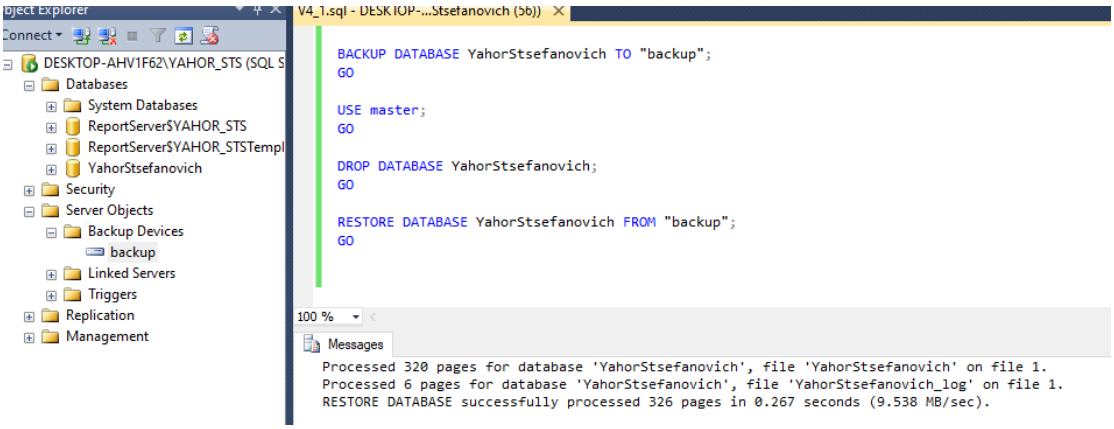
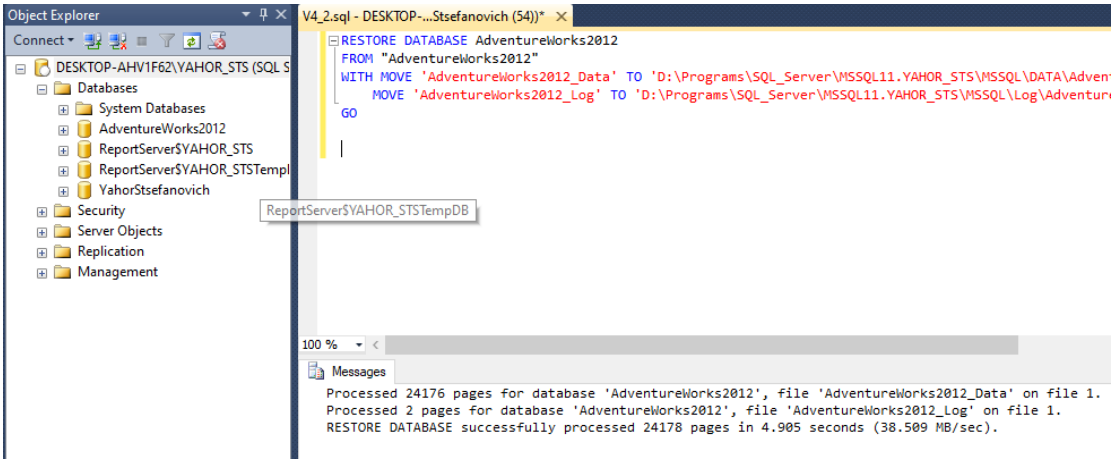
Оглавление

Task 1.1 Var 4


Object Explorer
Connect
DESKTOP-AHV1F62\YAHOR_STS (SQL S
Databases
System Databases
ReportServer$YAHOR_STS
ReportServer$YAHOR_STSTempl
YahorStsefanovich
Security
Server Objects
Backup Devices
backup
Linked Servers
Triggers
Replication
Management
V4_1.sql - DESKTOP-...Stsefanovich (56))
BACKUP DATABASE YahorStsefanovich TO "backup";
GO
USE master;
GO
DROP DATABASE YahorStsefanovich;
GO
RESTORE DATABASE YahorStsefanovich FROM "backup";
GO
100 %
Messages
Processed 320 pages for database 'YahorStsefanovich', file 'YahorStsefanovich' on file 1.
Processed 6 pages for database 'YahorStsefanovich', file 'YahorStsefanovich_log' on file 1.
RESTORE DATABASE successfully processed 326 pages in 0.267 seconds (9.538 MB/sec).

Task 1.2 Var4 - Backup AdventureWorks2012




```sql
/*Вывести на экран список отделов, принадлежащих группе 'Executive General and Administration'.*/
SELECT Name, GroupNAme
    FROM HumanResources.Department
    WHERE GroupName='Executive General and Administration';
```

100 %

Results | Messages

	Name	GroupNAme
1	Human Resources	Executive General and Administration
2	Finance	Executive General and Administration
3	Information Services	Executive General and Administration
4	Facilities and Maintenance	Executive General and Administration
5	Executive	Executive General and Administration

```sql
/*Вывести на экран максимальное количество оставшихся часов отпуска
 у сотрудников. Назовите столбец с результатом 'MaxVacationHours'.*/
SELECT MAX(VacationHours) AS MaxVacationHours
    FROM HumanResources.Employee;
```

100 %

Results | Messages

	MaxVacationHours
1	99

```sql
/*Вывести на экран сотрудников, название позиции которых включает слово 'Engineer'.*/
SELECT E.BusinessEntityID, E.JobTitle, E.Gender, E.BirthDate, E.HireDate
    FROM HumanResources.Employee AS E WHERE E.JobTitle LIKE '%Engineer%';
```

100 %

Results | Messages

	BusinessEntityID	JobTitle	Gender	BirthDate	HireDate
1	2	Vice President of Engineering	F	1965-09-01	2002-03-03
2	3	Engineering Manager	M	1968-12-13	2001-12-12
3	5	Design Engineer	F	1946-10-29	2002-02-06
4	6	Design Engineer	M	1953-04-11	2002-02-24
5	8	Research and Development Engineer	F	1980-07-06	2003-01-30
6	9	Research and Development Engineer	F	1973-02-21	2003-02-17
7	14	Senior Design Engineer	M	1973-07-17	2005-01-30
8	15	Design Engineer	F	1955-06-03	2005-02-18

Task2.1 Var4

```sql
/*Вывести на экран неповторяющийся список должностей в каждом отделе, отсортированный по названию отдела.
Посчитайте количество сотрудников, работающих в каждом отделе.*/
    SELECT DISTINCT D.Name,
                    E.JobTitle,
                    COUNT(E.BusinessEntityID) OVER (PARTITION BY edh.DepartmentID) AS EmpCount
    FROM HumanResources.Department AS D
    JOIN HumanResources.EmployeeDepartmentHistory AS EDH
        ON D.DepartmentID=EDH.DepartmentID
    JOIN HumanResources.Employee AS E
        ON EDH.BusinessEntityID=E.BusinessEntityID
    ORDER BY D.Name ASC;
```

100 %

Results | Messages

	Name	JobTitle	EmpCount
1	Document Control	Control Specialist	5
2	Document Control	Document Control Assistant	5
3	Document Control	Document Control Manager	5
4	Engineering	Design Engineer	7
5	Engineering	Engineering Manager	7
6	Engineering	Senior Design Engineer	7
7	Engineering	Senior Tool Designer	7
8	Engineering	Vice President of Engineering	7
9	Executive	Chief Executive Officer	2
10	Executive	Chief Financial Officer	2
11	Facilities and Maintenance	Facilities Administrative Assistant	7
12	Facilities and Maintenance	Facilities Manager	7
13	Facilities and Maintenance	Janitor	7
14	Facilities and Maintenance	Maintenance Supervisor	7
15	Finance	Accountant	11
16	Finance	Accounts Manager	11

```sql
/*Вывести на экран сотрудников, которые работают в ночную смену.*/
SELECT E.BusinessEntityID, E.JobTitle, S.Name, S.StartTime, S.EndTime FROM HumanResources.Employee AS E
    JOIN HumanResources.EmployeeDepartmentHistory AS EDH
        ON E.BusinessEntityID=EDH.BusinessEntityID
    JOIN HumanResources.Shift AS S
        ON S.ShiftID=EDH.ShiftID
    WHERE S.Name='Night';
```

100 %

Results | Messages

	BusinessEntityID	JobTitle	Name	StartTime	EndTime
1	40	Production Supervisor - WC60	Night	23:00:00.0000000	07:00:00.0000000
2	41	Production Technician - WC60	Night	23:00:00.0000000	07:00:00.0000000
3	42	Production Technician - WC60	Night	23:00:00.0000000	07:00:00.0000000
4	43	Production Technician - WC60	Night	23:00:00.0000000	07:00:00.0000000
5	44	Production Technician - WC60	Night	23:00:00.0000000	07:00:00.0000000
6	45	Production Technician - WC60	Night	23:00:00.0000000	07:00:00.0000000
7	46	Production Technician - WC60	Night	23:00:00.0000000	07:00:00.0000000
8	55	Production Supervisor - WC50	Night	23:00:00.0000000	07:00:00.0000000

Task 2.2 Var4

```sql
/* a) создайте таблицу dbo.StateProvince с такой же структурой как Person.StateProvince,
 кроме поля uniqueidentifier, не включая индексы, ограничения и триггеры;*/
CREATE TABLE dbo.StateProvince(
    StateProvinceID [int] IDENTITY(1,1) NOT NULL,
    StateProvinceCode [nchar](3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    CountryRegionCode [nvarchar](3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    IsOnlyStateProvinceFlag [dbo].[Flag] NOT NULL,
    Name dbo.[Name] NOT NULL,
    TerritoryID [int] NOT NULL,
    ModifiedDate [datetime] NOT NULL
) ON [PRIMARY];
GO
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/*b) используя инструкцию ALTER TABLE, создайте для таблицы dbo.StateProvince ограничение UNIQUE для поля Name;*/
ALTER TABLE dbo.StateProvince ADD CONSTRAINT NameContraint UNIQUE(Name);
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/* c) используя инструкцию ALTER TABLE, создайте для таблицы dbo.StateProvince ограничение для поля CountryRegionCode
 запрещающее заполнение этого поля цифрами;*/
ALTER TABLE dbo.StateProvince ADD CONSTRAINT CountryRegionCodePermitDigit
    CHECK (CountryRegionCode NOT LIKE '%^.*[0-9].*$%');
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/*d) используя инструкцию ALTER TABLE, создайте для таблицы dbo.StateProvince ограничение
  DEFAULT для поля ModifiedDate, задайте значение по умолчанию текущую дату и время;*/
ALTER TABLE dbo.StateProvince ADD CONSTRAINT DefaultModifiedDate DEFAULT GetDate() FOR ModifiedDate;
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/* e) заполните новую таблицу данными из Person.StateProvince.
 Выберите для вставки только те данные, где имя штата/государства совпадает с именем страны/региона в таблице CountryRegion;*/
INSERT INTO dbo.StateProvince
    SELECT
        SP.StateProvinceCode,
        SP.CountryRegionCode,
        SP.IsOnlyStateProvinceFlag,
        SP.Name,
        SP.TerritoryID,
        SP.ModifiedDate
    FROM Person.StateProvince AS SP
        JOIN Person.CountryRegion AS CR ON SP.CountryRegionCode=CR.CountryRegionCode
            WHERE SP.Name=CR.Name;
```

100 %

Messages

```
(6 row(s) affected)
```

```
/*f) удалите поле IsOnlyStateProvinceFlag, а вместо него создайте новое CountryNum типа int допускающее null значения.*/
ALTER TABLE dbo.StateProvince DROP COLUMN IsOnlyStateProvinceFlag;
ALTER TABLE dbo.StateProvince ADD CountryNum INT NULL;
```

100 %

Messages

```
Command(s) completed successfully.
```

Task 3.1 Var4

```
/*a) добавьте в таблицу dbo.StateProvince поле CountryRegionName типа nvarchar(50);*/
ALTER TABLE dbo.StateProvince ADD CountryRegionName NVARCHAR(50);
```

100 %

Messages

```
Command(s) completed successfully.
```

```
declare @StateProvinceVar table (
  StateProvinceID [int] NOT NULL,
  StateProvinceCode [nchar](3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
  CountryRegionCode [nvarchar](3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
  Name [dbo].[Name] NOT NULL,
  TerritoryID [int] NOT NULL,
  ModifiedDate [datetime] NOT NULL,
  CountryRegionName [nvarchar](50)
);

INSERT INTO @StateProvinceVar
SELECT
  SP.StateProvinceID,
  SP.StateProvinceCode,
  SP.CountryRegionCode,
  SP.Name,
  SP.TerritoryID,
  SP.ModifiedDate,
  CR.Name AS CountryRegionName
FROM
  [dbo].[StateProvince] AS SP
JOIN Person.CountryRegion AS CR ON SP.CountryRegionCode = CR.CountryRegionCode;

/*c) обновите поле CountryRegionName в dbo.StateProvince данными из табличной переменной;*/
UPDATE dbo.StateProvince SET CountryRegionName = V.CountryRegionName FROM @StateProvinceVar AS V
    WHERE StateProvince.StateProvinceID = V.StateProvinceID;
```

100 %

Messages

```
(6 row(s) affected)
```

```
/*d) удалите штаты из dbo.StateProvince, которые отсутствуют в таблице Person.Address;*/
DELETE FROM dbo.StateProvince WHERE StateProvinceID NOT IN (SELECT StateProvinceID FROM Person.Address);
```

100 %

Messages

```
(3 row(s) affected)
```

```
/*e) удалите поле CountryRegionName из таблицы, удалите все созданные ограничения и значения по умолчанию.*/
ALTER TABLE dbo.StateProvince
    DROP CONSTRAINT CountryRegionCodePermitDigit, DefaultModifiedDate, COLUMN CountryRegionName;
```

100 %

Messages

```
Command(s) completed successfully.
```

```
/*f) удалите таблицу dbo.StateProvince.*/
DROP TABLE dbo.StateProvince;
```

100 %

Messages

```
Command(s) completed successfully.
```

Task 3.2 Var4

```sql
/*a) выполните код, созданный во втором задании второй лабораторной работы.
Добавьте в таблицу dbo.StateProvince поля SalesYTD MONEY и SumSales MONEY.
Также создайте в таблице вычисляемое поле SalesPercent,
вычисляющее процентное выражение значения в поле SumSales от значения в поле SalesYTD.*/
ALTER TABLE dbo.StateProvince ADD SalesYTD MONEY;
ALTER TABLE dbo.StateProvince ADD SumSales MONEY;
ALTER TABLE dbo.StateProvince ADD SalesPercent AS ROUND(SalesYTD/SumSales*100, 0) PERSISTED;
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/*b) создайте временную таблицу #StateProvince, с первичным ключом по полю StateProvinceID.
Временная таблица должна включать все поля таблицы dbo.StateProvince за исключением поля SalesPercent.*/
CREATE TABLE #StateProvince (
 StateProvinceID [int] NOT NULL,
 StateProvinceCode [nchar](3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
 CountryRegionCode [nvarchar](3) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
 Name varchar(50) NOT NULL,
 TerritoryID [int] NOT NULL,
 ModifiedDate [datetime] NOT NULL,
 CountryNum [int],
 CountryRegionName [nvarchar](50),
 SalesYTD MONEY,
 SumSales MONEY
)
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/*c) заполните временную таблицу данными из dbo.StateProvince.
Поле SalesYTD заполните значениями из таблицы Sales.SalesTerritory.
Посчитайте сумму продаж (SalesYTD) для каждой территории (TerritoryID)
в таблице Sales.SalesPerson и заполните этими значениями поле SumSales.
Подсчет суммы продаж осуществите в Common Table Expression (CTE).*/
WITH SumSales_CTE AS
(
    SELECT ST.TerritoryID, SUM(ST.SalesLastYear) AS SumSales
    FROM dbo.StateProvince AS SP
    JOIN Sales.SalesTerritory AS ST ON SP.TerritoryID = ST.TerritoryID
    GROUP BY ST.TerritoryID
)
INSERT INTO #StateProvince
SELECT
  SP.StateProvinceID,
  SP.StateProvinceCode,
  SP.CountryRegionCode,
  SP.Name,
  SP.TerritoryID,
  SP.ModifiedDate,
  SP.CountryNum,
  ST.SalesLastYear,
  SS.SumSales
FROM
  [dbo].[StateProvince] AS SP
JOIN Sales.SalesTerritory AS ST ON SP.TerritoryID = ST.TerritoryID
JOIN SumSales_CTE AS SS ON SS.TerritoryID = SP.TerritoryID;
```

91 %

Messages

```
(6 row(s) affected)
```

```sql
/*d) удалите из таблицы dbo.StateProvince одну строку (где StateProvinceID = 5)*/
DELETE FROM dbo.StateProvince WHERE StateProvinceID = 5;
```

91 %

Messages

```
(1 row(s) affected)
```

```sql
/*e) напишите Merge выражение, использующее dbo.StateProvince как target,
а временную таблицу как source. Для связи target и source используйте StateProvinceID.
Обновите поля SalesYTD и SumSales, если запись присутствует в source и target.
Если строка присутствует во временной таблице, но не существует в target,
добавьте строку в dbo.StateProvince. Если в dbo.StateProvince присутствует такая строка,
которой не существует во временной таблице, удалите строку из dbo.StateProvince.*/
MERGE dbo.StateProvince AS TARGET
USING #StateProvince AS SOURCE ON
TARGET.StateProvinceID = SOURCE.StateProvinceID
WHEN MATCHED THEN
    UPDATE SET TARGET.SalesYTD = SOURCE.SalesYTD,
               TARGET.SumSales = SOURCE.SumSales
WHEN NOT MATCHED THEN
    INSERT VALUES (StateProvinceCode, CountryRegionCode,
            Name, TerritoryID, ModifiedDate, CountryNum, SalesYTD, SumSales)
WHEN NOT MATCHED BY SOURCE  THEN
    DELETE;
```

91 %

Messages

```
(6 row(s) affected)
```

Task 4.1 Var4

```sql
/*a) Создайте таблицу Production.ProductModelHst,
которая будет хранить информацию об изменениях в таблице Production.ProductModel.
Обязательные поля, которые должны присутствовать в таблице:
    ID — первичный ключ IDENTITY(1,1);
    Action — совершенное действие (insert, update или delete);
    ModifiedDate — дата и время, когда была совершена операция;
    SourceID — первичный ключ исходной таблицы;
    UserName — имя пользователя, совершившего операцию.
Создайте другие поля, если считаете их нужными.*/
CREATE TABLE Production.ProductModelHst(
    ID [INT] IDENTITY(1,1) NOT NULL,
    [Action] VARCHAR(10) NOT NULL CHECK ([Action] IN('insert', 'update', 'delete')),
    ModifiedDate DATETIME NOT NULL,
    SourceID [INT],
    UserName VARCHAR(30) NOT NULL
);
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/*b) Создайте один AFTER триггер для трех операций INSERT, UPDATE, DELETE
 для таблицы Production.ProductModel. Триггер должен заполнять
 таблицу Production.ProductModelHst с указанием типа операции
 в поле Action в зависимости от оператора, вызвавшего триггер.*/
CREATE TRIGGER onProductModelChanged
ON Production.ProductModel
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
DECLARE @eventType varchar(42);
DECLARE @sourceID int;
IF EXISTS(SELECT * FROM inserted)
BEGIN
    SELECT @sourceID = ProductModelID FROM inserted;
    IF EXISTS(SELECT * FROM deleted)
    BEGIN
        SELECT @eventType = 'update';
    END
ELSE
    BEGIN
        SELECT @eventType = 'insert';
    END
END
ELSE
BEGIN
    IF EXISTS(SELECT * FROM deleted)
    BEGIN
        SELECT @eventType = 'delete';
    END
    SELECT @sourceID = ProductModelID FROM deleted;
END
INSERT INTO Production.ProductModelHst([Action], ModifiedDate, SourceID, UserName)
    VALUES (@eventType, GETDATE(), @sourceID, USER_NAME());
END;
GO
```

75 %

Messages

```
Command(s) completed successfully.
```

```sql
/*c) Создайте представление VIEW, отображающее все поля таблицы Production.ProductModel.*/
CREATE VIEW ProductModelView AS SELECT * FROM Production.ProductModel;
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
/*d) Вставьте новую строку в Production.ProductModel через представление.
Обновите вставленную строку. Удалите вставленную строку.
Убедитесь, что все три операции отображены в Production.ProductModelHst.*/
INSERT INTO Production.ProductModel(Name)
    VALUES('Glory Glory Man United');
UPDATE Production.ProductModel
    SET Name = 'May the force be with you'
    WHERE Name = 'Glory Glory Man United';
DELETE FROM Production.ProductModel
    WHERE Name = 'May the force be with you';
SELECT * FROM Production.ProductModelHst;
```

esults | Messages

ID	Action	ModifiedDate	SourceID	UserName
1	insert	2019-09-21 13:53:04.277	129	dbo
2	update	2019-09-21 13:54:53.057	129	dbo
3	delete	2019-09-21 13:54:56.240	129	dbo

Task 4.2 Var4

```
/*a) Создайте представление VIEW, отображающее данные из таблиц Production.ProductModel,
Production.ProductModelProductDescriptionCulture, Production.Culture и Production.ProductDescription.
Сделайте невозможным просмотр исходного кода представления.
Создайте уникальный кластерный индекс в представлении по полям ProductModelID,CultureID.*/
CREATE VIEW dbo.ProductModelClusterView
WITH ENCRYPTION, SCHEMABINDING
AS
SELECT
   C.CultureID,
   C.Name AS C_Name,
   C.ModifiedDate AS C_ModifiedDate,
   PM.CatalogDescription,
   PM.Instructions,
   PM.Name AS PM_Name,
   PM.ProductModelID,
   PM.ModifiedDate AS PM_ModifiedDate,
   PD.Description,
   PD.ProductDescriptionID,
   PD.rowguid,
   PD.ModifiedDate AS PD_ModifiedDate,
   PMPDC.ModifiedDate AS PMPDC_ModifiedDate
FROM Production.ProductModel AS PM
JOIN Production.ProductModelProductDescriptionCulture AS PMPDC
   ON PM.ProductModelID = PMPDC.ProductModelID
JOIN Production.Culture AS C
   ON C.CultureID = PMPDC.CultureID
JOIN Production.ProductDescription AS PD
   ON PD.ProductDescriptionID = PMPDC.ProductDescriptionID;
GO

CREATE UNIQUE CLUSTERED INDEX PRODUCT_MODEL_INDX
   ON dbo.ProductModelClusterView(ProductModelID,CultureID);
GO
```

75 %

Messages

```
Command(s) completed successfully.
```

```
CREATE TRIGGER OnDeleteFromProductModelVIew
ON dbo.ProductModelClusterView
INSTEAD OF DELETE
AS
BEGIN
   -- Get Id's of deleted entities
   DECLARE @CultureID [int];
   DECLARE @ProductDescriptionID [int];
   DECLARE @ProductModelID [int];
   SELECT
       @CultureID = CultureID,
       @ProductDescriptionID = ProductDescriptionID,
       @ProductModelID = ProductModelID
   FROM deleted;

   --Delete Culture FROM ProductModelProductDescriptionCulture if not bound to it
   IF @CultureID NOT IN (SELECT CultureID FROM Production.ProductModelProductDescriptionCulture)
   BEGIN
   DELETE FROM Production.Culture
       WHERE CultureID = @CultureID;
   END;

   --Delete ProductDescription FROM ProductModelProductDescriptionCulture if not bound to it
   IF @ProductDescriptionID NOT IN (SELECT ProductDescriptionID FROM Production.ProductModelProductDescriptionCulture)
   BEGIN
   DELETE FROM Production.ProductDescription
       WHERE ProductDescriptionID = @ProductDescriptionID;
   END;

   --Delete ProductModel FROM ProductModelProductDescriptionCulture if not bound to it
   IF @ProductModelID NOT IN (SELECT ProductModelID FROM Production.ProductModelProductDescriptionCulture)
   BEGIN
   DELETE FROM Production.ProductModel
       WHERE ProductModelID = @ProductModelID;
   END;
END;
```

3 %

Messages

```
Command(s) completed successfully.
```

Inserting row into view:

```
/*c) Вставьте новую строку в представление, указав новые данные для ProductModel,
Culture и ProductDescription. Триггер должен добавить новые строки в таблицы Production.ProductModel,
Production.ProductModelProductDescriptionCulture, Production.Culture и Production.ProductDescription.
Обновите вставленные строки через представление. Удалите строки.*/
INSERT INTO dbo.ProductModelClusterView(
   CultureID, C_Name, PM_Name, [Description]
)
VALUES ('EPL', 'English Prenier Ligue', 'Foolball', 'We are going to the Wembely');
```

91 %

Messages

```
(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)
```

Updating row of view:

```
UPDATE dbo.ProductModelClusterView
   SET C_Name = 'BundesLigue',
       PM_Name = 'BasketBall',
       [Description] = 'Super Deutschland'
   WHERE CultureID = 'EPL' AND
         ProductModelID = IDENT_CURRENT('Production.ProductModel') AND
         ProductDescriptionID = IDENT_CURRENT('Production.ProductDescription');
```

91 %

Messages

```
(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)
```

Deleting row from view:

```
DELETE FROM dbo.ProductModelClusterView
   WHERE CultureID = 'EPL' AND
         ProductModelID = IDENT_CURRENT('Production.ProductModel') AND
         ProductDescriptionID = IDENT_CURRENT('Production.ProductDescription');
```

91 %

Messages

```
(1 row(s) affected)
```

Task 5.1 Var4

```sql
/*Создайте scalar-valued функцию, которая будет принимать в качестве входного
 параметра id заказа (Sales.SalesOrderHeader.SalesOrderID) и возвращать максимальную
  цену продукта из заказа (Sales.SalesOrderDetail.UnitPrice).*/
 CREATE FUNCTION dbo.GetMaxCost(@SalesOrderID [int])
 RETURNS money
 WITH EXECUTE AS CALLER
 AS
 BEGIN
   DECLARE @resultValue money;
   SET @resultValue = (SELECT MAX(SOD.UnitPrice) FROM Sales.SalesOrderHeader AS SOH
                       JOIN Sales.SalesOrderDetail AS SOD
                           ON SOH.SalesOrderID = SOD.SalesOrderID
                       WHERE @SalesOrderID = SOH.SalesOrderID);
   RETURN(@resultValue);
 END;
 GO
```

100 %

Messages

```
Command(s) completed successfully.
```

```sql
CREATE FUNCTION dbo.GetRowsByIdAndCount(
   @ProductID [int],
   @rowCount [int]
)
RETURNS TABLE
AS RETURN(
   SELECT
       ProductID,
       LocationID,
       Quantity,
       Bin,
       Shelf,
       rowguid,
       ModifiedDate
   FROM (
       SELECT
           ProductID,
           LocationID,
           MAX(Quantity) OVER(PARTITION BY ProductID) AS Quantity,
           Bin,
           Shelf,
           rowguid,
           ModifiedDate,
           ROW_NUMBER() OVER (PARTITION BY ProductID ORDER BY ProductID) AS rn
       FROM Production.ProductInventory
       WHERE ProductID = @ProductID
             AND Shelf = 'A'
   ) AS Result WHERE rn <= @rowCount
);
GO
```

83 %

Messages

```
Command(s) completed successfully.
```

100 %

Cross Apply `GetRowsByIdAndCount`

```
/*Вызовите функцию для каждого продукта, применив оператор CROSS APPLY.
 Вызовите функцию для каждого продукта, применив оператор OUTER APPLY.*/
SELECT *
FROM Production.Product AS P
CROSS APPLY dbo.GetRowsByIdAndCount(P.ProductID, 2);
```

100 %

Results | Messages

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice	Size	SizeUnitMeasureCode	W
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00	NULL	NULL	N
2	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00	NULL	NULL	N
3	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0,00	0,00	NULL	NULL	N
4	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0,00	0,00	NULL	NULL	N
5	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0,00	0,00	NULL	NULL	N
6	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0,00	0,00	NULL	NULL	N
7	4	Headset Ball B...	BE-2908	0	0	NULL	800	600	0,00	0,00	NULL	NULL	N
8	4	Headset Ball B...	BE-2908	0	0	NULL	800	600	0,00	0,00	NULL	NULL	N

Query executed successfully. | DESKTOP-AHV1F62\YAHOR_STS (... | DESKTOP-AHV1F62\Yahor ... | AdventureWorks2012 | 00:00:00 | 81 rows

Outer Apply `GetRowsByIdAndCount`

```
SELECT *
FROM Production.Product AS P
OUTER APPLY dbo.GetRowsByIdAndCount(P.ProductID, 2);
```

10 %

Results | Messages

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice	Size	SizeUnitMeasureCode
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00	NULL	NULL
2	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00	NULL	NULL
3	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0,00	0,00	NULL	NULL
4	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0,00	0,00	NULL	NULL
5	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0,00	0,00	NULL	NULL
6	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0,00	0,00	NULL	NULL
7	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0,00	0,00	NULL	NULL
8	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0,00	0,00	NULL	NULL
9	316	Blade	BL-2036	1	0	NULL	800	600	0,00	0,00	NULL	NULL
10	317	LL Crankarm	CA-5965	0	0	Black	500	375	0,00	0,00	NULL	NULL

Query executed successfully. | DESKTOP-AHV1F62\YAHOR_STS (... | DESKTOP-AHV1F62\Yahor ... | AdventureWorks2012 | 00:00:00 | 531 rows

```sql
CREATE FUNCTION dbo.GetRowsByIdAndCount(
   @ProductID [int],
   @rowCount [int]
)
RETURNS @ProductInventary TABLE(
           ProductID int,
           LocationID smallint,
           Quantity int,
           Bin tinyint,
           Shelf nvarchar,
           rowguid uniqueidentifier,
           ModifiedDate datetime
       )
AS
BEGIN
   INSERT INTO @ProductInventary
   SELECT ProductID, LocationID, Quantity, Bin, Shelf, rowguid, ModifiedDate
   FROM (
       SELECT
           ProductID, LocationID,
           MAX(Quantity) OVER(PARTITION BY ProductID) AS Quantity,
           Bin, Shelf, rowguid, ModifiedDate,
           ROW_NUMBER() OVER (PARTITION BY ProductID ORDER BY ProductID) AS countOfRows
       FROM Production.ProductInventory
       WHERE ProductID = @ProductID
             AND Shelf = 'A'
   ) AS Result WHERE countOfRows <= @rowCount
   RETURN;
END;
GO
```

83 %

Messages

```
Command(s) completed successfully.
```

Task 6.1 Var4

```
        ON D.DepartmentID = ED.DepartmentID
        -- if endDate null then get 2019;
        -- minus 1: if employee was sacked in currentYear, then we don't count him)
        WHERE @itemYear BETWEEN YEAR(ED.StartDate) AND ((ISNULL(YEAR(ED.EndDate), YEAR(GETDATE())) - 1)
        SET @itemYear = @itemYear + 1;
    END;

    set @query ='
                    SELECT Name,' + @listOfYears + '
                    FROM #everyYearEmployeesHistory
                    AS DepartmentEmployees
                    PIVOT (
                        COUNT (DepartmentID)
                        FOR currentYear IN (' + @listOfYears + ')
```

83 %

Results | Messages

	Name	2003	2004	2005
1	Document Control	5	5	5
2	Engineering	5	4	6
3	Executive	1	1	1
4	Facilities and Maintenance	1	7	7
5	Finance	11	11	11
6	Human Resources	6	6	6
7	Information Services	10	10	10
8	Marketing	6	6	9
9	Production	149	180	179
10	Production Control	5	5	6
11	Purchasing	2	9	11
12	Quality Assurance	5	6	7
13	Research and Developm...	4	4	4
14	Sales	0	0	11
15	Shipping and Receiving	6	6	6

Task 7.1 Var4

```sql
/* Вывести значения полей [BusinessEntityID], [Name], [AccountNumber]
из таблицы [Purchasing].[Vendor] в виде xml, сохраненного в переменную.
Создать хранимую процедуру, возвращающую таблицу,
заполненную из xml переменной представленного вида.
Вызвать эту процедуру для заполненной на первом шаге переменной.*/
IF OBJECT_ID ( 'dbo.vendorToXML', 'P' ) IS NOT NULL
DROP PROCEDURE dbo.vendorToXML;
GO

CREATE PROCEDURE [dbo].vendorToXML
    @xmlForVendors XML
AS
BEGIN
SELECT [BusinessEntityID] = Node.Data.value('(ID)[1]', 'INT')
        , [Name] = Node.Data.value('(Name)[1]', 'NAME')
        , [AccountNumber] = Node.Data.value('(AccountNumber)[1]', 'AccountNumber')
FROM @xmlForVendors.nodes('/Vendors/Vendor') Node(Data)
    END;

DECLARE @vendorsFromXML XML;
SELECT @vendorsFromXML = (
    SELECT [BusinessEntityID] AS ID, [Name], [AccountNumber]
    FROM [Purchasing].[Vendor]
        FOR XML RAW ('Vendor'), TYPE, ELEMENTS, ROOT ('Vendors')
);

EXEC [dbo].vendorToXML @vendorsFromXML;
```

83 %

Results | Messages

	BusinessEntityID	Name	AccountNumber
1	1492	Australia Bike Retailer	AUSTRALI0001
2	1494	Allenson Cycles	ALLENSON0001
3	1496	Advanced Bicycles	ADVANCED0001
4	1498	Trikes, Inc.	TRIKES0001
5	1500	Morgan Bike Accessories	MORGANB0001