دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

دستور کار آزمایشگاه پایگاه داده‌ها
ترم اول 99-98

استاد درس
دکتر علیرضا بصیری

ویرایش ششم تابستان 1398

تهیه کنندگان:

دانیال اکبری

مهران صادقی

سید میثم غفاری

بازنگری و بازنویسی:

محمد سلیمان‌نژاد

وحید مروج

سمانه سمیعی

تحت نظارت

دکتر علیرضا بصیری**

فهرست مطالب

1.فصل اول : آشنایی با SQL مقدماتی **4**

1.1. مقدمه 4

1.2. دستورات Data Definition Language 4

1.2.1. ساخت جدول 4

1.2.2. قیدها 5

1.2.3. ویرایش ساختار جداول 6

1.3. دستورات Data Manipulation Language 7

1.3.1. وارد کردن اطلاعات به یک جدول 7

1.3.2. دستورSELECT 7

1.3.3. دستور WHERE 8

1.3.4. دستور ORDER BY 8

1.3.5. دستور UPDATE 9

1.4. دستور JOIN و انواع آن 9

2.فصل دوم : آشنایی با Adventure Works2012 و چند دستور مهم در SQL **13**

2.1. معرفی پایگاه داده AdventureWorks 2012 13

2.1.1. روش نصب 13

2.1.2. آشنایی با Adventure Works 2012 13

2.2. عملگرها بر روی مجموعه نتایج (result set operators) 16

2.2.1. اجتماع (UNION) 16

2.2.2. اشتراک (INTERSECT) 17

2.2.3. تفاضل (EXCEPT) 17

2.3. دستور CASE 18

2.3.1. دستور CASE در قالب ساده 18

2.3.2. دستور CASE در قالب جستجویی 19

2.4. توابع تجمیعی در SQL 20

2.5. Group By و Having 21

2.6. تمرین ..... **Error! Bookmark not defined.**

1. فصل اول : آشنایی با SQL مقدماتی

1.1. 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.

1.2. دستورات Data Definition Language

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

1.2.1. 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؟

1.2.2. 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 1-1

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)
);
```

1.2.3. 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)
```

1.3. دستورات Data Manipulation Language

1.3.1. 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')
```

1.3.2. 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

1.3.3. 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

1.3.4. 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:

-1

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

-2

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

1.3.5. 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';
```

1.4. 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 1-2 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 1-2

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.

2. 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.

2.1. 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.

2.1.1. روش نصب

برای نصب Adventure Works 2012 چندین روش وجود دارد که یک روش را در اینجا برسی می کنیم:

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

2.1.2. 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 (1=در حال پردازش، 2= 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 2-1

- 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 2-2

- 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 2-3

- 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

2.2. عملگرها بر روی مجموعه نتایج (result set operators)

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

2.2.1. 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

Table1

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
```

2.2.2. 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
```

2.2.3. 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
```

2.3. 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:

2.3.1. 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
```

2.3.2. 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
```

2.4. 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 2-2

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 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];
```

2.5. 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