID=10

-- здесь специально указан несуществующий отдел, чтобы запрос не вернул записей

SELECT

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmout

FROM Employees

WHERE DepartmentID=3 -- данные только по ИТ-отделу

SELECT

'Адмистрация' Info,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=1 -- данные по Администрации

SELECT

'Адмистрация' Info,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=1 -- данные по Администрации

SELECT

'Бухгалтерия' Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=2 -- данные по Бухгалтерии

SELECT

'Адмистрация' Info,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=1 -- данные по Администрации

SELECT

'Бухгалтерия' Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=2 -- данные по Бухгалтерии

SELECT

'ИТ' Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=3 -- данные по ИТ отделу

'Адмистрация' Info,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=1 -- данные по Администрации

SELECT

'Бухгалтерия' Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=2 -- данные по Бухгалтерии

SELECT

'ИТ' Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=3 -- данные по ИТ отделу

SELECT

'Прочие' Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID IS NULL -- и еще не забываем данные по внештатникам

SELECT

DepartmentID,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmout,

AVG(Salary) SalaryAvg -- плюс выполняем пожелания директора

FROM Employees

GROUP BY DepartmentID

SELECT

DepartmentID,PositionID,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID,PositionID

SELECT

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID IS NULL AND PositionID IS NULL

SELECT

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID IS NULL AND PositionID IS NULL

SELECT

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=1 AND PositionID=2

SELECT

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID IS NULL AND PositionID IS NULL

SELECT

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=1 AND PositionID=2

-- ...

SELECT

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmout

FROM Employees

WHERE DepartmentID=3 AND PositionID=4

SELECT

'Строка константа' Const1, -- константа в виде строки

1 Const2, -- константа в виде числа

-- выражение с использованием полей участвующих в групперовке

CONCAT('Отдел № ',DepartmentID) ConstAndGroupField,

CONCAT('Отдел № ',DepartmentID,', Должность № ',PositionID) ConstAndGroupFields,

DepartmentID, -- поле из списка полей участвующих в групперовке

-- PositionID, -- поле учавствующее в групперовке, не обязательно дублировать здесь

COUNT(\*) EmplCount, -- кол-во строк в каждой группе

-- остальные поля можно использовать только с агрегатными функциями: COUNT, SUM, MIN, MAX,...

SUM(Salary) SalaryAmount,

MIN(ID) MINID

FROM Employees

GROUP BY DepartmentID,PositionID -- группировка по полям DepartmentID,PositionID

SELECT

CONCAT('Год рождения - ',YEAR(Birthday)) YearOfBirthday,

COUNT(\*) EmplCount

FROM Employees

GROUP BY YEAR(Birthday)

SELECT

CASE

WHEN YEAR(Birthday)>=2000 THEN 'от 2000'

WHEN YEAR(Birthday)>=1990 THEN '1999-1990'

WHEN YEAR(Birthday)>=1980 THEN '1989-1980'

WHEN YEAR(Birthday)>=1970 THEN '1979-1970'

WHEN Birthday IS NOT NULL THEN 'ранее 1970'

ELSE 'не указано'

END RangerName,

COUNT(\*) EmplCount

FROM Employees

GROUP BY

CASE

WHEN YEAR(Birthday)>=2000 THEN 'от 2000'

WHEN YEAR(Birthday)>=1990 THEN '1999-1990'

WHEN YEAR(Birthday)>=1980 THEN '1989-1980'

WHEN YEAR(Birthday)>=1970 THEN '1979-1970'

WHEN Birthday IS NOT NULL THEN 'ранее 1970'

ELSE 'не указано'

END RangeName,

COUNT(\*) EmplCount

FROM Employees

GROUP BY

CASE

WHEN YEAR(Birthday)>=2000 THEN 'от 2000'

WHEN YEAR(Birthday)>=1990 THEN '1999-1990'

WHEN YEAR(Birthday)>=1980 THEN '1989-1980'

WHEN YEAR(Birthday)>=1970 THEN '1979-1970'

WHEN Birthday IS NOT NULL THEN 'ранее 1970'

ELSE 'не указано'

END

SELECT

CASE

WHEN YEAR(Birthday)>=2000 THEN 'от 2000'

WHEN YEAR(Birthday)>=1990 THEN '1999-1990'

WHEN YEAR(Birthday)>=1980 THEN '1989-1980'

WHEN YEAR(Birthday)>=1970 THEN '1979-1970'

WHEN Birthday IS NOT NULL THEN 'ранее 1970'

ELSE 'не указано'

END RangeName,

COUNT(\*) EmplCount

FROM Employees

GROUP BY

CASE

WHEN YEAR(Birthday)>=2000 THEN 'от 2000'

WHEN YEAR(Birthday)>=1990 THEN '1999-1990'

WHEN YEAR(Birthday)>=1980 THEN '1989-1980'

WHEN YEAR(Birthday)>=1970 THEN '1979-1970'

WHEN Birthday IS NOT NULL THEN 'ранее 1970'

ELSE 'не указано'

END

SELECT

ID,

CASE

WHEN YEAR(Birthday)>=2000 THEN 'от 2000'

WHEN YEAR(Birthday)>=1990 THEN '1999-1990'

WHEN YEAR(Birthday)>=1980 THEN '1989-1980'

WHEN YEAR(Birthday)>=1970 THEN '1979-1970'

WHEN Birthday IS NOT NULL THEN 'ранее 1970'

ELSE 'не указано'

END

FROM Employees

SELECT

CASE DepartmentID

WHEN 1 THEN 'Администрация'

WHEN 2 THEN 'Бухгалтерия'

WHEN 3 THEN 'ИТ'

ELSE 'Прочие'

END Info,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount,

AVG(Salary) SalaryAvg -- плюс выполняем пожелания директора

FROM Employees

GROUP BY DepartmentID

ORDER BY Info -- добавим для большого удобства сортировку по колонке Info

SELECT

ISNULL(dep.Name,'Прочие') DepName,

COUNT(DISTINCT emp.PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(emp.Salary) SalaryAmount,

AVG(emp.Salary) SalaryAvg -- плюс выполняем пожелания директора

FROM Employees emp

LEFT JOIN Departments dep ON emp.DepartmentID=dep.ID

GROUP BY emp.DepartmentID,dep.Name

ORDER BY DepName

SELECT

DepartmentID,

SUM(CASE WHEN PositionID=1 THEN Salary END) [Бухгалтера],

SUM(CASE WHEN PositionID=2 THEN Salary END) [Директора],

SUM(CASE WHEN PositionID=3 THEN Salary END) [Программисты],

SUM(CASE WHEN PositionID=4 THEN Salary END) [Старшие программисты],

SUM(Salary) [Итого по отделу]

FROM Employees

GROUP BY DepartmentID

SELECT

DepartmentID=1,

SUM(IIF(PositionID=1,Salary,NULL)) [Бухгалтера],

SUM(IIF(PositionID=2,Salary,NULL)) [Директора],

SUM(IIF(PositionID=3,Salary,NULL)) [Программисты],

SUM(IIF(PositionID=4,Salary,NULL)) [Старшие программисты],

SUM(Salary) [Итого по отделу]

FROM Employees

GROUP BY DepartmentID

SELECT

DepartmentID,

SUM(IIF(PositionID=1,Salary,NULL)) [Бухгалтера],

SUM(IIF(PositionID=2,Salary,NULL)) [Директора],

SUM(IIF(PositionID=3,Salary,NULL)) [Программисты],

SUM(IIF(PositionID=4,Salary,NULL)) [Старшие программисты],

SUM(Salary) [Итого по отделу]

FROM Employees

GROUP BY DepartmentID

SELECT

emp.\*, -- вернуть все поля таблицы Employees

dep.Name DepartmentName,

-- к этим полям добавить поле Name из таблицы Departments

pos.Name PositionName

-- и еще добавить поле Name из таблицы Positions

FROM Employees emp

LEFT JOIN Departments dep ON emp.DepartmentID=dep.ID

LEFT JOIN Positions pos ON emp.PositionID=pos.ID

CREATE VIEW ViewEmployeesInfo

AS

SELECT

emp.\*, -- вернуть все поля таблицы

SELECT SUM(od.UnitPrice\* od.Quantity)

FROM ORDERS

JOIN [Order Details] od ON Orders.OrderID = od.OrderID

GROUP BY Orders.OrderID Order BY SUM(od.UnitPrice\* od.Quantity) DESC

SELECT City,Count(\*)

FROM Employees

GROUP BY City

ORDER BY Count(\*) DESC

SELECT \*

FROM Orders ord

JOIN [Order Details] od ON ord.OrderID=od.OrderID

JOIN Products pr ON od.ProductID=pr.ProductID

WHERE pr.ProductName LIKE'Z%'

CREATE VIEW ViewEmployeesInfo

AS

SELECT

emp.\*, -- вернуть все поля таблицы Employees

dep.Name DepartmentName, -- к этим полям добавить поле Name из таблицы Departments

pos.Name PositionName -- и еще добавь поле Name из таблицы Positions

FROM Employees emp

LEFT JOIN Departments dep ON emp.DepartmentID=dep.ID

LEFT JOIN Positions pos ON emp.PositionID=pos.ID

SELECT

DepartmentName,

COUNT(DISTINCT PositionID) PositionCount,

COUNT(\*) EmplCount,

SUM(Salary) SalaryAmount,

AVG(Salary) SalaryAvg

FROM ViewEmployeesInfo emp

GROUP BY DepartmentID,DepartmentName

ORDER BY DepartmentName

SELECT

ID,

Name,

Salary

FROM ViewEmployeesInfo

WHERE Salary IS NOT NULL

AND Salary>0

ORDER BY Name

SELECT

DepartmentID,

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID

HAVING SUM(Salary)>3000

SELECT

DepartmentID,

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID -- 1. получаем сгруппированные данные по всем отделам

SELECT

DepartmentID,

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID -- 1. получаем сгруппированные данные по всем отделам

HAVING SUM(Salary)>3000 -- 2. условие для фильтрации сгруппированных данных

SELECT

DepartmentID,

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID

HAVING SUM(Salary)>3000 AND COUNT(\*)<2 -- и число людей мельше 2-х

SELECT

DepartmentID

FROM Employees

GROUP BY DepartmentID

HAVING SUM(Salary)>3000 AND COUNT(\*)<2 -- и число людей меньше 2-х

SELECT

DepartmentID,

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID -- 1. сделать группировку

HAVING DepartmentID=3 -- 2. наложить фильтр на результат группировки

SELECT

DepartmentID,

SUM(Salary) SalaryAmount

FROM Employees

WHERE DepartmentID=3 -- 1. провести фильтрацию детальных данных

GROUP BY DepartmentID -- 2. сделать группировку только по записям

SELECT

TOP 1 -- 6. приминиться в последнюю очередь

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID

HAVING SUM(Salary)>3000

ORDER BY DepartmentID -- 5. сортировка результата

SELECT

DISTINCT -- показать только уникальные значения SalaryAmount

SUM(Salary) SalaryAmount

FROM Employees

GROUP BY DepartmentID