**Домашняя работа №8**

**Архитектура MS SQL**

**Задания**

1. Создайте вашу базу данных: таблицы, ограничения, ключи.

CREATE TABLE Dim\_Salesperson (

SalespersonID INT IDENTITY(1,1),

CONSTRAINT PK\_SalePersonID PRIMARY KEY (SalespersonID),

FirstName NVARCHAR(50) NOT NULL,

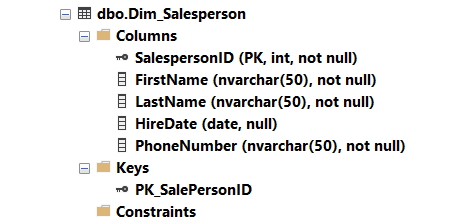
LastName NVARCHAR(50) NOT NULL,

HireDate DATE,

PhoneNumber NVARCHAR(50) NOT NULL

);

SELECT \* FROM Dim\_Salesperson



CREATE TABLE Dim\_Customer (

CustomerID INT IDENTITY(1,1),

CONSTRAINT PK\_CustomerID PRIMARY KEY (CustomerID),

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

BirthDate DATE,

Gender NVARCHAR(10) NOT NULL,

Address NVARCHAR(50) NOT NULL,

City NVARCHAR(50) NOT NULL,

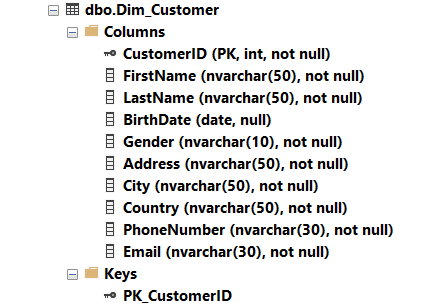
Country NVARCHAR(50) NOT NULL,

PhoneNumber NVARCHAR(30) NOT NULL,

Email NVARCHAR(30) NOT NULL

);

SELECT \* FROM Dim\_Customer



CREATE TABLE Dim\_Product (

ProductID INT IDENTITY(1,1),

CONSTRAINT PK\_ProductID PRIMARY KEY (ProductID),

ProductName NVARCHAR(50) NOT NULL,

ManufacturerName NVARCHAR(50) NOT NULL,

ManufacturerNum INT,

CONSTRAINT UQ\_ManufactNum UNIQUE (ManufacturerNum),

ProductPrice MONEY NOT NULL,

Size NVARCHAR (10) NOT NULL,

Color NVARCHAR (20) NOT NULL,

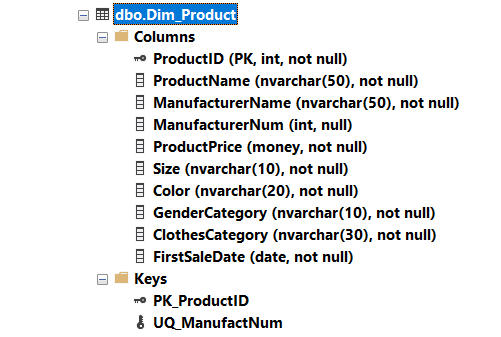
GenderCategory NVARCHAR(10) NOT NULL,

ClothesCategory NVARCHAR(30) NOT NULL,

FirstSaleDate DATE NOT NULL

);

SELECT \* FROM Dim\_Product



CREATE TABLE Dim\_Shipper (

ShipperID INT IDENTITY(1,1),

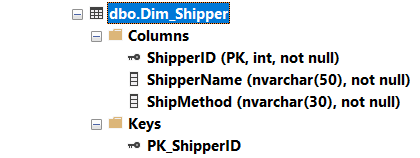
CONSTRAINT PK\_ShipperID PRIMARY KEY (ShipperID),

ShipperName NVARCHAR(50) NOT NULL,

ShipMethod NVARCHAR(30) NOT NULL

);

SELECT \* FROM Dim\_Shipper



CREATE TABLE Fct\_Order (

OrderID INT IDENTITY(1,1),

CONSTRAINT PK\_OrderID PRIMARY KEY (OrderID),

ProductID INT NOT NULL,

CONSTRAINT FK\_Order\_Product FOREIGN KEY (ProductID) REFERENCES Dim\_Product(ProductID),

OrderQty INT NOT NULL,

Discount FLOAT NOT NULL,

OrderDate DATE NOT NULL,

SalespersonID INT NOT NULL,

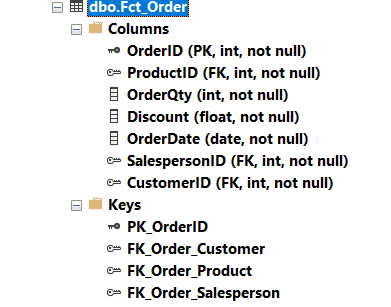
CONSTRAINT FK\_Order\_Salesperson FOREIGN KEY (SalespersonID) REFERENCES Dim\_Salesperson(SalespersonID),

CustomerID INT NOT NULL,

CONSTRAINT FK\_Order\_Customer FOREIGN KEY (CustomerID) REFERENCES Dim\_Customer(CustomerID)

);

SELECT \* FROM Fct\_Order



CREATE TABLE Fct\_Delivery (

DeliveryID INT IDENTITY(1,1),

CONSTRAINT PK\_DeliveryID PRIMARY KEY (DeliveryID),

CustomerID INT NOT NULL,

CONSTRAINT FK\_Delivery\_Customer FOREIGN KEY (CustomerID) REFERENCES Dim\_Customer(CustomerID),

OrderID INT NOT NULL,

CONSTRAINT FK\_Delivery\_Order FOREIGN KEY (OrderID) REFERENCES Fct\_Order(OrderID),

DeliveryDate DATE NOT NULL,

ShipperID INT NOT NULL,

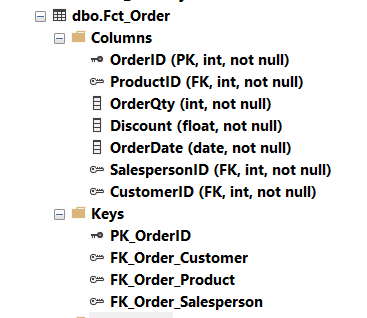
CONSTRAINT FK\_Delivery\_Shipper FOREIGN KEY (ShipperID) REFERENCES Dim\_Shipper(ShipperID),

DeliveryStatus NVARCHAR(30) NOT NULL,

AffectedDate DATE NOT NULL

);

SELECT \* FROM Fct\_Delivery





1. Напишите скрипт для получения 1 млн человек с различными именами и фамилиями. Полезная [ссылка](https://mockaroo.com/).

SELECT TOP 1000000

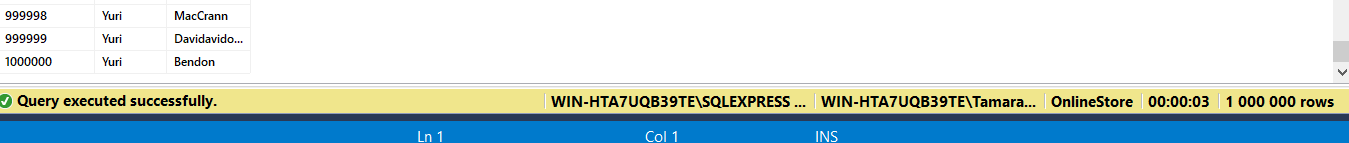
A.first\_name,

B.last\_name

FROM dbo.Atable A

CROSS JOIN

dbo.Btable B



1. Познакомьтесь с [обобщенным табличным выражением WITH](https://docs.microsoft.com/ru-ru/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-ver15).

Когда можем использовать данную структуру?

Common Table Expression (CTE) выражение с общей таблицей, которую можно использовать множество раз в запросе. CTE не сохраняет данные, а создает нечто вроде временного представления.

Можно использовать

- Для создания рекурсивных запросов, с помощью которых можно получить данные в иерархическом виде;

-При многократных ссылках на набор данных в пределах одного запроса;

-С целью заменить представления, временные таблицы, табличные переменные.

Чем отличается от подзапроса?

Общие табличные выражения позволяют существенно уменьшить объем кода, и скорость выполнения если многократно приходится обращаться к одним и тем же производным таблицам.