دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

دستور کار آزمایشگاه پایگاه داده‌ها
ترم اول ۹۹-۹۸

استاد درس
دکتر علیرضا بصیری

ویرایش ششم تابستان ۱۳۹۸

تهیه کنندگان:

دانیال اکبری

مهران صادقی

سید میثم غفاری

بازنگری و بازنویسی:

محمد سلیمان‌نژاد

وحید مروج

مهدی جودکی

تحت نظارت

دکتر علیرضا بصیری**

فهرست مطالب

۱. فصل اول : آشنایی با SQL مقدماتی

۱,۱. مقدمه

در این جلسه مطالب مربوط به SQL مقدماتی که در کلاس تدریس شده، مرور می‌شوند. پایگاه داده‌ای به نام University ساخته می‌شود و با مرور دستورات SQL این پایگاه داده، تکمیل می‌شود.

۱,۲. دستورات Data Definition Language

این دستورات برای تعریف جداول، نوع داده‌های موجود در جداول و ویرایش و حذف آن‌ها، استفاده می‌شود.

۱,۲,۱. ساخت جدول

برای ساخت جداول از دستور Create Table استفاده می‌شود. ساختار کلی این دستور بصورت زیر است:

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
);
```

مثال:

```
CREATE TABLE Students
(
  FirstName varchar(20) NOT NULL,
  LastName varchar(30) NOT NULL ,
  StudentNumber char(7) PRIMARY KEY,
  BirthYear int,
);
```

چرا در ساخت جدول بجای فیلد Age از BirthYear استفاده شده؟

۱,۲,۲. قیدها

قیدهایی که در ساخت جداول استفاده می شوند عبارتند از:

NOT NULL	نشان می دهد که ستون مربوطه نمی‌تواند مقدار NULL داشته باشد.
UNIQUE	نشان می دهد که هر سطر از جدول باید مقداری یکتا برای این ستون داشته باشد.
PRIMARY KEY	به عنوان ترکیبی از دو قید قبلی کار می کند و با هر مقدار از آن می توان یک و دقیقاً یک سطر از جدول را مشخص کرد.
FOREIGN KEY	نشان می دهد که مقادیر این ستون باید از بین مقادیر موجود در ستونی مشابه اما در جدولی دیگر، باشند.
CHECK	نشان می دهد که مقادیر این ستون باید شرط خاصی داشته باشند.
DEFAULT	یک مقدار اولیه برای مقادیر این ستون مشخص می کند.

جدول ۱-۱

مثال:

```
CREATE TABLE Departments
(
  Name varchar(20) NOT NULL ,
  ID char(5) PRIMARY KEY,
  Budget numeric(12,2),
  Category varchar(15) Check (Category in
('Engineering','Science'))

);


CREATE TABLE Teachers
(
  FirstName varchar(20) NOT NULL,
  LastName varchar(30) NOT NULL ,
  ID char(7),
  BirthYear int,
  DepartmentID char(5),
  Salary numeric(7,2) Default 10000.00,
  PRIMARY KEY (ID),
  FOREIGN KEY (DepartmentID) REFERENCES Departments(ID),
 );
```

```
CREATE TABLE Students
(
  FirstName varchar(20) NOT NULL,
  LastName varchar(30) NOT NULL ,
  StudentNumber char(7) PRIMARY KEY,
  BirthYear int,
  DepartmentID char(5),
  AdvisorID char(7),
  FOREIGN KEY (DepartmentID) REFERENCES Departments(ID),
  FOREIGN KEY (AdvisorID) REFERENCES Teachers(ID)
);
```

۱,۲,۳. ویرایش ساختار جداول

برای تغییر ساختار جداول از دستور ALTER TABLE با ساختار کلی زیر استفاده می شود:

برای اضافه کردن ستون:

```
ALTER TABLE table_name
  ADD column_name column_type
```

برای حذف کردن ستون:

```
ALTER TABLE table_name
  DROP COLUMN column_name
```

برای تغییر نوع داده یک ستون:

```
ALTER TABLE table_name
  ALTER COLUMN column_name column_type
```

مثال:

```
ALTER TABLE Departments
ALTER COLUMN Name varchar(50)
```

۱,۳. دستورات Data Manipulation Language

۱,۳,۱. وارد کردن اطلاعات به یک جدول

برای وارد کردن اطلاعات به یک جدول از دستور INSERT با ساختار کلی زیر استفاده می شود:

```
INSERT INTO table (column1, column2, ... ) VALUES (expression1,
expression2, ... );
```

مثال:

```
INSERT INTO Departments (Name, ID, Budget, Category) VALUES
('Electrical & Computer Engineering Department', 'ECE',
1200000.00 , 'Engineering')


INSERT INTO Departments (Name, ID, Budget) VALUES ('Mechanical
Engineering Department', 'ME', 1000000.00)

INSERT INTO Departments (Name, ID, Category) VALUES ('Physics
Department', 'P' , 'Science')
```

۱,۳,۲. دستور SELECT

برای خواندن اطلاعات از پایگاه داده، از دستور SELECT با ساختار کلی زیر استفاده می شود:

```
SELECT expressions
FROM table
```

مثال:

```
SELECT Name, ID FROM Departments
```

نتیجه:

	Name	ID
1	Electrical & Computer Engineering Department	ECE
2	Mechanical Engineering Department	ME
3	Physics Department	P

مثال:

```
SELECT * from Departments
```

نتیجه:

	Name	ID	Budget	Category
1	Electrical & Computer Engineering Department	ECE	1200000.00	Engineering
2	Mechanical Engineering Department	ME	1000000.00	NULL
3	Physics Department	P	NULL	Science

۱,۳,۳. دستور WHERE

با استفاده از این دستور، نتایج پرس و جوی نوشته شده را فیلتر می کنیم.

مثال:

```
SELECT * from Departments WHERE Category <> 'NULL'
```

نتیجه:

	Name	ID	Budget	Category
1	Electrical & Computer Engineering Department	ECE	1200000.00	Engineering
2	Physics Department	P	NULL	Science

۱,۳,۴. دستور ORDER BY

با استفاده از این دستور، نتایج پرس و جوی نوشته شده را مرتب می کنیم.

مثال:

```
1- SELECT * from Departments WHERE Budget>550000 ORDER BY Name
   asc
2- SELECT * from Departments WHERE Budget>550000 ORDER BY Name
   desc
```

نتیجه:

۱-

	Name	ID	Budget	Category
1	Electrical & Computer Engineering Department	ECE	900000.00	Engineering
2	Mechanical Engineering Department	ME	1000000.00	NULL

۲-

	Name	ID	Budget	Category
1	Mechanical Engineering Department	ME	1000000.00	NULL
2	Electrical & Computer Engineering Department	ECE	900000.00	Engineering

۱,۳,۵. دستور UPDATE

برای ویرایش اطلاعات پایگاه داده، از دستور UPDATE با ساختار کلی زیر استفاده می شود:

```
UPDATE table
SET column1 = expression1,
    column2 = expression2,
    ...
WHERE conditions;
```

مثال:

```
UPDATE Departments
SET Category = 'Engineering'
WHERE ID = 'ME';
```

۱,۴. دستور JOIN و انواع آن

هنگام نوشتن برخی پرس‌وجوها تمام اطلاعات مورد نیاز در یک جدول وجود ندارد و باید اطلاعات چند جدول را با هم در اختیار داشته باشیم. همانطور که به یاد دارید، برای این منظوراز دستور JOIN استفاده می‌کنیم که برخی انواع پرکاربرد این دستور را در جدول ۱-۲ بیان می‌کنیم.

نوع JOIN	توضیح
INNER JOIN	در این روش سطرهایی نمایش داده می شوند که در هر دو جدولی که با هم Join شده اند وجود دارند. در واقع رکوردهایی در نتیجه ظاهر می‌شوند که متناظرشان (بر اساس فیلدهایی که JOIN روی آنها انجام شده است) در جدول دیگر هم رکورد وجود داشته باشد.
RIGHT JOIN	در نتیجه‌ی این JOIN کلیه‌ی رکوردهای جدول سمت راست عبارت JOIN وجود دارند و برای رکوردهایی که متناظرشان در جدول سمت چپ رکوردی وجود ندارد، مقدار NULL وجود خواهد داشت.
LEFT JOIN	در نتیجه‌ی این JOIN کلیه‌ی رکوردهای جدول سمت چپ عبارت JOIN وجود دارند و برای رکوردهایی که متناظرشان در جدول سمت راست رکوردی وجود ندارد، مقدار NULL وجود خواهد داشت.
FULL JOIN	در نتیجه‌ی این JOIN کلیه‌ی رکوردهای جدول‌های هر دو سمت عبارت JOIN وجود خواهند داشت.

جدول ۱-۲

نکته: در صورتی که هنگام JOIN برای یک رکورد از جدول اول n رکورد در جدول دوم وجود داشته باشد، در خروجی n رکورد مشاهده می شود.

در مثال زیر ساختار این دستورات و نتایج حاصل از آن‌ها را مرور خواهیم کرد:

فرض کنید محتویات دو جدول PASSENGER (حاوی اطلاعات مسافران) و PASSENGER_PHONE (حاوی شماره تلفن‌های مسافران) به صورت زیر است:

PASSENGER:

SSN	First_Name	Last_Name	Gender
1111111111	Mike	Anderson	Male
2222222222	Julie	Brown	Female
3333333333	Sue	Jones	Female
4444444444	Andrew	James	Male
5555555555	Ben	Mayer	Male

PASSENGER_PHONE:

Passenger_SSN	Passenger_Phone
1111111111	1230000123
2222222222	4560000456
2222222222	7890000789
5555555555	2580000258
5555555555	3690000369

INNER JOIN:

```
SELECT SSN, First_Name, Last_Name, Gender, Passenger_Phone
FROM Passenger INNER JOIN Passenger_Phone ON
(Passenger.SSN=Passenger_Phone.Passenger_SSN)
```

نتیجه:

SSN	First_Name	Last_Name	Gender	Passenger_Phone
1111111111	Mike	Anderson	Male	1230000123
2222222222	Julie	Brown	Female	4560000456
2222222222	Julie	Brown	Female	7890000789
5555555555	Ben	Mayer	Male	2580000258
5555555555	Ben	Mayer	Male	3690000369

RIGHT JOIN

```
SELECT SSN, First_Name, Last_Name, Gender, Passenger_Phone
```

```
FROM Passenger RIGHT JOIN Passenger_Phone ON
(Passenger.SSN=Passenger_Phone.Passenger_SSN)
```

نتیجه:

SSN	First_Name	Last_Name	Gender	Passenger_Phone
1111111111	Mike	Anderson	Male	1230000123
2222222222	Julie	Brown	Female	4560000456
2222222222	Julie	Brown	Female	7890000789
5555555555	Ben	Mayer	Male	2580000258
5555555555	Ben	Mayer	Male	3690000369

LEFT JOIN

```
SELECT SSN, First_Name, Last_Name, Gender, Passenger_Phone
FROM Passenger LEFT JOIN Passenger_Phone ON
(Passenger.SSN=Passenger_Phone.Passenger_SSN)
```

نتیجه:

SSN	First_Name	Last_Name	Gender	Passenger_Phone
1111111111	Mike	Anderson	Male	1230000123
2222222222	Julie	Brown	Female	4560000456
2222222222	Julie	Brown	Female	7890000789
3333333333	Sue	Jones	Female	NULL
4444444444	Andrew	James	Male	NULL
5555555555	Ben	Mayer	Male	2580000258
5555555555	Ben	Mayer	Male	3690000369

FULL JOIN

```
SELECT SSN, First_Name, Last_Name, Gender, Passenger_Phone
FROM Passenger FULL JOIN Passenger_Phone ON
(Passenger.SSN=Passenger_Phone.Passenger_SSN
```

نتیجه:

SSN	First_Name	Last_Name	Gender	Passenger_Phone
1111111111	Mike	Anderson	Male	1230000123
2222222222	Julie	Brown	Female	4560000456
2222222222	Julie	Brown	Female	7890000789
3333333333	Sue	Jones	Female	NULL
4444444444	Andrew	James	Male	NULL

5555555555	Ben	Mayer	Male	2580000258
5555555555	Ben	Mayer	Male	3690000369

- در این مثال استثنائاً نتیجه‌ی FULL JOIN و LEFT JOIN یکسان شد.

۲. فصل دوم : آشنایی با Adventure Works2012 و چند دستور مهم در SQL

در این جلسه ضمن معرفی پایگاه داده AdventureWorks 2012 ، دو ساختار دستور Case و دستوراتی برای کار با مجموعه نتایج معرفی می شود.

۲,۱. معرفی پایگاه داده AdventureWorks 2012

در این آزمایشگاه سعی خواهد شد اکثر مثال‌ها و تمرین‌ها روی این پایگاه داده معروف انجام پذیرند و بنابراین در این بخش با این پایگاه داده و روش نصب آن آشنا خواهیم شد.

۲,۱,۱. روش نصب

برای نصب Adventure Works 2012 چندین روش وجود دارد که یک روش را در اینجا بررسی می‌کنیم:

ابتدا AdventureWorks2012_Database.zip را از سامانه الکترونیکی دروس، دانلود کنید.

محتویات فایل زیپ را در آدرس زیر، کپی کنید:

C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA

سپس در SQL SERVER 2012 در قسمت OBJECT EXPLORER روی DATABASES کلیک راست کنید و گزینه‌ی ATTACH را انتخاب نمایید و در پنجره‌ای که باز می‌شود با کلیک روی ADD فایل mdf کپی شده را انتخاب کنید تا به SQL SERVER اضافه شود.

برای مشاهده‌ی سایر روش‌های نصب Adventure Works 2012 به آدرس زیر بروید:

http://social.technet.microsoft.com/wiki/contents/articles/3735.sql-server-samples-readme.aspx#Readme_for_Adventure_Works_Sample_Databases

۲,۱,۲. آشنایی با Adventure Works 2012

Adventure Works نام یک شرکت بین‌المللی کاملاً فرضی است که محصولات آن انواع دوچرخه و محصولات مرتبط با آن است، و پایگاه داده Adventure Works 2012 هم بر اساس محصولات، مشتریان، سفارش‌ها، کارمندان و ... این شرکت فرضی بنا نهاده شده.

از آنجایی که در این آزمایشگاه عمده مثال‌ها و تمرین‌ها بر روی این پایگاه داده انجام می‌شود، آشنایی کلی با شمای این پایگاه داده ضروری است، از طرفی هم انتظار نمی‌رود با همه‌ی جداول در این پایگاه داده بطور کامل آشنا باشید اما باید در ابتدای این آزمایشگاه با جداول مهم این پایگاه داده آشنا شوید.

- جدول SalesOrderHeader

این جدول حاوی اطلاعات کلی یک سفارش است. (چیزی شبیه به سربرگ یک فاکتور)

برخی از مهمترین فیلدهای این جدول، در جدول زیر مشاهده می شود:

SalesOrderID	یک عدد صحیح به عنوان شناسه سفارش
OrderDate	تاریخ ثبت سفارش
Status	یک عدد صحیح، نشانگر وضعیت سفارش (۱=در حال پردازش، ۲= تأیید شده و ...)
CustomerID	شناسه صاحب سفارش (کلید خارجی از جدول Customer)
TerritoryID	شناسه محلی که سفارش در آن ثبت شده (کلید خارجی از جدول SalesTerritory)
SubTotal	مجموع قیمت کالاهای موجود در سفارش
TotalDue	قیمت نهایی سفارش برای مشتری (مجموع قیمت کالاهای موجود در سفارش + هزینه ارسال سفارش + مالیات)

جدول ۲-۱

- جدول SalesOrderDetail

این جدول حاوی اطلاعات کالاهای موجود در یک سفارش است. (ردیف‌های فاکتور)

برخی از مهمترین فیلدهای این جدول، در جدول زیر مشاهده می شود:

SalesOrderID	شناسه سفارش (کلید خارجی از جدول SalesOrderHeader)
SalesOrderDetailID	کلید اصلی جدول
OrderQty	تعداد کالای سفارش داده شده
ProductID	شناسه کالای سفارش داده شده
UnitPrice	قیمت واحد کالای سفارش داده شده
LineTotal	قیمت کل این سطر (قیمت واحد کالا x تعداد کالا)

جدول ۲-۲

- جدول Product

این جدول حاوی اطلاعات کالاهای AdventureWorks است.

برخی از مهمترین فیلدهای این جدول، در جدول زیر مشاهده می شود:

ProductID	شناسه کالا (کلید اصلی جدول)
Name	نام کالا
Color	رنگ کالا
StandardCost	قیمت تمام شده کالا برای AdventureWorks
ListPrice	قیمت کالا برای فروش
Size	اندازه کالا
Weight	وزن کالا
DaysToManufacture	تعداد روزهای مورد نیاز برای تولید کالا
ProductLine	نوع کالا (جاده، کوهستان و ...)
Style	نوع کالا (مخصوص آقایان، مخصوص بانوان، قابل استفاده برای همه)

جدول ۲-۳

- جدول SalesTerritory

این جدول حاوی اطلاعات مکان‌هایی است که AdventureWorks در آن مکان‌ها فعالیت دارد.

اطلاعات بیشتر در مورد این جداول و سایر جداول مهم این پایگاه داده را حتماً در آدرس زیر مطالعه کنید.همچنین در آینده نیز برای نوشتن پرس‌وجوهای مختلف با مراجعه به آدرس زیر مطمئن شوید که اطلاعات را از جداول صحیح استخراج می‌کنید:

http://technet.microsoft.com/en-us/library/ms124438(v=sql.100).aspx

۲,۲. عملگرها بر روی مجموعه نتایج (result set operators)

همانطور که می‌دانید حاصل هر پرس و جو بصورت ... select ... from ، یک مجموعه جواب است. در این بخش با عملگرهایی آشنا می‌شویم که روی این مجموعه نتایج کار می‌کنند.

۲,۲,۱. اجتماع (UNION)

همانطور که از اسمش پیداست این عملگر اجتماع دو مجموعه جواب را به عنوان خروجی بر می‌گرداند. برای استفاده از آن، عبارت "UNION" را در میان دو select statement قرار می‌دهیم و اجتماع آن‌ها را (بدون در نظر گرفتن تکرارها) در خروجی خواهیم داشت.

نحوه استفاده از عملگر اجتماع :

select_statement UNION *select_statement*

توجه: بدیهی است که دو مجموعه جواب حاصل از دو select statement که در دو طرف عبارت "UNION" قرار می‌گیرند دارای ساختار کاملاً یکسان باشند. (تعداد و ترتیب قرار گرفتن ستون‌ها یکسان و نوع داده‌ی ستون‌های متناظر هم یکسان) و البته این مطلب برای همه‌ی عملگرهایی که در ادامه به آن‌ها خواهیم پرداخت صادق است.

به عنوان مثال دو جدول زیر را در نظر بگیرید:

Coloumn A char(3)	Coloumn B int
ABC	1
DEF	2
GHI	3

Table2

Coloumn C char(3)	Column D int
GHI	3
JKL	4
MNO	5

Table۱

در مثال زیر اجتماع تمام رکوردهای دو جدول فوق مشاهده می‌شود:

```
SELECT * FROM Table1
UNION
SELECT * FROM Table2
```

و نتیجه برابر خواهد بود با:

```
ColumnA  ColumnB
-------  --------
abc      1
def      2
ghi      3
jkl      4
mno      5
```

همانطور که مشاهده کردید، union تکرار را در نظر نمی‌گیرد. اگر بخواهیم اجتماع دو مجموع جواب را با احتساب تکرارها داشته باشیم از عبارت union all استفاده می کنیم. مثال قبل را با union all امتحان می‌کنیم:

```
SELECT * FROM Table1
UNION ALL
SELECT * FROM Table2
```

و نتیجه برابر خواهد بود با:

```
ColumnA  ColumnB
-------  --------
abc      1
def      2
ghi      3
ghi      3
jkl      4
mno      5
```

۲,۲,۲. اشتراک (INTERSECT)

این عملگر هم دقیقاً مثل عملگر union بر روی دو مجموعه جواب کار می‌کند با این تفاوت که اشتراک دو مجموعه جواب را بر می‌گرداند. به مثال زیر توجه کنید:

```
SELECT * FROM Table1
INTERSECT
SELECT * FROM Table2
```

و نتیجه برابر خواهد بود با:

```
ColumnA  ColumnB
-------  --------
ghi      3
```

۲,۲,۳. تفاضل (EXCEPT)

این عملگر همانطور که در مثال زیر مشاهده می‌شود، حاصل تفاضل دو مجموعه نتیجه را به عنوان خروجی برمی-گرداند.

```
SELECT * FROM Table1
EXCEPT
SELECT * FROM Table2
```

و نتیجه برابر خواهد بود با:

```
ColumnA  ColumnB
-------  --------
abc      1
def      2
```

۲,۳. دستور CASE

به کمک این دستور می‌توان تعدادی شرط را بررسی نمود و یکی از نتایج ممکن را انتخاب کرد. این دستور در دو قالب استفاده می‌شود:

۲,۳,۱. دستور CASE در قالب ساده

دستور CASE در این قالب، یک عبارت را با تعدادی عبارت دیگر مقایسه می‌کند و سپس در مورد نتیجه تصمیم‌گیری می‌کند.قالب این دستور بصورت زیر است:

```
CASE input_expression
       WHEN when_expression1 THEN result_expression1
       WHEN when_expression2 THEN result_expression2
       WHEN when_expression3 THEN result_expression3
        …
       WHEN when_expressionN THEN result_expressionN

    [
        ELSE else_result_expression
    ]
END
```

مثال: در این مثال با استفاده از قالب ساده‌ی دستور CASE در میان کتاب‌های یک انتشارات، دسته‌بندی به نحوی تغییر داده شده که برای مشتریان قابل‌فهم‌تر باشد:

```
SELECT
   CASE type
      WHEN 'popular_comp' THEN 'Popular Computing'
      WHEN 'mod_cook' THEN 'Modern Cooking'
      WHEN 'business' THEN 'Business'
      WHEN 'psychology' THEN 'Psychology'
      WHEN 'trad_cook' THEN 'Traditional Cooking'
      ELSE 'Not yet categorized'
   END AS Category,
Title, Price
FROM titles
WHERE price IS NOT NULL
ORDER BYCategory
```

و نتیجه برابر است با:

```
Category            Title                          Price
------------------- ------------------------------ -------
Business            Cooking with Computers: Surrep 11.95
Business            Straight Talk About Computers  19.99
Business            The Busy Executive's Database  19.99
Business            You Can Combat Computer Stress 2.99
Modern Cooking      Silicon Valley Gastronomic Tre 19.99
Modern Cooking      The Gourmet Microwave          2.99
Popular Computing   But Is It User Friendly?       22.95
Popular Computing   Secrets of Silicon Valley      20.00
Psychology          Computer Phobic AND Non-Phobic 21.59
Psychology          Emotional Security: A New Algo 7.99
Psychology          Is Anger the Enemy?            10.95
Psychology          Life Without Fear              7.00
Psychology          Prolonged Data Deprivation: Fo 19.99
Traditional Cooking Fifty Years in Buckingham Pala 11.95
Traditional Cooking Onions, Leeks, and Garlic: Coo 20.95
Traditional Cooking Sushi, Anyone?                 14.99
```

۲,۳,۲. دستور CASE در قالب جستجویی

چندین عبارت BOOLEAN را بررسی می‌کند و بر این اساس در مورد نتیجه تصمیم‌گیری می‌کند. قالب این دستور بصورت زیر است:

```
CASE
    WHEN Boolean_expression1 THEN result_expression1
    WHEN Boolean_expression2 THEN result_expression2
    WHEN Boolean_expression3 THEN result_expression3
    …
    WHEN Boolean_expressionN THEN result_expressionN
    [
        ELSE else_result_expression
    ]
END
```

مثال: در این مثال با استفاده از قالب جستجویی دستور CASE برای کتاب‌ها دسته بندی جدیدی بر اساس قیمت آن‌ها صورت پذیرفته:

```
SELECT
   CASE
      WHEN price IS NULL THEN 'Not yet priced'
      WHEN price < 10 THEN 'Very Reasonable Title'
      WHEN price >= 10 and price < 20 THEN 'Coffee Table Title'
      ELSE 'Expensive book!'
   END AS "Price Category",
Title
FROM titles
ORDER BY price
```

و نتیجه برابر خواهد بود با:

```
Price Category        Title
--------------------- --------------------
Not yet priced        The Psychology of Co
Not yet priced        Net Etiquette
Very Reasonable Title You Can Combat Compu
Very Reasonable Title The Gourmet Microwav
Very Reasonable Title Life Without Fear
Very Reasonable Title Emotional Security:
Coffee Table Title    Is Anger the Enemy?
Coffee Table Title    Cooking with Compute
Coffee Table Title    Fifty Years in Bucki
Coffee Table Title    Sushi, Anyone?
Coffee Table Title    The Busy Executive's
Coffee Table Title    Straight Talk About
Coffee Table Title    Silicon Valley Gastr
Coffee Table Title    Prolonged Data Depri
Expensive book!       Secrets of Silicon V
Expensive book!       Onions, Leeks, and G
Expensive book!       Computer Phobic AND
Expensive book!       But Is It User Frien
```

۲,۴. توابع تجمیعی در SQL

این نوع از توابع بر روی مجموعه ای از داده ها عمل می‌کنند (مجموعه مقادیر یک ستون) و تنها یک مقدار را بر می‌گرداند. برخی از توابع پرکاربرد این گروه عبارت اند از :

نام تابع	توضیحات
AVG( )	**میانگین مقادیر را بر می گرداند**
COUNT( )	**تعداد سطرها را بر می گرداند**
SUM( )	**مجموع مقادیر یک لیست را بر می گرداند**
MAX( )	**بزرگترین مقدار را بر می گرداند**
MIN( )	**کوچکترین مقدار را بر می گرداند**
DISTINCT( )	**مقادیر تکراری را از لیست انتخاب حذف می‌کند**
Group By( )	**به منظور دسته بندی اطلاعات به کار می‌رود**

جدول ۲-۲

مثلاً تابع COUNT() تعداد مقادیر موجود در یک لیست را بر می گرداند، مقادیر NULL و مقادیر تکراری را نیز شامل می‌شود. به منظور حذف مقادیر تکراری می توان از واژه DISTINCT استفاده کرد.

ساختار :

```
SELECT COUNT(column_name)
FROM table_name;

SELECT COUNT(DISTINCT column_name)
FROM table_name;
```

مثال :

```
SELECT COUNT(*)
FROM [Sales].[Customer];
```

۲,۵. Group By و Having

گاهی اوقات به منظور گزارش گیری و دستیابی به اطلاعات هوشمند تجاری نیاز به خلاصه کردن اطلاعات وجود دارد. با استفاده از دستورها می توانید اطلاعات را دسته بندی کرده و خلاصه‌ای از اطلاعات یک فیلد خاص، در هر دسته را در یک مقدار نشان دهید.

مثال:

```
SELECT [CustomerID], avg([SubTotal]) as Customer_AVG_Pay
FROM [Sales].[SalesOrderHeader]
GROUP BY [CustomerID]
```

نتیجه:

Results | Messages

	CustomerID	Customer_AVG_Pay
1	14324	1707.1427
2	22814	4.99
3	11407	53.99
4	28387	583.97
5	19897	596.96
6	15675	2402.1266
7	24165	1523.42
8	27036	7.28
9	18546	29.48
10	11453	2725.66
11	17195	1665.3337

در این پرس و جوها، هر فیلدی که در مقابل Select آمده اما مؤلفه ی تابع تجمیعی نیست، باید به عنوان مؤلفه Group By ظاهر شود.

مثال :

```
SELECT [TerritoryID]
      ,max([SalesQuota]) max_SalesQuota
      ,avg([Bonus]) avg_Bonus
  FROM [AdventureWorks2012].[Sales].[SalesPerson]
  group by [TerritoryID]
```

نتیجه:

	TerritoryID	max_SalesQuota	avg_Bonus
1	NULL	NULL	0.00
2	1	300000.00	4133.3333
3	2	300000.00	4100.00
4	3	250000.00	2500.00
5	4	250000.00	2775.00
6	5	300000.00	6700.00
7	6	250000.00	2750.00
8	7	250000.00	985.00
9	8	250000.00	75.00
10	9	250000.00	5650.00

وقتی می خواهید برای مقدار یک تابع تجمیعی در پرس و جو شرطی تعیین کنیم، باید بجای where از having استفاده کنید.

مثال:

```
SELECT [TerritoryID]
      ,max([SalesQuota]) max_SalesQuota
      ,avg([Bonus]) avg_Bonus
  FROM [AdventureWorks2012].[Sales].[SalesPerson]
  group by [TerritoryID]
  having max([SalesQuota]) > 30000
```

نتیجه:

	TerritoryID	max_SalesQuota	avg_Bonus
1	1	300000.00	4133.3333
2	2	300000.00	4100.00
3	3	250000.00	2500.00
4	4	250000.00	2775.00
5	5	300000.00	6700.00
6	6	250000.00	2750.00
7	7	250000.00	985.00
8	8	250000.00	75.00
9	9	250000.00	5650.00
10	10	250000.00	5150.00

۲,۶. تمرین

۱- برای adventure worker 2012 پرس و جویی بنویسید اطلاعات کلی سفارشات در حال پردازشی را برگرداند که مبلغ آن ها بین ۱۰۰۰۰۰ و ۵۰۰۰۰۰ است و محل سفارش از کشور فرانسه یا یکی از کشورهای منطقه آمریکای شمالی است.

۲- برای Adventure Works 2012 پرس و جویی بنویسید که برای هر سفارش، شماره شناسایی سفارش، شماره شناسایی مشتری، مبلغ سفارش، تاریخ سفارش و نام محلی که سفارش در آن ثبت شده را در خروجی نشان دهد.

۳- برای adventure worker 2012 پرس و جویی بنویسید که نشان دهد هر کالا در کدام منطقه بیشترین تعداد سفارش را داشته است.

۴- در adventure worker 2012 جدولی با نام NAmerica_Sales ایجاد کنید که شمای آن مطابق با شمای پرس جوی تمرین شماره ۱ باشد .

سپس از بین رکوردهای موجود در خروجی تمرین ۱ ، فقط رکوردهای مربوط به کشورهای منطقه آمریکای شمالی را وارد این جدول نمایید .

یک ستون به جدول فوق اضافه کنید که نوع داده ای آن char(4) باشد و نتوان چیزی جز LOW یا High یا Mid در آن نوشت .

سپس کاری کنید که مقدار این ستون برای سفارش هایی که مبلغ آن از مبلغ میانگین سفارشات منطقه آمریکای شمالی بیشتر است High ، برای سفارش هایی که مبلغ آن ها با مبلغ میانگین سفارشات منطقه آمریکای شمالی برابر است Mid و برای سایر سفارش ها Low باشد.

۵- با اجرای پرس و جوی زیر میزان حقوق کارمندان را به ازای هر ساعت کار همراه با BusinessEntityID هر کارمند خواهید داشت:

```
USE AdventureWorks2012
GO
SELECT BusinessEntityID ,max(Rate)FROM HumanResources.EmployeePayHistory
   GROUPBY BusinessEntityID

ORDERBY BusinessEntityID
```

حال پرس و جویی بنویسید که نتیجه‌ی آن دارای سه ستون است. ستون اول BusinessEntityID هر کارمند، ستون دوم شامل مقادیر جدید حقوق کارکنان است که این‌گونه محاسبه می‌شود:

طیف حقوق پرداختی را بصورت صعودی مرتب کنید و آن را به چهار قسمت تقسیم کنید:

- ✓ کارمندانی که میزان حقوقشان به ازای هر ساعت کار جز یک چهارم اول طیف است، (کسانی که کمترین حقوق را دارند) حقوقشان ۲۰ درصد افزایش یابد.
- ✓ کارمندانی که میزان حقوقشان به ازای هر ساعت کار جز یک چهارم دوم طیف است، حقوقشان ۱۵ درصد افزایش یابد.
- ✓ کارمندانی که میزان حقوقشان به ازای هر ساعت کار جز یک چهارم سوم طیف است، حقوقشان ۱۰ درصد افزایش یابد و بقیه ، حقوقشان ۵ درصد افزایش یابد.

و ستون سوم سطح دریافتی هر کارمند را نشان می‌دهد و بدین‌صورت محاسبه می‌شود:

- ✓ کارمندانی که میزان حقوقشان (قبل از اضافه شدن درصدی به آن) به ازای هر ساعت کار کمتر از ۲۹ بوده سطح ۳
- ✓ کارمندانی که میزان حقوقشان (قبل از اضافه شدن درصدی به آن) به ازای هر ساعت کار بین۲۹ و ۵۰ بوده سطح
- ✓ و بقیه کارمندان در سطح ۱ هستند.
- ✓ نمونه خروجی مورد نظر در شکل ۲-۱ مشاهده می شود.

	BID	NewSalary	LEVEL
1	1	131.775000	1
2	2	69.807650	1
3	3	49.759580	2
4	4	34.323130	2
5	5	37.596145	2
6	6	37.596145	2
7	7	58.052920	1
8	8	46.995210	2
9	9	46.995210	2
10	10	48.852920	2
11	11	34.615440	3
12	12	30.000000	3
13	13	30.000000	3
14	14	41.466355	2

شکل ۲-۱

۳. فصل سوم: مجوزها، **login** ها و امنیت

۳,۱. مقدمه

در این فصل به بررسی برخی از مهمترین موارد امنیتی در SQL Server می‌پردازیم. بحث پیرامون مسائل امنیتی بسیار گسترده و در مواردی پیچیده است و از حوصله این درس خارج است. بنابراین سعی شده در این فصل به‌طور خلاصه و مفید مفاهیم مهم و اولیه در بحث امنیت، بیان شود.

در ابتدا کار را با آشنایی با مجوزها و مدیریت آن‌ها شروع می‌کنیم و در ادامه به بحث login ها در SQL Server می‌پردازیم و در آخر هم به معرفی انواع نقش‌ها و نحوه مدیریت آن‌ها اشاره خواهیم کرد.

۳,۲. مجوزها[۱]

مجوزها مجموعه کارهایی که یک کاربر می‌تواند انجام دهد را مشخص می‌کنند. این مجوزها شامل مجوز login کردن، مجوز دسترسی به پایگاه داده و جداول و محتویات آن می‌شود.

در اینجا به بررسی مجوزهای مهم برای کار در پایگاه داده‌ها می‌پردازیم که به این مجوزها اصطلاحاً Object Permissions گفته می‌شود.

۳,۲,۱. Object Permissions

در نگاه اول به نظر می‌رسد بعد از دادن مجوز login و مجوز دسترسی به پایگاه داده، مجوز دیگری مورد نیاز نباشد اما در sql server برای استفاده از سطوح دسترسی متفاوت برای کاربران و نقش‌های متفاوت، از Object Permissions استفاده می‌شود که قابل‌ارائه برای جداول، viewها و رویه‌های ذخیره شده هستند. این مجوزها شش نوع مختلف دارند:

۱- SELECT

کاربرانی که این مجوز را داشته باشند، می‌توانند دستور select را برای view یا جدولی که این مجوز را برای آن دارند، اجرا کنند.

۲- INSERT

کاربرانی که این مجوز را داشته باشند، می‌توانند دستور insert را اجرا کنند. (دقت کنید که این مجوز کاملاً مستقل از مجوز select است)

---

[۱] Permissions

۳- UPDATE

کاربرانی که این مجوز را داشته باشند، می‌توانند دستور update را اجرا کنند. (دقت کنید در اینجا هم مثل مجوز insert، کاربری که این مجوز را دارد لزوماً مجوز select را ندارد.)

۴- DELETE

کاربرانی که این مجوز را داشته باشند، می‌توانند دستور delete را اجرا کنند.

۵- REFRENCES

کاربرانی که این مجوز را داشته باشند، می‌توانند از قیودی مانند کلید خارجی استفاده کنند.

۶- EXECUTE

کاربرانی که این مجوز را داشته باشند، می‌توانند رَویه‌ها یا توابع را اجرا کنند.

۳,۲,۲. مدیریت مجوزهای کاربران

برای مدیریت مجوزهای کاربران دو روش وجود دارد. هم می‌توان بصورت گرافیکی و با استفاده از SQL Server Management Studio این کار را انجام داد و هم می‌توان با اجرای کدهای مربوطه، این کار را انجام داد. در ادامه هر دو روش را بررسی خواهیم کرد.

- **مدیریت مجوزهای کاربران با استفاده از SQL Server Management Studio**

برای این منظور در Object Explorer وارد قسمت Security شوید و سپس به قسمت Logins بروید و در آنجا با انتخاب کاربر موردنظر، روی آن کلیک راست کنید و Properties را انتخاب کنید. در هر یک از صفحات پنجره‌ای که باز می‌شود، مجوزهایی وجود دارد که بصورت گرافیکی می‌توانید آن‌ها را لغو یا واگذار کنید

- **مدیریت مجوزهای کاربران با اجرای کد**

برای این منظور ۳ دستور متفاوت وجود دارد که آن‌ها را در ادامه بررسی می‌کنیم.

- **GRANT**

به کمک دستور GRANT مجوزها را به کاربران واگذار می‌کنیم. ساختار کلی این دستور بصورت زیر است:

```
GRANT ALL [ PRIVILEGES ] | permission_name [ ,...n ]
ON [ schema_name ]. object_name [ ( column [ ,...n ] )]
TO <database_principal> [ ,...n ]
[ WITH GRANT OPTION ]
```

در ادامه ساختار فوق را بررسی می‌کنیم.

پس از عبارت GRANT می‌توان عبارت ALL یا ALL PRIVILEGES را قرارداد که به معنی واگذاری کلیه‌ی مجوزهای ممکن برای آن OBJECT خواهد بود. با توجه به اینکه مجوز برای چه چیزی واگذار می‌شود، معنای متفاوتی خواهد داشت. (مثلاً اگر مجوز برای جدول یا VIEW باشد، ALL به معنی INSERT, DELETE, SELECT, UPDATE, REFRENCES خواهد بود اما اگر مجوز برای یک رَویه باشد، ALL به معنی EXECUTE خواهد بود.)

البته می‌توان بجای واگذاری همه‌ی مجوزهای ممکن، فقط نام مجوزهایی را که می‌خواهیم واگذار کنیم را بعد از عبارت GRANT بنویسیم.

پس از مشخص شدن مجوزها، عبارت ON و پس از آن نیز نام OBJECT موردنظر را می‌آوریم. و بعد از آن عبارت TO را نوشته و پس از آن کاربر یا نقشی را که باید مجوز به آن واگذار شود را مشخص می‌کنیم در نهایت هم می‌توان با نوشتن عبارت WITH GRANT OPTION این اجازه را داد تا کاربرانی که این مجوز را می‌گیرند، بتوانند این مجوز را به دیگران هم بدهند.

- **DENY**

به کمک دستور DENY مجوزها را از کاربران سلب می‌کنیم. در واقع این دستور بیان می‌کند که کاربران چه کارهایی را نمی‌توانند انجام دهند. ساختار کلی این دستور بصورت زیر است:

```
DENY  ALL [ PRIVILEGES ] | permission [ ,...n ]
ON [ schema_name ]. object_name [ ( column [ ,...n ] ) ]
TO <database_principal> [ ,...n ]
[ CASCADE ]
```

تنها عبارت جدیدی که در این ساختار به چشم می‌خورد، عبارت CASCADE است که بیانگر آن است که مجوزهای ذکر شده در عبارت DENY، نه تنها از کاربران ذکر شده سلب می‌شود بلکه از سایر کاربرانی که این مجوزها توسط کاربران ذکر شده به آن‌ها داده شده نیز، سلب می‌شود.

توجه! همانطور که می‌دانید یک کاربر ممکن است چند نقش متفاوت داشته باشد. درصورتی‌که یک مجوز را به یکی از نقش‌های یک کاربر واگذار کنیم و همان مجوز را از نقش دیگری که آن کاربر دارد بگیریم، همیشه قدرت دستور DENY بیشتر است و بنابراین آن مجوز از کاربر بطور کل سلب می‌شود.

- **REVOKE**

این دستور اثر دستور GRANT یا DENY که قبلاً اجرا شده را لغو می‌کند. ساختار کلی این دستور بصورت زیر است:

```
REVOKE ALL [ PRIVILEGES ] | permission [ ,...n ]
ON [ schema_name ]. object_name [ ( column [ ,...n ] ) ]
{ FROM | TO } <database_principal> [ ,...n ]
[ CASCADE ]
```

۳,۳. LOGIN

به‌طورکلی می‌توان گفت هر کاربر با دو روش می‌تواند در SQL Server شناخته شود:

۱- استفاده از Windows Authentication

در این روش کاربر وارد Windows می‌شود و SQL Server توسط ارتباط‌های مطمئن، اطلاعات کاربر را از Windows دریافت می‌کند و بدین ترتیب کاربر شناخته می‌شود.

۲- استفاده از SQL Server Authentication

در این روش برای ورود به SQL Server از یک LOGIN که پیشتر ساخته‌ایم استفاده می‌کنیم. به این منظور ابتدا با در محیط SQL Server یک LOGIN ایجاد می‌کنیم و سپس از SQL Server خارج می‌شویم و از اینجا به بعد می‌توانیم با هر دو روش وارد SQL Server شویم. در ادامه این روش را بیشتر توضیح خواهیم داد.

۳,۳,۱. روش ساخت LOGIN

در اینجا دو روش را برای ساخت LOGIN در SQL Server توضیح می‌دهیم:

۱- ساخت LOGIN با اجرای دستور CREATE LOGIN

در این روش با اجرای دستور CREATE LOGIN یک LOGIN را می‌سازیم. این دستور دارای ساختار کلی زیر است:

```
CREATE LOGIN login_name { WITH    PASSWORD = { 'password' } [ MUST_CHANGE ]
[ , SID = sid | DEFAULT_DATABASE = database | DEFAULT_LANGUAGE = language  [ ,...
] ]
| FROM WINDOWS [ WITH DEFAULT_DATABASE = database | DEFAULT_LANGUAGE = language [
,... ] ]}
```

در این ساختار برخی از عبارات مهم را بررسی خواهیم کرد و اطلاعات بیشتر و ساختار مفصل‌تر را می‌توانید از آدرس زیر بدست آورید:

http://msdn.microsoft.com/en-us/library/ms189751.aspx

- [MUST_CHANGE]

آوردن این عبارت بعد از تعیین رمز ورود برای LOGIN اختیاری است و بیانگر این است که کاربر باید پس از اولین ورود با این LOGIN رمز ورود را عوض کند.

- همچنین می‌توان برخی تنظیمات اولیه را برای LOGIN در تعریف در نظر گرفت مثل SID = sid (یک شناسه برای LOGIN)، DEFAULT_DATABASE (تعیین پایگاه داده پیش‌فرض برای این LOGIN)، DEFAULT_LANGUAGE تعیین زبان پیش‌فرض LOGIN و ...
- FROM

آوردن این عبارت به معنای آن است که این لاگین ویژه‌ی SQL Server نیست بلکه مربوط به منبع دیگری است که اجازه‌ی دسترسی به SQL Server به آن داده شده.

۲- ساخت LOGIN با استفاده از SQL Server Management Studio

بدین منظور در قسمت Object Explorer شاخه‌ی Security را انتخاب کرده و روی زیرشاخه‌ی Logins کلیک راست کرده و New Login را انتخاب کنید.

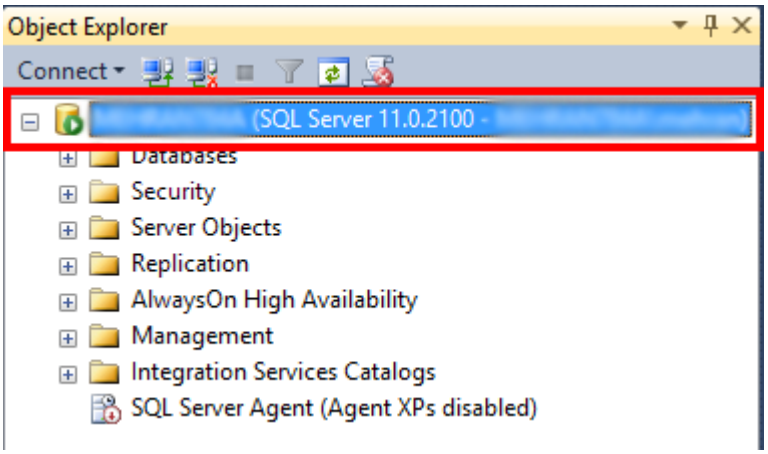
در پنجره‌ای که باز می‌شود ابتدا گزینه‌ی SQL Server Authentication را انتخاب نمایید و سپس مشخصات login موردنظرتان را تنظیم نمایید و در نهایت Ok را بزنید تا login ساخته شود.

توجه!

پس از ساخت Login لازم است مراحل زیر را طی کنید:

- روی SQL Server Instance کلیک راست کنید و گزینه Properties و سپس Security را انتخاب کنید.

(SQL Server Instance در شکل ۵-۱، مشخص شده است)



شکل ۳-۱

- از پنجره بازشده گزینه‌ی SQL Server and Windows authentication mode را انتخاب کنید.
- حال سرویس SQL Server را (از قسمت سرویس‌های ویندوز) restart کنید.

۳,۴. نقش‌ها[۲]

هر نقش در واقع مجموعه ای از مجوزهای داده شده یا گرفته شده، است که به کاربران اختصاص داده می‌شود.در هر لحظه به یک کاربر می‌توان چندین نقش را اختصاص داد. نقش‌ها را می توان به دو دسته تقسیم کرد:

۱- نقش‌های مربوط به Server [۳]

۲- نقش‌های مربوط به پایگاه داده[۴]

۳,۴,۱. نقش‌های مربوط به Server

این نقش‌ها که درون SQL Server موجود هستند بیشتر به منظور نگهداری سیستم و انجام وظایف مدیریتی مربوط به server و همچنین اعطای مجوزهای مربوط به کارهای مستقل از پایگاه داده‌ها (مثل ساخت login) استفاده می‌شوند. ۹ نقش مربوط به Server فیکس شده در SQL Server را در زیر بررسی می‌کنیم.

نقش	توضیح
Sysadmin	صاحب این نقش می‌تواند هر عملی را در SQL Server انجام دهد.
Serveradmin	صاحب این نقش توانایی انجام تنظیمات مربوط به پیکربندی server و shutdown کردن server را دارد.
Setupadmin	صاحب این نقش توانایی مدیریت linked Server ها را دارد.
Securityadmin	این نقش توانایی مدیریت loginها، و ساخت مجوزهای پایگاه داده را به کاربران می‌دهد.
Processadmin	صاحبان این نقش توانایی مدیریت processهایی که روی موتور پایگاه داده اجرا می‌شوند را دارند
Dbcreator	توانایی صاحبان این نقش به ایجاد، تغییر و حذف پایگاه‌های داده محدود می‌شود.
Diskadmin	صاحب این نقش توانایی مدیریت فایل‌های مرتبط با server را دارد.
Bulkadmin	صاحب این نقش اجازه‌ی ورود وسیع اطلاعات (اجرای دستور Bulk Insert) را دارد.
Public	همه‌ی login ها در SQL Server بصورت پیش‌فرض این نقش را دارند.

جدول ۳-۱

از SQL Server 2012 امکانی اضافه شد که در نتیجه‌ی آن کاربران می‌توانند نقش‌های مربوط به سرور را خود چنان تعریف کنند که متناسب با نیازشان باشد. برای این منظور سه گام زیر باید پیموده شود:

---

[۲] Roles

[۳] Server Roles

[۴] Database Roles

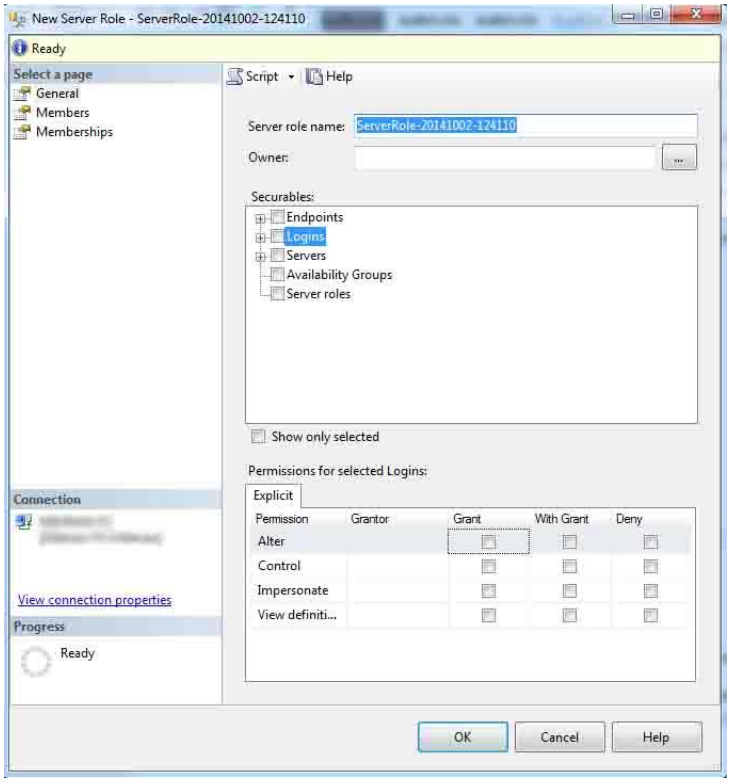
۱- ساخت یک نقش مربوط به Server

۲- تخصیص مجوزهای مربوط به Server به نقش ساخته شده

۳- تخصیص نقش به loginهای موردنظر

در محیط SQL Server Management Studio هم با پیمودن گام‌های زیر می‌توان نقش مربوط به Server موردنظر را ساخت:

۱- در قسمت Object Explorer بخش Security را باز کنید، سپس روی Server Roles کلیک راست کنید و New Server Role را انتخاب کنید. تا پنجره‌ی زیر باز شود:



شکل ۳-۲

۲- در قسمت Server Role Name نامی برای نقش موردنظر وارد کنید.

۳- در قسمت Securables مشخص کنید که این نقش در کدام موارد باید مجوز داشته باشد و در قسمت Permissions مربوطه، مجوزهای موردنظر برای نقش را انتخاب کنید.

۴- سپس در صفحه‌ی Membersهمین پنجره، Add را بزنید و سپس با کلیک روی Browse ، می-توانید loginهای موجود و قابل تخصیص به نقش موردنظرتان را ببینید. پس از این می‌توانید نقش موردنظر را بسازید.

۳,۴,۲. نقش‌های مربوط به پایگاه داده

این نقش‌ها از نظر دامنه[۵] به یک پایگاه داده محدود می‌شوند. یعنی داشتن یک نقش پایگاه داده‌ای در یک پایگاه داده، به معنای داشتن آن نقش در سایر پایگاه داده‌ها نیست. ۹ نقش مربوط به پایگاه داده فیکس شده در SQL Server را در ادامه بررسی می‌کنیم.

نقش	توضیح
db_owner	این نقش را باید به کسانی داد که می‌خواهیم کنترل کامل روی پایگاه داده داشته باشند.
db_securityadmin	صاحب این نقش می‌تواند مدیریت کلیه‌ی نقش‌های مربوط به پایگاه داده و تخصیص مجوزها در آن پایگاه داده را انجام دهد.
db_accessadmin	صاحب این نقش مدیریت تخصیص کاربران و loginها به پایگاه داده مربوطه را انجام می‌دهد.
db_backupoperator	صاحب این نقش می‌تواند از پایگاه داده نسخه پشتیبانی[۶] تهیه کند.
db_ddladmin	صاحب این نقش توانایی اجرای دستورات DDL را در پایگاه داده مربوطه خواهد داشت.
db_datawriter	صاحب این نقش می‌تواند عملیات Insert, Delete, Update را روی جداول پایگاه داده مربوطه، انجام دهد.
db_datareader	صاحب این نقش می‌تواند کلیه داده‌های جداول پایگاه داده مربوطه را بخواند.
db_denydatawriter	صاحب این نقش **نمی‌تواند** عملیات Insert, Delete, Update را روی جداول پایگاه داده مربوطه، انجام دهد.
db_denydatareader	صاحب این نقش **نمی‌تواند** هیچ یک از داده‌های جداول پایگاه داده مربوطه را بخواند.

جدول ۳-۲

اما معمولاً این نقش‌های فیکس شده برای شروع کار هستند و بخش عمده‌ای از کار امنیتی یک پایگاه داده مربوط به تشخیص و ساخت نقش‌های مختلف برای پایگاه داده است. برای این نقش‌ها هم دقیقاً مثل آنچه که برای مجوزهای کاربران ذکر شد، می‌توان مجوزها را با دستورات GRANT, REVOKE, DENY مدیریت کرد.

---

[۵] Scope

[۶] Backup

برای ساخت یک database role با استفاده از SQL Server management studio در قسمت Object Explorer پایگاه داده موردنظرتان را باز کنید و در بخش Security مربوط به آن و در زیر بخش Role روی Database Roles کلیک راست کنید و گزینه New Database Role را انتخاب کنید و در پنجره‌ای که باز می‌شود نام موردنظر را انتخاب کنید (سایر موارد در این پنجره اختیاری هستند) و سپس OK را بزنید.

تمرین پژوهشی:

در مورد Application Role ها تحقیق کنید و موارد استفاده و نحوه استفاده از آن‌ها (دستورات T-SQL لازم برای تعریف و استفاده از آن‌ها) را بنویسید.

۳,۵. تمرین

۱- گام اول: ابتدا در SQL Server یک login بسازید.

گام دوم: نقشی به نام Role1 بسازید که توانایی مدیریت کامل پایگاه داده‌ها (ایجاد، حذف، تغییر، attach, detach) را داشته باشد.

گام سوم: نقش Role1 را به login ساخته شده در گام اول مرتبط سازید.

گام چهارم: با login ساخته شده در گام اول وارد SQL Server شوید و در پایگاه داده AdventureWorks2012 جدولی بسازید و در آن داده‌هایی وارد کنید و آن‌ها را بخوانید.

- تحقیق کنید که چگونه می‌توانید AdventureWorks2012 را یک‌بار حذف کنید و دوباره آن را Attach کنید.

۲- گام اول: در پایگاه داده AdventureWorks2012 کاربری با نقش Role2 را تعریف کنید که به داده‌های جداول هیچ دسترسی نداشته باشد و فقط بتواند مجوزها را مدیریت کند.

گام دوم: اجازه‌ی خواندن داده‌ها را به Role2 بدهید.

۳,۶. منابع

Professional Microsoft SQL Server 2008 Programming
Wiley.Microsoft.SQL.Server.2008.Bible.Aug.2009
Training Kit (Exam 70-462)_ Administering Microsoft SQL Server 2012 Databases

۴. فصل چهارم : SQL پیشرفته

۴,۱. مقدمه

در این فصل با دستورات پیشرفته‌تر زبان SQL آشنا می‌شویم. در ابتدا به بررسی توابع AGGREGATE و سپس پنجره‌بندی در SQL SERVER می‌پردازیم و در ادامه امکاناتی را در SQL SERVER معرفی می‌کنیم که به کمک آن‌ها نتایج پرس و جوها را می‌توان به روش‌های مختلف دسته بندی کرد تا برای تجزیه و تحلیل و تصمیم‌گیری‌های مدیریتی مناسب باشند.

۴,۲. توابع تجمیعی در SQL Server[۷]

این نوع از توابع بر روی مجموعه ای از داده‌ها عمل می‌کنند و تنها یک مقدار را با محاسبه مقادیر یک ستون بر می‌گرداند. برخی از توابع پرکاربرد این گروه در جدول ۵-۱ مشاهده می شوند.

نام تابع	توضیحات
AVG( )	**میانگین مقادیر را بر می گرداند**
COUNT( )	**تعداد سطرها را بر می گرداند**
SUM( )	**مجموع مقادیر یک لیست را بر می گرداند**
MAX( )	**بزرگترین مقدار را بر می گرداند**
MIN( )	**کوچکترین مقدار را بر می گردان**
DISTINCT( )	**مقادیر تکراری را از لیست انتخاب حذف می‌کند**
Group By( )	**به منظور دسته بندی اطلاعات به کار می‌رود**

جدول ۴-۱

مثلاً تابع COUNT() تعداد مقادیر موجود در یک لیست را بر می‌گرداند، مقادیر NULL و مقادیر تکراری را نیز شامل می‌شود. به منظور حذف مقادیر تکراری می توان از واژه DISTICT استفاده کرد.

ساختار :

```
SELECT COUNT(column_name) FROM table_name;

SELECT COUNT(DISTINCT column_name) FROM table_name;
```

مثال :

```
SELECT COUNT(*)
FROM AdventureWorks2012.HumanResources.Employee;
```

[۷] SQL Aggregate Functions

۴,۳. Group By ( )

گاهی اوقات به منظور گزارش گیری و دستیابی به اطلاعات هوشمند تجاری نیاز به خلاصه کردن اطلاعات وجود دارد. با استفاده از این دستور می توانید اطلاعات را دسته بندی کرده و خلاصه‌ای از اطلاعات یک فیلد خاص، در هر دسته را در یک مقدار نشان دهید.

مثال:

```
SELECT SalesOrderID, SUM(OrderQty)
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
```

رکوردهای مختلف جدول SalesOrderDetail بر اساس فیلد SalesOrderID گروه‌بندی شده و مقدار فیلد OrderQty برای تمام اعضای این گروه با یکدیگر جمع شده است.

در این پرس و جوها، هر فیلدی که در مقابل Select آمده اما مؤلفه ی تابع تجمیعی نیست، باید به عنوان مؤلفه Group By ظاهر شود.

مثال :

```
SELECT SalesOrderID, SUM(OrderQty) AS QtySum, COUNT(SalesOrderID) AS DetailCount
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
```

مثال:

```
SELECT ProductID, SpecialOfferID
FROM Sales.SalesOrderDetail
GROUP BY ProductID, SpecialOfferID
ORDER BY ProductID, SpecialOfferID
```

وقتی می خواهید برای مقدار یک تابع تجمیعی در پرس و جو شرطی تعیین کنیم، باید بجای where از having استفاده کنید.

مثال:

```
SELECT DepartmentID, avg(Salary) as Average_Salary FROM Teachers
GROUP BY DepartmentID
having avg(salary)>11000
```

نتیجه:

	DepartmentID	Average_Salary
1	ECE	11500.000000
2	P	13000.000000

۴,۴. پرس و جو های تودرتو

گاهی برای نوشتن یک پرس و جو، نیاز داریم از چند پرس و جوی تودرتو استفاده کنیم. این پرس و جوها معمولاً به سه روش نوشته می شوند که در ادامه آن ها را به تفکیک بیان می کنیم.

- دسته اول این پرس و جوها، آنهایی هستند که حاوی پرس و جویی بعد از عبارت Where هستند. این پرس و جوها معمولاً بعد از عباراتی مانند (some, all) >,=,< / in/ not in/ unique/ not exists/ exists می آیند.

مثال:

```
select FirstName, LastName, ID
from Teachers
where ID in (select AdvisorID from students)
```

این پرس و جو نام تمام استادانی را بر می گرداند که راهنمای حداقل یک دانشجو هستند.

- دسته دوم پرس‌وجوهای تودرتو شامل آنهایی است که حاوی پرس و جویی بعد از عبارت From هستند.

مثال:

```
select Name, avg(selected_salary)
from (select Name, Salary as selected_salary
      from Teachers inner join Departments on
      (Teachers.DepartmentID = Departments.ID)
where Salary>11000) as s
group by Name
```

این پرس و جو نام تمام دانشکده ها به همراه میانگین حقوق استادانی از آن دانشکده که حقوق بالای ۱۱۰۰۰ دارند را بر می گرداند.

- دسته سوم پرس‌وجوهای تودرتو شامل آنهایی است که از پرس و جویی که توسط عبارت With در ابتدای آن ها تعریف شده، بهره می برند.

مثال:

```
with Payments(value1, value2) as (select DepartmentID,sum(Salary)
from Teachers
group by DepartmentID)
select Budget, value2, Budget-value2
from Departments join Payments on (value1 = ID)
```

در این پرس و جو ابتدا به کمک With جمع حقوق استادان هر دانشکده محاسبه شده و در نهایت، شناسه تمام دانشکده ها به همراه میزان حقوقی که هر دانشکده به استادان پرداخت می کند، بودجه‌ی دانشکده و تفاضل میزان پرداختی از بودجه، نمایش داده می شود.

Windowing .۴,۵

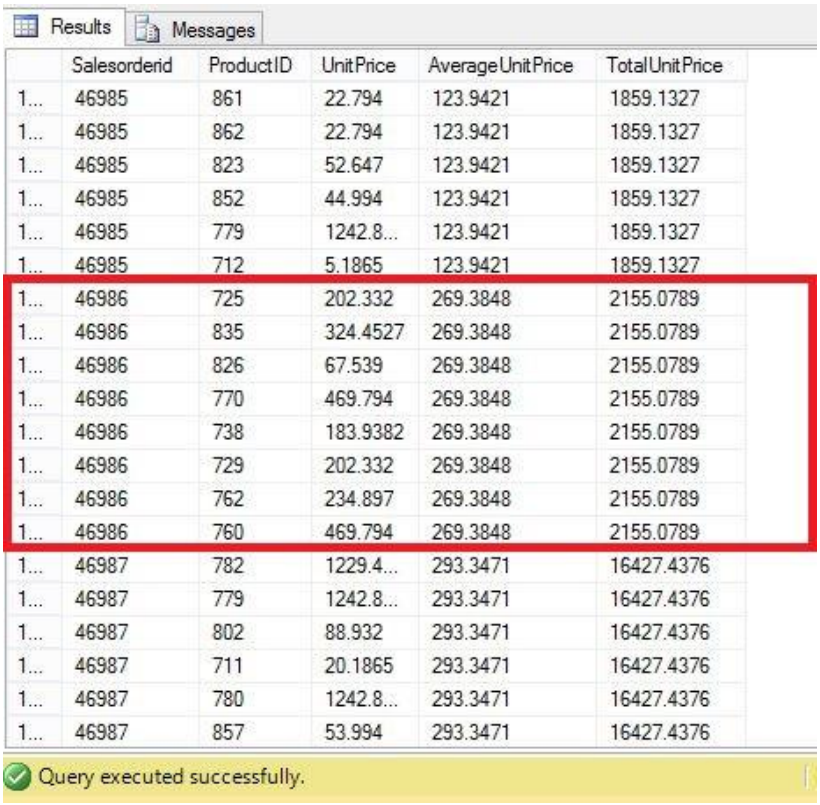
معمولاً از این نوع توابع روی مجموعه ای از سطرهای یک جدول، در جهت اعمال عملیات‌های محاسباتی، ارزیابی داده‌ها، رتبه‌بندی و غیره ... استفاده می‌گردد. به بیان ساده‌تر به وسیله Window Functionها می‌توان، سطرهای یک جدول را گروه‌بندی نمود. و روی گروه‌ها از توابع تجمیعی (Aggregate Functions) استفاده کرد. این نوع توابع از قابلیت و انعطاف‌پذیری زیادی برخوردار هستند و به وسیله آن‌ها می‌توان نتایج (خروجی) بسیار مفیدی را از Queryها بدست آورد، معمولاً از این نوع توابع در Data Mining (داده‌کاوی) و گزارش‌گیری‌ها استفاده می‌گردد. کلمه "Window" در Window Function، به مجموعه سطرهایی اشاره می‌کند، که محاسبات و ارزیابی و... روی آن‌ها اعمال می‌گردد.

از عبارت‌های زیرمی توان برای تعریف محدوده پنجره استفاده کرد که در ادامه بیشتر با آن ها آشنا می‌شویم :

- UNBOUNDED PRECEDINGیا UNBOUNDEDFOLLOWING به معنای تا ابتدا و یا تا انتهای رکوردها است.
- CURRENT ROW به معنای سطر فعلی است.
- <n> ROWS PRECEGING یا <n> ROWS FOLLOWING به معنای n سطر قبل و یا بعد از رکورد فعلی است.

مثال اول : می‌خواهیم براساس فیلد SalesOrderID جدول SalesOrderDetail را Partition بندی نماییم و از توابع جمعی AVG و SUM روی فیلد درآمد(UnitPrice) استفاده کنیم.

```
select SalesOrderID,ProductID,UnitPrice,
 avg(UnitPrice) OVER (PARTITION by SalesOrderID) as AverageUnitPrice,
 sum(UnitPrice) OVER (PARTITION by SalesOrderID) as TotalUnitPrice
from Sales.SalesOrderDetail
order by SalesOrderID
```

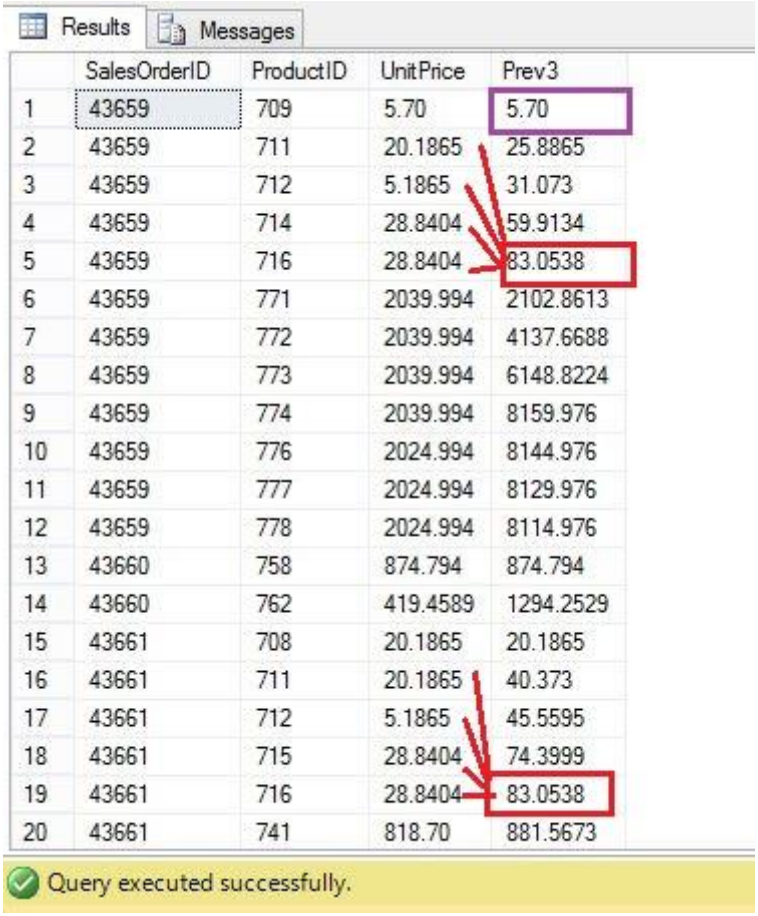


	Salesorderid	ProductID	UnitPrice	AverageUnitPrice	TotalUnitPrice
1...	46985	861	22.794	123.9421	1859.1327
1...	46985	862	22.794	123.9421	1859.1327
1...	46985	823	52.647	123.9421	1859.1327
1...	46985	852	44.994	123.9421	1859.1327
1...	46985	779	1242.8...	123.9421	1859.1327
1...	46985	712	5.1865	123.9421	1859.1327
1...	46986	725	202.332	269.3848	2155.0789
1...	46986	835	324.4527	269.3848	2155.0789
1...	46986	826	67.539	269.3848	2155.0789
1...	46986	770	469.794	269.3848	2155.0789
1...	46986	738	183.9382	269.3848	2155.0789
1...	46986	729	202.332	269.3848	2155.0789
1...	46986	762	234.897	269.3848	2155.0789
1...	46986	760	469.794	269.3848	2155.0789
1...	46987	782	1229.4...	293.3471	16427.4376
1...	46987	779	1242.8...	293.3471	16427.4376
1...	46987	802	88.932	293.3471	16427.4376
1...	46987	711	20.1865	293.3471	16427.4376
1...	46987	780	1242.8...	293.3471	16427.4376
1...	46987	857	53.994	293.3471	16427.4376

مطابق شکل، جدول براساس فیلد SalesOrderID به گروه‌های مختلف تشکیل شده است، و عملیات میانگین و جمع روی فیلد UnitPrice انجام شده است و عملیات Sort روی هر گروه بطور مستقل انجام گرفته است. چنین کاری را نمی‌توانستیم به‌وسیله Group By انجام دهیم.

مثال دوم : نحوه استفاده از ROWS PRECEDING، در این مثال قصد داریم عملیات جمع را روی فیلد UnitPrice انجام دهیم. بطوریکه جمع هر مقدار برابر است با سه مقدار قبلی + مقدار جاری:

```
select SalesOrderID, ProductID, UnitPrice,
sum(UnitPrice) OVER (PARTITION by SalesOrderID ORDER BY ProductID
            ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as Prev3
From Sales.SalesOrderDetail
order by SalesOrderID, ProductID;
```



Results | Messages

	SalesOrderID	ProductID	UnitPrice	Prev3
1	43659	709	5.70	5.70
2	43659	711	20.1865	25.8865
3	43659	712	5.1865	31.073
4	43659	714	28.8404	59.9134
5	43659	716	28.8404	83.0538
6	43659	771	2039.994	2102.8613
7	43659	772	2039.994	4137.6688
8	43659	773	2039.994	6148.8224
9	43659	774	2039.994	8159.976
10	43659	776	2024.994	8144.976
11	43659	777	2024.994	8129.976
12	43659	778	2024.994	8114.976
13	43660	758	874.794	874.794
14	43660	762	419.4589	1294.2529
15	43661	708	20.1865	20.1865
16	43661	711	20.1865	40.373
17	43661	712	5.1865	45.5595
18	43661	715	28.8404	74.3999
19	43661	716	28.8404	83.0538
20	43661	741	818.70	881.5673

Query executed successfully.

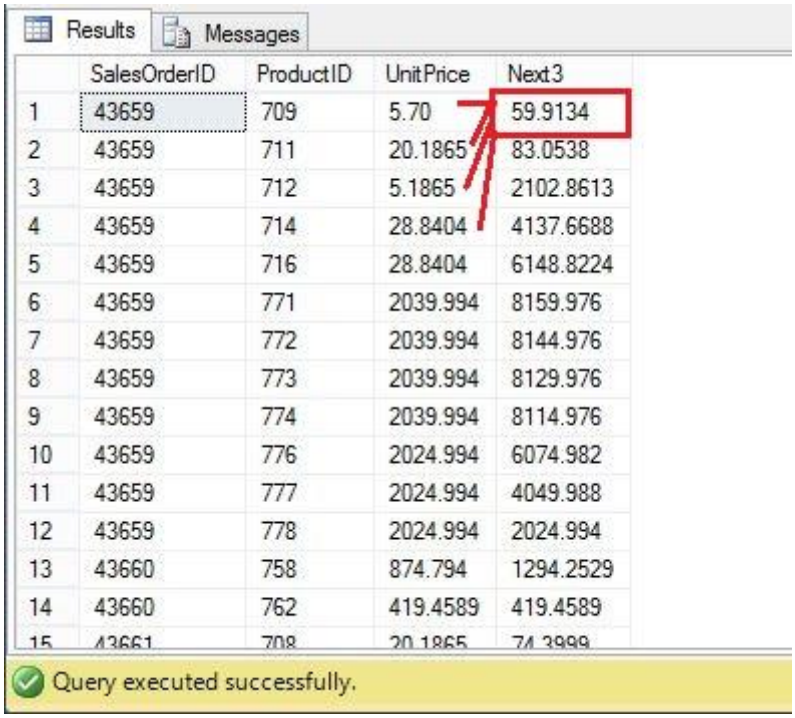
در Script بالا، جدول را براساس فیلد SalesOrderID گروه‌بندی می‌کنیم. هر گروه را بطور مستقل، روی فیلد ProductID بصورت صعودی مرتب می‌نماییم. حال برای آنکه بتوانیم سیاست جمع، روی فیلد UnitPrice ، را پیاده‌سازی نماییم ، قطعه کد زیر را در Script بالا اضافه کردیم.

```
ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as Prev3
```

مقدار جمع فیلد UnitPrice سطر اول، که قبل از آن سطری وجود ندارد، برابر است با مقدار خود، یعنی 5.70، مقدار جمع فیلد UnitPrice سطر دوم برابر است با حاصل جمع مقدار UnitPrice سطر اول و دوم ، یعنی 25.8865 . این روند تا سطر چهار ادامه دارد، اما برای بدست آوردن مقدار جمع فیلد UnitPrice سطر پنجم، مقدار جمع فیلد UnitPrice سطر دوم، سوم، چهارم و پنجم در نظر گرفته می‌شود، و مقدار فیلد UnitPrice سطر اول در حاصل جمع در نظر گرفته نمی‌شود، بنابراین مقدار جمع فیلد UnitPrice سطر پنجم برابر است با 83.0538. در صورت مسئله گفته بودیم، مقدار جمع فیلد UnitPrice هر سطر جاری برابر است با حاصل جمع مقدار سطر جاری و مقادیر سه سطر ماقبل خود.

مثال سوم: نحوه استفاده از ROWS FOLLOWING، این مثال عکس مثال دوم است، یعنی حاصل جمع مقدار فیلد UnitPrice هر سطر برابر است با حاصل جمع سطر جاری با سه سطر بعد از خود. بنابراین Script زیر را اجرا نمایید:

```
select SalesOrderID,ProductID, UnitPrice,
 sum(UnitPrice) OVER (PARTITION by SalesOrderID ORDER BY ProductID
              ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING) as Next3
From Sales.SalesOrderDetail
order by SalesOrderID, ProductID;
```



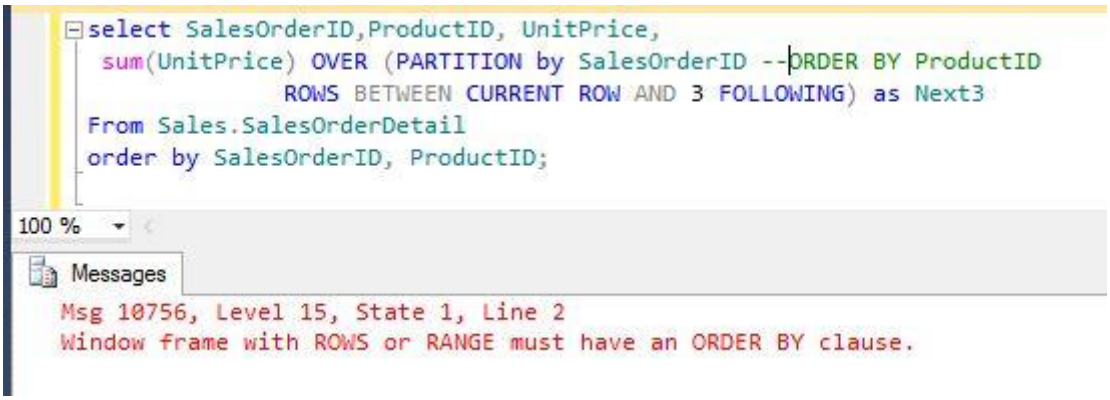
Results | Messages

	SalesOrderID	ProductID	UnitPrice	Next3
1	43659	709	5.70	59.9134
2	43659	711	20.1865	83.0538
3	43659	712	5.1865	2102.8613
4	43659	714	28.8404	4137.6688
5	43659	716	28.8404	6148.8224
6	43659	771	2039.994	8159.976
7	43659	772	2039.994	8144.976
8	43659	773	2039.994	8129.976
9	43659	774	2039.994	8114.976
10	43659	776	2024.994	6074.982
11	43659	777	2024.994	4049.988
12	43659	778	2024.994	2024.994
13	43660	758	874.794	1294.2529
14	43660	762	419.4589	419.4589
15	43661	708	20.1865	74.3999

Query executed successfully.

مطابق شکل مقدار جمع فیلد اول برابراست با حاصل جمع مقدار سطر جاری و سه سطر بعد از آن.

نکته‌ای که در مثال‌های دوم و سوم می‌بایست به آن ها توجه نمود این است که در زمان استفاده از Row یا Range استفاده از Order by در Partition الزامی است. در غیر این صورت با خطا مواجه می‌شوید.



```
select SalesOrderID,ProductID, UnitPrice,
 sum(UnitPrice) OVER (PARTITION by SalesOrderID --ORDER BY ProductID
              ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING) as Next3
From Sales.SalesOrderDetail
order by SalesOrderID, ProductID;
```

100 %

Messages

```
Msg 10756, Level 15, State 1, Line 2
Window frame with ROWS or RANGE must have an ORDER BY clause.
```

Grouping Sets ۴,۶.

از نسخه SQL SERVER 2008 ابزار جدیدی به نام GROUPING SETS افزوده شد. برای استفاده از این امکان مجموعه‌هایی را که می‌خواهیم بر اساس آن‌ها توابع AGGREGATE عمل کنند را به عنوان مؤلفه‌های GROUPING SETS در جلوی عبارت GROUP BY می‌نویسیم. با استفاده از این امکان در واقع در مجموعه جواب، یک ردیف به ازای هر مقدار از هر یک از مؤلفه‌های GROUPING SETS، ایجاد می‌شود. در واقع می‌توان این‌گونه تصور کرد که تابع AGGREGATE موردنظر برای هر یک از مؤلفه‌های GROUPING SETS جداگانه محاسبه می‌شود و اجتماع نتایج در خروجی نمایش داده می‌شود. برای درک بهتر به مثال زیر توجه کنید. خروجی هر دو پرس و جوی زیر یکسان است:

پرسو جوی ۱ (بدون استفاده از GROUPING SETS)

```
USE AdventureWorks2012
GO
SELECTNULLAS Color, Class,sum(ProductID),avg(ListPrice)FROM Production.Product
GROUPBY Class
UNION
SELECT Color,NULL,sum(ProductID),avg(ListPrice)FROM Production.Product
GROUPBY COLOR
```

پرس و جوی ۲ (با استفاده از GROUPING SETS)

```
USE AdventureWorks2012
GO
SELECT Color, Class,sum(ProductID)ASSUM,avg(ListPrice)AS AVERAGE FROM
Production.Product
GROUPBYGROUPINGSETS(Color, Class)
ORDERBY color
```

و نتیجه برای هر دو پرس و جو عبارت است از:

	Color	Class	SUM	AVERAGE
1	NULL	NULL	127933	16.8641
2	NULL	NULL	135472	16.314
3	NULL	H	67454	1679.4964
4	NULL	L	80705	370.6887
5	NULL	M	55581	635.5816
6	Black	NULL	75460	725.121
7	Blue	NULL	23931	923.6792
8	Grey	NULL	842	125.00
9	Multi	NULL	6138	59.865
10	Red	NULL	28304	1401.95
11	Silver	NULL	34613	850.3053
12	Silver/Black	NULL	6566	64.0185
13	White	NULL	3168	9.245
14	Yellow	NULL	32257	959.0913

پرسش؟ چرا در این مجموعه جواب، دو رکورد مجزا با مقدار NULL برای هر دو فیلد Class, Color وجود دارد؟

۴,۷ .ROLLUP

ROLLUP از جمله دستوراتی است که به کمک آن می‌توان پرس‌وجوهایی را طراحی و اجرا کرد که ماهیت گزارشی دارند و می‌توانند به عنوان گزارش‌های مدیریتی در نظر گرفته شوند که قرار است مبدأ تصمیم‌سازی‌های مدیریتی باشند. برای استفاده از آن، عبارت ROLLUP را بعد از عبارت GROUP BY می‌آوریم و فیلدهایی که می‌خواهیم بر اساس آن ها گزارش تهیه شود را به عنوان مؤلفه‌های ROLLUP در نظر می‌گیریم. در این صورت ROLLUP موجب اضافه شدن یک سطر بصورت مجموع مقادیر محاسبه شده برای تابع AGGREGATE براساس مؤلفه تعریف شده در ROLLUP ، خواهد شد.

به مثال زیر توجه کنید:

```
USE AdventureWorks2012
GO
SELECT
CASE GROUPING(Class)
WHEN 0 THEN Class
WHEN 1 THEN'All Classes'
END AS Class,
SUM(listprice)AS Price
FROM Production.Product
GROUP BY ROLLUP(Class, Style)
```

نتیجه:

	Class	Price
1	NULL	2408.02
2	NULL	509.93
3	NULL	839.83
4	NULL	434.94
5	NULL	4192.72
6	H	2914.86
7	H	134803.85
8	H	137718.71
9	L	1502.53
10	L	34454.28
11	L	35956.81
12	M	2466.17
13	M	20838.50
14	M	19914.88
15	M	43219.55
16	All Classes	221087.79

توجه! عبارت GROUPING(Class) به منظور بررسی گروه‌بندی شدن یا نشدن هر رکورد استفاده شده است. درصورتی‌که نتیجه یک رکورد حاصل گروه‌بندی بر اساس فیلد Class باشد مقدار یک و در غیر این صورت مقدار صفر بر می گرداند.

با افزودن GROUPING(Style) به پرس و جوی فوق، نتیجه‌ای دقیق‌تر بدست می‌آید:

```
USE AdventureWorks2012
GO
SELECT
CASE GROUPING(Class)
WHEN 0 THEN Class
WHEN 1 THEN'All Classes'
END AS Class,
CASE GROUPING(Style)
WHEN 0 THEN Style
WHEN 1 THEN'All Styles'
END AS Style,
SUM(listprice)AS Price
FROM Production.Product
GROUP BY ROLLUP(Class, Style)
```

نتیجه:

	Class	Style	Price
1	NULL	NULL	2408.02
2	NULL	M	509.93
3	NULL	U	839.83
4	NULL	W	434.94
5	NULL	All Styles	4192.72
6	H	NULL	2914.86
7	H	U	134803.85
8	H	All Styles	137718.71
9	L	NULL	1502.53
10	L	U	34454.28
11	L	All Styles	35956.81
12	M	NULL	2466.17
13	M	U	20838.50
14	M	W	19914.88
15	M	All Styles	43219.55
16	All Classes	All Styles	221087.79

توجه! برای اینکه با این دستور بهتر آشنا شوید، مثالی مشابه با جدول دیگری از AdventureWorks2012 بسازید و پرس و جوی مربوط به آن را بنویسید.

۴,۸. CUBE

دستور دیگری که برای تهیه‌ی گزارش از آن استفاده می‌شود CUBE است. نحوه‌ی استفاده از آن مشابه ROLLUP است، اما دستور CUBE در واقع پیشرفته‌تر از ROLLUP است و گزارش‌های دقیق‌تری را برای ما تولید می‌کند. این دستور در واقع به ازای هر گروه‌بندی ممکن از داده‌ها، یک مقدار متناسب با تابع AGGREGATE استفاده شده در پرس و جو را به مجموعه نتایج اضافه می‌کند. به مثال زیر توجه کنید:

```
USE AdventureWorks2012
GO
SELECT
CASE GROUPING(Class)
WHEN 0 THEN Class
WHEN 1 THEN'All Classes'
END AS Class,
CASE GROUPING(Style)
WHEN 0 THEN Style
WHEN 1 THEN'All Styles'
END AS Style,
SUM(listprice)AS Price
FROM Production.Product
GROUP BY CUBE(Class, Style)
```

نتیجه:

	Class	Style	Price
1	NULL	NULL	2408.02
2	H	NULL	2914.86
3	L	NULL	1502.53
4	M	NULL	2466.17
5	All Classes	NULL	9291.58
6	NULL	M	509.93
7	All Classes	M	509.93
8	NULL	U	839.83
9	H	U	134803.85
10	L	U	34454.28
11	M	U	20838.50
12	All Classes	U	190936.46
13	NULL	W	434.94
14	M	W	19914.88
15	All Classes	W	20349.82
16	All Classes	All Styles	221087.79
17	NULL	All Styles	4192.72
18	H	All Styles	137718.71
19	L	All Styles	35956.81
20	M	All Styles	43219.55

همانطور که مشاهده می‌کنید، سطرهای ۵ و ۷ و ۱۲ و ۱۵ که در این مجموعه جواب وجود دارند، در مجموعه جواب مثال ROLLUP وجود نداشت و این تفاوت CUBE با ROLLUP است.

توجه! برای اینکه با این دستور بهتر آشنا شوید، مثالی مشابه با جدول دیگری از AdventureWorks2012 بسازید و پرس و جوی مربوط به آن را بنویسید.

۴,۹. تمرین

۱- کد زیر را در محیط MS SQL SERVER وارد نمایید و آن را اجرا کنید. نتیجه چیست؟

بطور دقیق توضیح دهید که ستون آخر از جدولِ نتایج، بیانگر چه چیزی است؟

```
USE AdventureWorks2012
GO
SELECT Sales.SalesOrderHeader.OrderDate, Sales.SalesOrderDetail.LineTotal,
AVG(Sales.SalesOrderDetail.LineTotal)OVER (PARTITION BY
Sales.SalesOrderHeader.CustomerID
ORDER BY Sales.SalesOrderHeader.OrderDate, Sales.SalesOrderHeader.SalesOrderID
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
FROM  Sales.SalesOrderHeader JOIN  Sales.SalesOrderDetail ON
(SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID)
```

۲- پرس و جویی بنویسید که تعداد سفارش‌ها و مجموع ارزش سفارش‌ها را برای هر Territory همراه با منطقه آن Territory نمایش دهد ضمن آنکه مجموع تعداد و ارزش سفارش‌ها را برای تمام Territoryها در یک منطقه و در پایان گزارش برای همه‌ی مناطق نشان دهد. خروجی باید دقیقاً مطابق شکل ۴-۱ باشد.

	TerritoryName	Region	SalesTotal	SalesCount
1	France	Europe	7251555.6473	2672
2	Germany	Europe	4915407.596	2623
3	United Kingdom	Europe	7670721.0356	3219
4	All Teritories	Europe	19837684.2789	8514
5	Canada	North America	16355770.4553	4067
6	Central	North America	7909009.0062	385
7	Northeast	North America	6939374.4813	352
8	Northwest	North America	16084942.5482	4594
9	Southeast	North America	7879655.0731	486
10	Southwest	North America	24184609.6011	6224
11	All Teritories	North America	79353361.1652	16108
12	Australia	Pacific	10655335.9598	6843
13	All Teritories	Pacific	10655335.9598	6843
14	All Teritories	All Regions	109846381.4039	31465

شکل ۴-۱

توضیح دهید آیا ترتیب نوشتن فیلدهای مؤلفه‌ی rollup در خروجی تأثیر دارد؟

۳- پرس و جویی بنویسید که تعداد سفارش و مجموع ارزش سفارش را برای هر زیر گروه کالا همراه با گروه آن نمایش دهد همچنین در پایان این گزارش رکورد هایی حاوی تعداد و ارزش برای همه ی زیر گروه ها در یک گروه مشخص نشان داده شده باشد. (خروجی دقیقا مشابه شکل ۴ -۲ است)

subCat	cat	salesCount	salesTotal
Bike Racks	Accessories	3166	18193146/3309
Bike Stands	Accessories	249	165804/2285
Bottles and Cages	Accessories	10552	26280952/7128
Cleaners	Accessories	3319	18413114/3457
Fenders	Accessories	2121	2126159/3501
Helmets	Accessories	19541	134483767/0441
Hydration Packs	Accessories	2761	16860360/8546
Locks	Accessories	1087	15035605/0924
Pumps	Accessories	1130	15666409/1595
Tires and Tubes	Accessories	18006	16861223/611
All sub	Accessories	61932	264086542/7296
Mountain Bikes	Bikes	28321	357041832/9383
Road Bikes	Bikes	47196	624874728/6655
Touring Bikes	Bikes	14751	207928846/7541
All sub	Bikes	90268	1189845408/3579
All sub	All cat	152200	1453931951/0875

شکل۴-۲

۴,۱۰. منابع

- Results
  Set Operators:
  http://technet.microsoft.com/en-us/library/aa213247(v=sql.80).aspx
  and
  SQL Tutorial pdf from tutorialspoint.com

- Calse
  http://technet.microsoft.com/en-us/library/aa258235(v=sql.80).aspx

- SQL Server Aggregate Functions and Windowing:
  Database System Concepts, *Sixth Edition*, Avi Silberschatz, Henry F. Korth, S. Sudarshan
  Beginning T-SQL With Microsoft SQL Server 2005 And 2008

  http://www.mssqltips.com/sqlservertip/1221/sql-server-tsql-aggregate-functions/

- Grouping Sets , Rollup, Cube:
  R3-Wiley.Microsoft.SQL.Server.2008.Bible.Aug.2009

۵. فصل پنجم : SQL پیشرفته: PIVOT ، توابع

۵,۱ PIVOT

به کمک این دستور می‌توان جای سطر و ستون را در گزارش عوض کرد به عبارت بهتر بجای آنکه چندین سطر داشته باشیم، جهت راحت شدن تجزیه و تحلیل گزارش، تعدادی ستون به گزارش اضافه می‌کنیم.

ابتدا به مثال ساده‌ی زیر توجه فرمایید:

یک پرس و جوی ساده برای محصولات فروخته شده در یک فروشگاه نتایجی شبیه آنچه در جدول ۵-۲ آمده، خواهد داشت:

SalesPerson	Product	SalesAmount
Bob	Pickles	$100.00
Sue	Oranges	$50.00
Bob	Pickles	$25.00
Bob	Oranges	$300.00
Sue	Oranges	$500.00

جدول ۵-۱

واضح است که این نتیجه جهت تجزیه‌وتحلیل داده‌ها و تصمیم گیری مدیریتی مناسب نیست (بالأخص وقتی وارد محیطی بزرگ‌تر از یک فروشگاه کوچک می‌شویم و با حجم وسیعی از کالاهای فروخته شده سروکار داریم) و نیاز داریم اطلاعات بصورت مناسبی گردآوری شوند برای این منظور از PIVOT استفاده می‌کنیم. نحوه استفاده از دستور PIVOT بصورت زیر است:

بخش اول

```
SELECT <non-pivoted column>,
       [first pivoted column] AS <column name>,
       [second pivoted column] AS <column name>,
       ...
       [last pivoted column] AS <column name>
```

بخش دوم

```
FROM
    (<SELECT query that produces the data>)
    AS <alias for the source query>
```

بخش سوم

```
PIVOT
(
    <aggregation function>(<column being aggregated>)
FOR
[<column that contains the values that will become column headers>]
    IN ( [first pivoted column], [second pivoted column],
    ... [last pivoted column])
) AS <alias for the pivot table>
```

۱. در نخستین SELECT فیلدهایی که قرار است در گزارش نهایی وجود داشته باشند را به اضافه‌ی مقادیری که قرار است از سطر به ستون تبدیل شوند را می‌آوریم. (مثلاً برای مثال قبل می‌خواهیم نام فروشنده و در جلوی آن مجموع فروش هر کالا در ستونی متفاوت وجود داشته باشد.)

۲. در قسمت دوم با نوشتن یک دستور SELECT دیگر کلیه‌ی اطلاعات مورد نیاز را از جدول موردنظر استخراج می‌کنیم.

۳. در قسمت سوم ابتدا تابع محاسباتی مدنظرمان را برای فیلدی که در SELECT دوم آمده، می‌نویسیم. (مثلاً در این مثال SUM(SalesAmount) را می‌نویسیم)

سپس در بخش FOR نام فیلدی که قرار است مقادیر مختلف آن به عنوان ستون در نتیجه حاضر شوند را می‌آوریم و در قسمت IN نیز، مقادیر مدنظر از این فیلد را که در قسمت اول (به عنوان شمای خروجی) مشخص کرده بودیم، می‌آوریم.

پس برای مثال فوق به چنین پرس و جویی می‌رسیم:

```
SELECT SalesPerson, [Oranges] AS Oranges, [Pickles] AS Pickles
FROM
(SELECT SalesPerson, Product, SalesAmount
FROM ProductSales ) ps
PIVOT
(
SUM (SalesAmount)
FOR Product IN
( [Oranges], [Pickles])
) AS pvt
```

بخش اول

بخش دوم

بخش سوم

نتیجه مطابق آنچه در ج

SalesPerson	Oranges	Pickles
Bob	$300.00	$125.00
Sue	$550.00	

جدول ۵-۲

برای روشن‌تر شدن مفهوم PIVOT به مثال کاربردی‌تر زیر توجه کنید:

مثلاً پرس و جوی زیر شناسه و STYLE و کلاس کالاهایی را در خروجی نشان خواهد داد که کلاس تعریف شده دارند.

```
USE AdventureWorks2012
GO
SELECT Style, Class, ProductID
FROM Production.Product
WHERE Class <>'NULL'
```

با اجرای این پرس و جو نتیجه شامل ۲۴۷ ردیف اطلاعات بصورت زیر است:

مسلماً نمی‌توان از نتیجه فوق به عنوان یک گزارش استفاده کرد. پس با استفاده از دستور PIVOT تعداد کالاهای هر کلاس را در ستون‌های مختلف برای هر STYLE می‌آوریم:

```
USE AdventureWorks2012
GO
SELECT Style, U, M, L
FROM
(SELECT Style, ProductID, Class
FROM Production.Product)AS SourceTable
PIVOT
(
count(ProductID)
FOR Class IN(U, M, L)
)as PVT
```

بخش اول

بخش دوم

بخش سوم

نتیجه:

	Style	U	M	L
1	NULL	0	24	29
2	M	0	0	0
3	U	0	22	68
4	W	0	22	0

توجه! برای اینکه با این دستور بهتر آشنا شوید، مثالی مشابه با جدول دیگری از AdventureWorks2012 بسازید و پرس و جوی مربوط به آن را بنویسید.

	Style	Class	ProductID
1	NULL	L	317
2	NULL	M	318
3	NULL	L	356
4	NULL	M	357
5	NULL	L	400
6	NULL	L	490
7	NULL	L	507
8	NULL	M	508
9	NULL	L	510
10	NULL	M	511
11	NULL	L	514

۵,۲. توابع در SQL Server

در این بخش در خصوص توابع و شیوه به‌کارگیری آن‌ها در SQL بحث خواهد شد.

۵,۲,۱. مقدمه

توابع مؤلفه‌هایی هستند که بر اساس ورودی نتایجی را به کاربر برمی‌گردانند. SQL Server از توابع از پیش تعریف شده زیادی پشتیبانی می کند این توابع در چهار دسته زیر قرار می‌گیرند.

نوع تابع	توضیحات
Rowset Function	شیء ای را بر می گرداند که می توان از آن مانند table reference استفاده کرد
Aggregate Functions	بر روی دسته‌ای از مقادیر عمل می کند ولی تنها یک مقدار خروجی دارد، به منظور جمع‌بندی مقادیر از این توابع استفاده می شود.
Ranking Functions	برای هرکدام از سطرهای موجود در یک محدوده مقداری را به عنوان امتیازبندی آن سطر ارائه می کند.
Scalar Functions	بر روی یک مقدار داده عمل می کند و تنها یک مقدار خروجی دارد.

جدول ۵۵-۳

علاوه بر این به منظور امکان پاسخ‌گویی به سایر نیازهای کاربران، پایگاه داده Microsoft SQL Server امکان تعریف توابع جدیدی تحت عنوان User Defined Functions را نیز به کاربران خود داده است.

در بین انواع توابع معرفی‌شده Aggregate Functions و Scalar Functions جز توابع پرکاربرد SQL محسوب می شوند.

در ادامه به معرفی برخی از توابع از پیش تعریف شده این دو گروه پرداخته می شود و در نهایت نحوه ایجاد یک User Defined Function بررسی خواهد شد

۵,۲,۲. توابع داخلی SQL Server [۸]

در این بخش به بررسی برخی از توابع پرکاربرد از پیش تعریف شده در SQL Server می‌پردازیم.

[۸] SQL Server Built in Functions

توابع تجمعی[9]

این نوع از توابع بر روی مجموعه ای از داده ها عمل می‌کنند و تنها یک مقدار را با محاسبه مقادیر یک ستون بر می گرداند. که با آن‌ها در فصل سوم آشنا شده‌اید.

توابع مقیاسی[10]

این توابع برای هرکدام از رکوردها مقدار منحصربه‌فردی را بر می گرداند.

```
SELECT UPPER(s.ContactName)
FROM Customers s
```

در این مثال نام مشتریان به صورت حروف بزرگ نمایش داده می شود. تابع UPPER() بر روی تمام رکوردها عمل کرده و برای هرکدام از آن ها یک مقدار خروجی منحصربه‌فرد ایجاد کرده است.

```
SELECT SUBSTRING(ContactName,1, CHARINDEX(' ',ContactName) - 1)
FROM Customers
```

درصورتی‌که فرض کنیم نام کوچک و بزرگ مشتریان با یک کاراکتر فاصله از یکدیگر جدا شده است، مثال بالا نام کوچک تمام مشتریان را استخراج کرده و نمایش می دهد.

تابع SUBSTRING( ) قسمتی از یک متغیر رشته‌ای را جدا کرده و بر می گرداند و تابع CHARINDEX( ) مکان کاراکتر مشخص‌شده‌ای که در این مثال یک فاصله می‌باشد را در رشته ورودی به دست می‌آورد و به عنوان خروجی بر می گرداند.

```
insert into Orders (CustomerID, OrderDate) VALUES (null,CONVERT(DATE,GETDATE()))
```

به منظور انعطاف‌پذیری بیشتر برنامه می توان از تابع Convert به منظور تبدیل انواع داده به یکدیگر استفاده کرد، در مثال بالا تاریخ و زمان فعلی کامپیوتر استخراج شده و به متغیری از نوع تاریخ تبدیل می شود و در جدول Orders رکورد جدیدی بر مبنای آن ایجاد خواهد شد.

برای بررسی سایر توابع اسکالر از پیش تعریف شده به راهنمای Microsoft sql server و یا به سایت زیر مراجعه فرمایید.

http://msdn.microsoft.com/en-us/library/ms711813(v=vs.85).aspx

در شکل زیر توابع پر استفاده روی رشته ها را در SQL Server مشاهده میکنم:

Function	Example	Result	Description

[9] SQL Server Aggregate Functions

[10] SQL Server Scalar Functions

CONCAT	CONCAT('WWW','.com')	WWW.com	Adds two or more strings together
LEN	LEN('com')	3	Returns the length of a string
LOWER	LOWER('SQL!')	sql!	Converts a string to lower-case
LTRIM	LTRIM(' SQL !')	SQL !	Removes leading spaces from a string
PATINDEX	PATINDEX('%sc%','W3Sc.com')	3	Returns the position of a pattern in a string
REPLACE	REPLACE('TSQL','T','M')	MSQL	Replaces all occurrences of a substring within a string, with a new substring
RTRIM	RTRIM('SQL ')	SQL	Removes trailing spaces from a string
SUBSTRING	SUBSTRING('SQL',1,2)	SQ	Extracts some characters from a string
TRIM	TRIM(' SQL ')	SQL	Removes leading and trailing spaces (or other specified characters) from a string
UPPER	UPPER('sQl')	SQL	Converts a string to upper-case
REPLICATE	REPLICATE('SQ', 2)	SQSQ	Repeats a string a specified number of times
RIGHT	RIGHT('SQL',2)	QL	Extracts a number of characters from a string (starting from right)
LEFT	LEFT('SQL', 2)	SQ	Extracts a number of characters from a string (starting from left)

توابع رتبه‌بندی[11]

این توابع رتبه‌ای را برای هرکدام از سطرهای یک پارتیشن، بر می گرداند. بر اساس نوع تابع استفاده شده، برخی از سطرها ممکن است، مقدار یکسانی را دریافت کنند. توابع رتبه‌بندی غیرقطعی‌اند.

مثال زیر نحوه استفاده از این توابع را نشان می دهد، در صورت نیاز می توانید در رابطه با نحوه عملکرد هر تابع، به توضیحات آن تابع مراجعه کنید.

```
USE AdventureWorks2012;
GO
SELECT p.FirstName, p.LastName
    ,ROW_NUMBER() OVER (ORDER BY a.PostalCode) AS "Row Number"
    ,RANK() OVER (ORDER BY a.PostalCode) AS Rank
    ,DENSE_RANK() OVER (ORDER BY a.PostalCode) AS "Dense Rank"
    ,NTILE(4) OVER (ORDER BY a.PostalCode) AS Quartile
    ,s.SalesYTD
    ,a.PostalCode
FROM Sales.SalesPerson AS s
    INNER JOIN Person.Person AS p
        ON s.BusinessEntityID = p.BusinessEntityID
    INNER JOIN Person.Address AS a
        ON a.AddressID = p.BusinessEntityID
WHERE TerritoryID IS NOT NULL AND SalesYTD <> 0;
```

نتیجه:

	FirstName	LastName	Row Number	Rank	Dense Rank	Quartile	SalesYTD	PostalCode
1	Michael	Blythe	1	1	1	1	3763178.1787	98027
2	Linda	Mitchell	2	1	1	1	4251368.5497	98027
3	Jillian	Carson	3	1	1	1	3189418.3662	98027
4	Garrett	Vargas	4	1	1	1	1453719.4653	98027
5	Tsvi	Reiter	5	1	1	2	2315185.611	98027
6	Pamela	Ansman-Wolfe	6	1	1	2	1352577.1325	98027
7	Shu	Ito	7	7	2	2	2458535.6169	98055
8	José	Saraiva	8	7	2	2	2604540.7172	98055
9	David	Campbell	9	7	2	3	1573012.9383	98055
10	Tete	Mensa-Annan	10	7	2	3	1576562.1966	98055
11	Lynn	Tsoflias	11	7	2	3	1421810.9242	98055
12	Rachel	Valdez	12	7	2	4	1827066.7118	98055
13	Jae	Pak	13	7	2	4	4116871.2277	98055
14	Ranjit	Varkey Chud...	14	7	2	4	3121616.3202	98055

سایر توابع از پیش تعریف شده

به منظور دسترسی به سایر توابع از پیش تعریف شده در SQL Server بر اساس دسته بندی کاربردی به سایت زیر مراجعه فرمایید.

http://technet.microsoft.com/en-us/library/aa258899(v=sql.80).aspx

[11] Ranking Functions

۵،۲،۳. توابع تعریف شده توسط کاربر

توابع تعریف شده توسط کاربر به دو دسته اصلی تقسیم می‌شوند، توابع scalar value-returning که یک مقدار را برمی‌گردانند و توابع table value-returning که یک جدول برمی‌گردانند. البته همانطور که در جدول ۴-۲ مشاهده می‌شود، توابع table value-returning خود به دو دسته inline و multistatement تقسیم می‌شوند.

نوع	ویژگی‌ها
Scalar	شبیه توابع در زبان‌های برنامه‌نویسی دیگر هستند و فقط یک مقدار برمی‌گردانند.
Inline table	یک جدول برمی‌گرداند که توسط یک دستور **select** مشخص می‌شود.
Multistatement table	یک جدول برمی‌گرداند که ساختار دقیق آن در بدنه‌ی تابع مشخص می‌شود.(نوع داده‌ها و ستون‌ها در دستور **Returns** مشخص می‌شوند)

جدول ۵۵-۴

در این بخش به نحوه ایجاد توابع دلخواه، با عملکرد موردنظر خود، می‌پردازیم.

توابع مقیاسی

User defined scalar functionها، یک مقدار خروجی از نوع داده‌ای تعیین‌شده پس از عبارت Return را برمی‌گردانند. برای توابع inline نیاز به تعریف بدنه تابع نیست، مقدار اسکالر حاصل یک Statement تنها است. برای توابع اسکالر چند عبارتی (MultiStatement)، بدنه تابع(Function Body) با استفاده از بلاک Begin…END تعریف خواهد شد. این بلاک‌ها شامل مجموعه ای از Transact-SLQ ها می‌باشد که تنها یک مقدار خروجی دارد. نوع مقدار خروجی می‌تواند هر نوعی به جز text, ntext, image, cursor و timestamp باشد.

مثال زیر یک تابع اسکالر چند عبارتی (Multi Statement) ایجاد می کند. تابع ProductID را به عنوان ورودی می‌پذیرد و مقدار تجمیعی (Aggregated)، کالای مشخص شده در انبار را استخراج می کند.

```
IF OBJECT_ID (N'dbo.ufnGetInventoryStock', N'FN') IS NOT NULL
    DROP FUNCTION ufnGetInventoryStock;
GO
CREATE FUNCTION dbo.ufnGetInventoryStock(@ProductID int)
RETURNS int
AS
-- Returns the stock level for the product.
BEGIN
    DECLARE @ret int;
    SELECT @ret = SUM(p.Quantity)
    FROM Production.ProductInventory p
    WHERE p.ProductID = @ProductID AND p.LocationID = '6';
     IF (@ret IS NULL)
        SET @ret = 0;
```

```
    RETURN @ret;
END;
GO
```

در پرس و جوی زیر با استفاده از تابع تعریف شده در قسمت قبل، موجودی انبار برای محصولاتی که ProductModelID آن ها بین ۷۵ تا ۸۰ نمایش داده خواهد شد.

```
SELECT ProductModelID, Name, dbo.ufnGetInventoryStock(ProductID)AS CurrentSupply
FROM Production.Product
WHERE ProductModelID BETWEEN 75 and 80;
GO
```

Table-Valued Functions

توابع User defined table-valued، جدولی را به عنوان خروجی برمی‌گردانند. برای توابع یک خطی (inline) نیاز به تعریف بدنه تابع نیست، در این نوع توابع، خروجی حاصل یک عبارت select است.

مثال زیر یک تابع inline table-value ایجاد می کند. این تابع CustomerID (StoreID) را به عنوان ورودی می‌پذیرد و ستون‌های ProductID، name، تجمیع(Aggregate) فروش year-to-date را به عنوان YTD، برای هر محصولی که به فروشگاه فروخته شده است بر می گرداند.

```
IF OBJECT_ID (N'Sales.ufn_SalesByStore', N'IF') IS NOT NULL
    DROP FUNCTION Sales.ufn_SalesByStore;
GO
CREATE FUNCTION Sales.ufn_SalesByStore (@storeid int)
RETURNS TABLE
AS
RETURN
(
    SELECT P.ProductID, P.Name, SUM(SD.LineTotal) AS 'Total'
    FROM Production.Product AS P
    JOIN Sales.SalesOrderDetail AS SD ON SD.ProductID = P.ProductID
    JOIN Sales.SalesOrderHeader AS SH ON SH.SalesOrderID = SD.SalesOrderID
    JOIN Sales.Customer AS C ON SH.CustomerID = C.CustomerID
    WHERE C.StoreID = @storeid
    GROUP BY P.ProductID, P.Name
);
GO
```

با استفاده از پرس و جوی زیر تابع ایجاد شده را برای مقدار ورودی ۶۰۲ نمایش می دهد.

```
SELECT * FROM Sales.ufn_SalesByStore (602);
```

برای Table-valued function های چند عبارتی (Multi statement) بدنه تابع با استفاده از بلاک BEGIN…END تعریف خواهد شد، این بلاک شامل مجموعه ای از عبارت‌های TSQL می‌باشد که سطرهای خاصی را ایجاد کرده و به جدول خروجی اضافه خواهد کرد.

مثال زیر EmployeeID را به عنوان ورودی می‌پذیرد و لیستی از تمام کارمندانی که به کارمند مشخص شده به صورت مستقیم و یا غیرمستقیم گزارش می‌دهند را مشخص می کند.

در قسمت بعدی نیز نمونه از اجرای این تابع آورده شده است.

```
IF OBJECT_ID (N'dbo.ufn_FindReports', N'TF') IS NOT NULL
    DROP FUNCTION dbo.ufn_FindReports;
GO
CREATE FUNCTION dbo.ufn_FindReports (@InEmpID INTEGER)
RETURNS @retFindReports TABLE
(
    EmployeeID int primary key NOT NULL,
    FirstName nvarchar(255) NOT NULL,
    LastName nvarchar(255) NOT NULL,
    JobTitle nvarchar(50) NOT NULL,
    RecursionLevel int NOT NULL
)
--Returns a result set that lists all the employees who report to the
--specific employee directly or indirectly.*/
AS
BEGIN
WITH   EMP_cte(EmployeeID,   OrganizationNode,   FirstName,   LastName,   JobTitle,
RecursionLevel) -- CTE name and columns
    AS (
        SELECT  e.BusinessEntityID,  e.OrganizationNode,  p.FirstName,  p.LastName,
e.JobTitle, 0 -- Get the initial list of Employees for Manager n
        FROM HumanResources.Employee e
                    INNER JOIN Person.Person p
                    ON p.BusinessEntityID = e.BusinessEntityID
        WHERE e.BusinessEntityID = @InEmpID
        UNION ALL
        SELECT  e.BusinessEntityID,  e.OrganizationNode,  p.FirstName,  p.LastName,
e.JobTitle, RecursionLevel + 1 -- Join recursive member to anchor
        FROM HumanResources.Employee e
            INNER JOIN EMP_cte
            ON e.OrganizationNode.GetAncestor(1) = EMP_cte.OrganizationNode
                    INNER JOIN Person.Person p
                    ON p.BusinessEntityID = e.BusinessEntityID
        )
-- copy the required columns to the result of the function
   INSERT @retFindReports
   SELECT EmployeeID, FirstName, LastName, JobTitle, RecursionLevel
   FROM EMP_cte
   RETURN
END;
GO
-- Example invocation
SELECT EmployeeID, FirstName, LastName, JobTitle, RecursionLevel
FROM dbo.ufn_FindReports(1);
```

توجه! هدف مثال‌های ذکر شده، این است که با ساختار کلی نوشتن انواع توابع، آشنا شوید.GO

۵,۳. تمرین

۱- پرس و جویی بنویسید که نام تمامی کالاها را همراه با تعداد فروش هرکدام در هر منطقه بصورت ستونی جداگانه برای هر کالا نشان دهد. بخشی از نتیجه مطلوب در شکل ۵-۱ مشاهده می شود.

	Name	Europe	North America	Pacific
1	All-Purpose Bike Stand	67	117	65
2	AWC Logo Cap	1000	1925	457
3	Bike Wash - Dissolver	294	793	240
4	Cable Lock	30	230	0
5	Chain	55	167	28
6	Classic Vest, L	44	113	44
7	Classic Vest, M	140	356	59
8	Classic Vest, S	152	461	68

۲- پرس و جوی زیر را اجرا کنید.

```
select Person.BusinessEntityID, PersonType, Gender
from Person.Person join HumanResources.Employee on (Person.BusinessEntityID
= Employee.BusinessEntityID)
```

(این پرس و جو PersonType و جنسیت کارمندان در AdventureWorks2012 را نمایش می دهد.
(PersonType برای کارمندان یا "EM" است و یا "SP"))

این پرس و جو را به نحوی تغییر دهید که نتیجه بصورت آنچه در شکل ۴-۴ دیده میشود، تغییر یابد:

	PersonType	M	F
1	EM	196	77
2	SP	10	7

شکل ۵-۲

۳- پرس و جویی بنویسید که نام تمام کالاهایی را برگرداند که طول نام آن ها کمتر از ۱۵ کاراکتر باشد و در نام آن‌ها، دومین کاراکتر از آخر e باشد. (مثل Freewheel)

۴- تابعی بنویسید که که یک کلمه ۱۰ کاراکتری را بعنوان ورودی دریافت کند و در صورتی که این کلمه به فرمت تاریخ شمسی (با فرمت 1395/01/12) با سال ۴ رقمی و ماه و روز ۲ رقمی که با / از هم جدا شده اند ، نیست در خروجی "فرمت تاریخ ناصحیح است" چاپ شود در غیر این صورت خروجی مطابق مثال زیر باشد :

ورودی : 1395/01/12 ----------> خروجی : ۱۲ فروردین ماه ۱۳۹۵ شمسی

۵- تابعی بنویسید که دو عدد سال و ماه و نام کالا را دریافت کند و مناطقی که حداقل یکبار در ماه و سال مورد نظر، کالای مد نظر را فروخته اند برگرداند.

۵,۴. منابع

Pivot:
http://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx
http://www.mssqltips.com/sqlservertip/1019/crosstab-queries-using-pivot-in-sql-ser
Beginning T-SQL With Microsoft SQL Server 2005 And 2008
http://www.mssqltips.com/sqlservertip/1221/sql-server-tsql-aggregate-functions/

Ranking Functions:
http://msdn.microsoft.com/en-us/library/ms189798.aspx

Scalar Functions:
http://msdn.microsoft.com/en-us/library/ms709434(v=vs.85).aspx
User Defined Classification:
http://msdn.microsoft.com/en-us/magazine/cc164062.aspx

Scalar User Defined Functions:
http://technet.microsoft.com/en-us/library/ms177499(v=sql.105).aspx

Table-valued user defined functions:
http://technet.microsoft.com/en-us/library/ms177499(v=sql.105).aspx

http://technet.microsoft.com/en-us/library/aa258901(v=sql.80).aspx
http://msdn.microsoft.com/en-us/library/ms178544.aspx
http://www.w3schools.com/sql/sql_functions.asp