آموزش جامع SQL

(به همراه مثال‌هایی بر مبنای سیستم خرید و فروش)

شرکت نرم افزاری جادوی فکر - (مجری پروژه نرم افزاری حوزه هوش مصنوعی و بلاکچین)

www.jadoosoft.com

📘فصل اول: اصول طراحی بانک اطلاعاتی درSQL Server

Chapter 1: Principles of Database Design in Microsoft SQL Server

---

🔷۱.۱ مقدمه‌ای بر پایگاه داده

پایگاه داده رابطه‌ای (Relational Database) مجموعه‌ای از جداول مرتبط با هم است. درSQL Server ، پایگاه داده‌ها در قالب **دیتابیس‌ها** (Databases)ساخته می‌شوند و هر دیتابیس شامل جداول، نماها(Views) ، ایندکس‌ها(Indexes) ، و سایر اشیاء است.

🔷۱.۲ اجزای اصلی پایگاه داده رابطه‌ای

جزء	توضیح
Table(جدول)	ساختار اصلی ذخیره‌سازی داده‌ها
Row(سطر)	یک رکورد از داده‌ها
Column(ستون)	یک ویژگی از موجودیت
Primary Key	شناسه یکتا برای هر سطر
Foreign Key	کلیدی برای ایجاد ارتباط با جدول دیگر

◆۱.۳ انواع روابط بین جداول

نوع رابطه	مثال
یک به یک(1:1)	هر کارمند فقط یک کارت شناسایی دارد
یک به چند (1:N)	هر مشتری می‌تواند چندین فاکتور داشته باشد
چند به چند (M:N)	هر کالا می‌تواند در چند فاکتور باشد و بالعکس (با جدول واسط پیاده‌سازی می‌شود)

◆۱.۴ مراحل طراحی پایگاه داده

1. **شناسایی موجودیت‌ها** :کالا، طرف حساب، فاکتور خرید، فاکتور فروش
2. **شناسایی ویژگی‌ها** :نام کالا، قیمت، موجودی، نوع طرف حساب و...
3. **ایجاد جداول برای موجودیت‌ها**
4. **ایجاد کلیدهای اصلی و خارجی برای اتصال جداول**
5. **توجه به نرمال‌سازی داده‌ها برای کاهش افزونگی**

◆۱.۵ طراحی جداول سیستم خرید و فروش

در SQL Server ، از نوع داده INT IDENTITY برای ایجاد کلیدهای خودافزاینده (auto-increment) استفاده می‌شود.

📂جدول کالاها(Products)

```
CREATE TABLE Products (
    ProductID INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Price DECIMAL(18, 2) NOT NULL,
    Stock INT DEFAULT 0
);
```

📂جدول طرف حساب‌ها(Accounts)

```
CREATE TABLE Accounts (
    AccountID INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Type NVARCHAR(10) CHECK (Type IN ('Customer', 'Supplier')) NOT NULL,
    Phone NVARCHAR(20)
);
```

📂جدول فاکتورهای فروش(SalesInvoices)

```
CREATE TABLE SalesInvoices (
```

```
    InvoiceID INT IDENTITY(1,1) PRIMARY KEY,
    AccountID INT NOT NULL,
    InvoiceDate DATE DEFAULT GETDATE(),
    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);
```

📂جدول آیتم‌های فروش(SalesItems)

```
CREATE TABLE SalesItems (
    ItemID INT IDENTITY(1,1) PRIMARY KEY,
    InvoiceID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL,
    UnitPrice DECIMAL(18, 2) NOT NULL,
    FOREIGN KEY (InvoiceID) REFERENCES SalesInvoices(InvoiceID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

🔷۱.۶ نرمال‌سازی(Normalization)

نرمال‌سازی فرآیندی است برای ساختاردهی بهتر جداول:

- 1NF: هیچ ستونی نباید چند مقدار داشته باشد.

- 2NF: تمام ستون‌های غیرکلیدی باید به کل کلید اصلی وابسته باشند.
- 3NF: نباید وابستگی غیرمستقیم بین ستون‌های غیرکلیدی وجود داشته باشد.

🔷۱.۷ قوانین طراحی صحیح درSQL Server

✅استفاده از IDENTITY برای کلید اصلی
✅استفاده از NVARCHAR برای پشتیبانی از زبان فارسی
✅تعیین DEFAULT برای مقادیر ابتدایی
✅استفاده از CHECK برای کنترل صحت مقادیر
✅تعریف کلید خارجی با FOREIGN KEY برای اتصال جداول

✅جمع‌بندی فصل اول

در این فصل:

- با مفاهیم طراحی دیتابیس آشنا شدیم
- روابط بین موجودیت‌ها را بررسی کردیم
- ساختار کامل سیستم خرید و فروش را طراحی کردیم
- جداول مورد نیاز را با سینتکس Microsoft SQL Server ایجاد کردیم

فصل دوم: مفاهیم پایه‌ای SQL در Microsoft SQL Server

Chapter 2: SQL Fundamentals in SQL Server

در این فصل با دستورات پایه‌ای SQL Server (T-SQL) آشنا می‌شویم که برای کار با جداول، درج داده، بازیابی اطلاعات، بروزرسانی و حذف رکوردها استفاده می‌شوند. مثال‌ها بر اساس جداول سیستم خرید و فروش از فصل اول طراحی شده‌اند.

◆ ۲.۱ درج داده با INSERT INTO

```
-- درج کالاها
INSERT INTO Products (Name, Price, Stock)
VALUES
(N'ماوس بی‌سیم', 850000, 20),
(N'کیبورد مکانیکی', 1200000, 15),
(N'لپ‌تاپ ایسوس', 35000000, 8);
```

```
-- درج طرف حساب‌ها
INSERT INTO Accounts (Name, Type, Phone)
VALUES
(N'فروشگاه یکتا', 'Supplier', '09121111111'),
```

(N'شرکت داده‌پرداز آریا', '09129999999', 'Customer');

-- درج فاکتور فروش (فرض بر این است که AccountID = 2 مربوط به مشتری است)

INSERT INTO SalesInvoices (AccountID)

VALUES;(2)

-- درج آیتم‌های فاکتور فروش

INSERT INTO SalesItems (InvoiceID, ProductID, Quantity, UnitPrice)

VALUES

(1, 1, 2, 850000), -- ۲ ماوس بی‌سیم

(1, 3, 1, 35000000); -- ۱ لپ‌تاپ ایسوس

🔷 ۲.۲ بازیابی داده‌ها با SELECT

-- نمایش تمام کالاها

SELECT * FROM Products;

-- نمایش نام و قیمت کالاها

SELECT Name, Price FROM Products;

-- نمایش کالاهایی با قیمت بیش از ۲ میلیون

```
SELECT * FROM Products
WHERE Price > 2000000;
```

◆ ۲.۳ بروزرسانی داده‌ها با UPDATE

```
-- افزایش موجودی یک کالا
UPDATE Products
SET Stock = Stock + 10
WHERE Name = N'ماوس بی‌سیم';

-- تغییر شماره تماس طرف حساب
UPDATE Accounts
SET Phone = '09125555555'
WHERE Name = N'شرکت داده‌پرداز آریا';
```

◆ ۲.۴ حذف داده‌ها با DELETE

```
-- حذف یک آیتم فروش خاص
DELETE FROM SalesItems
WHERE ItemID = 2;

-- حذف یک کالا خاص
DELETE FROM Products
WHERE Name = N'کیبورد مکانیکی';
```

🔷 ۲.۵ ایجاد شرط در انتخاب داده‌ها با WHERE

-- نمایش کالاهایی که موجودی آن‌ها کمتر از ۱۰ عدد است

```
SELECT * FROM Products

WHERE Stock < 10;
```

-- نمایش طرف حساب‌هایی که مشتری هستند

```
SELECT * FROM Accounts

WHERE Type = 'Customer;'
```

🔷 ۲.۶ استفاده از TOP برای محدودسازی نتایج

-- نمایش ۳ کالای اول

```
SELECT TOP 3 * FROM Products;
```

🔷 ۲.۷ گرفتن تاریخ فعلی با GETDATE()

-- نمایش فاکتورهای فروش با تاریخ

```
SELECT InvoiceID, InvoiceDate

FROM SalesInvoices

WHERE InvoiceDate >= GETDATE() - 30 -- فاکتورهای ۳۰ روز اخیر ;
```

✅ جمع‌بندی فصل دوم

در این فصل آموختیم چگونه در SQL Server:

عملیات	دستور

درج داده INSERT INTO

بازیابی داده SELECT

بروزرسانی داده UPDATE

حذف داده DELETE

شرط‌گذاری WHERE

محدودسازی TOP, GETDATE()

در فصل سوم، وارد فیلتر کردن داده‌ها با شروط ترکیبی (IN, BETWEEN, LIKE, AND, OR) می‌شویم.

📘فصل سوم: فیلتر کردن داده‌ها با استفاده از شروط درSQL Server

Chapter 3: Filtering Data with Conditions in SQL Server

◆۳.۱ استفاده از WHEREبرای اعمال شرط

دستور WHEREبرای فیلتر کردن ردیف‌ها بر اساس یک شرط خاص استفاده می‌شود.

--کالاهایی که قیمت آن‌ها بیشتر از ۲ میلیون تومان است

```
SELECT * FROM Products
WHERE Price > 2000000;
```

--طرف حساب‌هایی که نوع آن‌ها Customer است

```
SELECT * FROM Accounts
WHERE Type = 'Customer';
```

◆۳.۲ ترکیب شروط با AND و OR

- AND: همه شروط باید برقرار باشند
- OR: کافی است فقط یکی از شروط برقرار باشد

```
--کالاهایی که قیمت‌شان بیشتر از ۲ میلیون و موجودی کمتر از ۱۰ است
SELECT * FROM Products
WHERE Price > 2000000 AND Stock < 10;

--کالاهایی که یا قیمت‌شان کمتر از ۱ میلیون است یا نام آن‌ها شامل "ماوس" است
SELECT * FROM Products
WHERE Price < 1000000 OR Name LIKE N'%ماوس%';
```

◆۳.۳ استفاده از IN برای بررسی چند مقدار

```
--نمایش کالاهایی با ProductID برابر با 1 یا 3 یا 5
SELECT * FROM Products
WHERE ProductID IN (1, 3, 5);
```

--طرف حساب‌هایی که نوع آن‌ها Customer یا Supplier است

```
SELECT * FROM Accounts
WHERE Type IN ('Customer', 'Supplier');
```

◆۳.۴ استفاده از BETWEENبرای بررسی بازه‌ها

--کالاهایی با قیمت بین ۵۰۰ هزار تا ۳ میلیون تومان

```
SELECT * FROM Products
WHERE Price BETWEEN 500000 AND 3000000;
```

--فاکتورهایی ثبت شده در سال ۲۰۲۴

```
SELECT * FROM SalesInvoices
WHERE InvoiceDate BETWEEN '2024-01-01' AND '2024-12-31';
```

◆۳.۵ جستجوی الگوها با LIKE

LIKEبرای جستجوی رشته‌ای با استفاده از الگو به‌کار می‌رود.

نماد	معنی
%	هر تعداد کاراکتر

نماد	معنی
_	فقط یک کاراکتر

--کالاهایی که نام آن‌ها با "لپ‌تاپ" شروع می‌شود

```
SELECT * FROM Products
WHERE Name LIKE N'لپ‌تاپ%';
```

--کالاهایی که حاوی "بی‌سیم" هستند

```
SELECT * FROM Products
WHERE Name LIKE N'%بی‌سیم%';
```

--کالاهایی که نام آن‌ها دقیقاً ۵ حرف دارد

```
SELECT * FROM Products
WHERE Name LIKE '_____';
```

◆۳.۶ بررسی مقدار تهی با IS NULL و IS NOT NULL

--طرف حساب‌هایی که شماره تماس ثبت نشده دارند

```
SELECT * FROM Accounts
WHERE Phone IS NULL;

--طرف حساب‌هایی که شماره تماس دارند
SELECT * FROM Accounts
WHERE Phone IS NOT NULL;
```

✅جمع‌بندی فصل سوم

عملگر	کاربرد
WHERE	اعمال شرط روی رکوردها
AND, OR	ترکیب شروط
IN	بررسی عضویت در مجموعه‌ای از مقادیر
BETWEEN	بررسی بازه عددی یا تاریخی
LIKE	جستجوی رشته‌ای با الگو
IS NULL, IS NOT NULL	بررسی تهی بودن یا نبودن مقدار

در فصل بعد، سراغ **مرتب‌سازی و محدودسازی نتایج** می‌رویم با استفاده از ORDER BY و TOPو همچنین روش‌های صفحه‌بندی.(Pagination)

📘فصل چهارم: مرتب‌سازی و محدودسازی نتایج درSQL Server

Chapter 4: Sorting and Limiting Results in SQL Server

مرتب‌سازی و محدود کردن نتایج برای ساخت گزارش‌های کاربردی بسیار اهمیت دارد، مخصوصاً زمانی که بخواهید فقط **بالاترین یا پایین‌ترین مقادیر** را ببینید یا داده‌ها را برای **صفحه‌بندی** (Pagination) مدیریت کنید.

🔷۴.۱ مرتب‌سازی با ORDER BY

دستور ORDER BYنتایج را بر اساس یک یا چند ستون، به صورت **صعودی** (ASC) یا **نزولی** (DESC) مرتب می‌کند.

```
--کالاها را بر اساس قیمت صعودی نمایش بده

SELECT * FROM Products

ORDER BY Price ASC;


--کالاها را بر اساس موجودی نزولی مرتب کن

SELECT * FROM Products

ORDER BY Stock DESC;
```

🔷۴.۲ مرتب‌سازی بر اساس چند ستون

```
--ابتدا بر اساس موجودی، سپس در صورت تساوی، بر اساس قیمت
SELECT * FROM Products
ORDER BY Stock DESC, Price ASC;
```

۴.۳🔷 محدودسازی نتایج با TOP

درSQL Server ، برای نمایش تعداد محدودی از ردیف‌ها، از کلمه کلیدی TOPاستفاده می‌کنیم.

```
--نمایش ۳ کالای اول (از نظر ترتیب)
SELECT TOP 3 * FROM Products
ORDER BY Price DESC;
```

۴.۴🔷 صفحه‌بندی با OFFSET ... FETCH

در SQL Server 2012 به بعد، می‌توان از OFFSETو FETCHبرای پیاده‌سازی صفحه‌بندی استفاده کرد.

```
--نمایش صفحه دوم (ردیف‌های ۶ تا ۱۰) با فرض ۵ ردیف در هر صفحه
SELECT * FROM Products
ORDER BY ProductID
OFFSET 5 ROWS
FETCH NEXT 5 ROWS ONLY;
```

🔷۴.۵ مرتب‌سازی آیتم‌های فروش به‌صورت سفارشی

--نمایش آیتم‌های فروش بر اساس قیمت واحد نزولی

SELECT * FROM SalesItems

ORDER BY UnitPrice DESC;

✅جمع‌بندی فصل چهارم

دستور	کاربرد
ORDER BY	مرتب‌سازی نتایج بر اساس ستون دلخواه
ASC / DESC	صعودی / نزولی
TOP N	محدودسازی تعداد نتایج
OFFSET ... FETCH	صفحه‌بندی پیشرفته

در فصل بعد، به موضوع **نام مستعار (Alias) برای ستون‌ها و جداول با استفاده از AS** می‌پردازیم، که برای ساده‌سازی و خوانایی بیشتر کوئری‌ها در گزارش‌ها بسیار کاربرد دارد.

🟦فصل پنجم: استفاده از نام مستعار (Alias) درSQL Server

Chapter 5: Using Aliases with AS in SQL Server

نام مستعار (Alias) در SQL Server به شما اجازه می‌دهد برای ستون‌ها و جداول، **نام‌های کوتاه‌تر، ساده‌تر یا خواناتر** تعریف کنید. این قابلیت به‌ویژه در گزارش‌گیری و استفاده از JOINبسیار مفید است.

🔷۵.۱ نام مستعار برای ستون‌ها با AS

برای تغییر نام ستون در خروجی (بدون تغییر در ساختار جدول)، از ASاستفاده می‌شود.

--نمایش نام کالا و قیمت با عنوان‌های سفارشی

```
SELECT Name AS ProductName, Price AS UnitPrice
FROM Products;
```

🔶می‌توانید از ASاستفاده نکنید و فقط فاصله بگذارید (هر دو معتبرند):

```
SELECT Name ProductName, Price UnitPrice
FROM Products;
```

🔷۵.۲ استفاده از Alias در توابع

--محاسبه مبلغ کل هر آیتم فروش

```
SELECT Quantity * UnitPrice AS TotalPrice
```

```
FROM SalesItems;
```

◆۵.۳ نام مستعار برای جداول درJOIN

استفاده از نام مستعار برای جداول، کوئری‌ها را کوتاه‌تر و خواناتر می‌کند، به‌ویژه وقتی چندین جدول در JOINاستفاده می‌شود.

```
--نمایش نام کالا و مقدار فروخته‌شده

SELECT p.Name AS ProductName, si.Quantity

FROM SalesItems si

JOIN Products p ON si.ProductID = p.ProductID;
```

◆۵.۴ استفاده از Alias درSubquery

```
--زیرپرس‌وجو با نام مستعار

SELECT ProductName, TotalSales

FROM (

    SELECT Name AS ProductName,

        SUM(Quantity * UnitPrice) AS TotalSales

    FROM Products p
```

```
    JOIN SalesItems si ON p.ProductID = si.ProductID
    GROUP BY Name
) AS SalesSummary;
```

✅جمع‌بندی فصل پنجم

استفاده	توضیح
ASبرای ستون	نمایش عنوان دلخواه در خروجی
ASبرای جدول	سادگی در JOINو Subquery
بدون AS هم قابل‌استفاده است	اما توصیه می‌شود برای خوانایی استفاده شود

در فصل بعد، سراغ **توابع تجمیعی (Aggregate Functions)** می‌رویم: SUM, COUNT, AVG, MIN, MAXو نحوه استفاده آن‌ها در گزارش‌گیری از داده‌ها.

📘فصل ششم: توابع تجمیعی (Aggregate Functions) درSQL Server

Chapter 6: Aggregate Functions in SQL Server

توابع تجمیعی در SQL Server برای انجام محاسبات روی **مجموعه‌ای از ردیف‌ها** استفاده می‌شوند، مثل: مجموع قیمت‌ها، میانگین مقادیر، شمارش ردیف‌ها و یافتن بیشینه یا کمینه.

◆۶.۱ تابع – COUNT()شمارش تعداد ردیف‌ها

```
--تعداد کل کالاها

SELECT COUNT(*) AS TotalProducts

FROM Products;


--تعداد مشتری‌ها

SELECT COUNT(*) AS TotalCustomers

FROM Accounts

WHERE Type = 'Customer';
```

◆۶.۲ تابع – SUM()جمع کل

```
--مجموع قیمت کالاها

SELECT SUM(Price) AS TotalPrice

FROM Products;
```

--مجموع مبلغ یک فاکتور فروش خاص

```
SELECT SUM(Quantity * UnitPrice) AS InvoiceTotal
FROM SalesItems
WHERE InvoiceID = 1;
```

◆۶.۳ تابع AVG() – میانگین

--میانگین قیمت کالاها

```
SELECT AVG(Price) AS AveragePrice
FROM Products;
```

◆۶.۴ تابع MIN() و MAX() – کمینه و بیشینه

--کمترین قیمت

```
SELECT MIN(Price) AS CheapestProduct
FROM Products;
```

--بیشترین قیمت

```
SELECT MAX(Price) AS MostExpensiveProduct
FROM Products;
```

🔷۶.۵ ترکیب با شرط WHERE

```
--مجموع فروش فاکتورهایی که بعد از سال 2024 ثبت شده‌اند
SELECT SUM(Quantity * UnitPrice) AS RecentSales
FROM SalesItems si
JOIN SalesInvoices sih ON si.InvoiceID = sih.InvoiceID
WHERE sih.InvoiceDate >= '2024-01-01';
```

🔷۶.۶ استفاده از DISTINCT در توابع تجمیعی

```
--sتعداد انواع کالاهای منحصربه‌فرد فروخته شده
SELECT COUNT(DISTINCT ProductID) AS UniqueProductsSold
FROM SalesItems;
```

✅جمع‌بندی فصل ششم

تابع	توضیح
COUNT()	شمارش ردیف‌ها
SUM()	جمع مقادیر عددی
AVG()	میانگین

تابع	توضیح
MIN() / MAX()	کوچک‌ترین / بزرگ‌ترین مقدار
JOINو WHEREهمراه با	برای تحلیل‌های دقیق‌تر
DISTINCT	برای حذف تکرار در شمارش

در فصل بعد، با استفاده از GROUP BYیاد می‌گیریم چگونه این توابع را به صورت **گروه‌بندی‌شده** روی داده‌ها اعمال کنیم، مثلاً مجموع فروش به تفکیک کالا یا مشتری.

📘فصل هفتم: گروه‌بندی داده‌ها با GROUP BYو فیلتر کردن گروه‌ها با HAVINGدر SQL Server

Chapter 7: GROUP BY and HAVING in SQL Server

وقتی می‌خواهیم توابع تجمیعی مانند SUM()یا COUNT()را **برای هر گروه از داده‌ها** (مثلاً به ازای هر کالا یا هر مشتری) محاسبه کنیم، از GROUP BYاستفاده می‌کنیم. اگر بخواهیم روی همین گروه‌ها شرط اعمال کنیم، از HAVINGاستفاده می‌شود.

◆۷.۱ استفاده از GROUP BY

```
--مجموع فروش به تفکیک کالا

SELECT ProductID, SUM(Quantity * UnitPrice) AS TotalSales

FROM SalesItems

GROUP BY ProductID;
```

🔶بهتر است با اطلاعات کالا ترکیب شود:

--مجموع فروش بر اساس نام کالا

SELECT p.Name AS ProductName, SUM(si.Quantity * si.UnitPrice) AS TotalSales

FROM SalesItems si

JOIN Products p ON si.ProductID = p.ProductID

GROUP BY p.Name;

🔷۷.۲ شمارش به تفکیک گروه‌ها

--تعداد فاکتورهای ثبت‌شده برای هر مشتری

SELECT a.Name AS CustomerName, COUNT(si.InvoiceID) AS InvoiceCount

FROM SalesInvoices sih

JOIN Accounts a ON sih.AccountID = a.AccountID

GROUP BY a.Name;

🔷۷.۳ استفاده از HAVINGبرای فیلتر روی گروه‌ها

بر خلاف WHEREکه روی ردیف‌ها عمل می‌کند، HAVINGبرای فیلتر کردن **نتایج گروه‌بندی‌شده** استفاده می‌شود.

sql

--کالاهایی که مجموع فروش آن‌ها بیش از ۱۰ میلیون است

```
SELECT p.Name AS ProductName, SUM(si.Quantity * si.UnitPrice) AS TotalSales
FROM SalesItems si
JOIN Products p ON si.ProductID = p.ProductID
GROUP BY p.Name
HAVING SUM(si.Quantity * si.UnitPrice) > 10000000;
```

◆۷.۴ تفاوت WHERE و HAVING

ویژگی	WHERE	HAVING
اجرا قبل از GROUP BY	✅	❌
اجرا بعد از GROUP BY	❌	✅
می‌توان از توابع تجمیعی استفاده کرد	❌	✅

ویژگی	WHERE	HAVING
مخصوص فیلتر ردیف‌ها	✅	❌
مخصوص فیلتر گروه‌ها	❌	✅

✅جمع‌بندی فصل هفتم

مفهوم	کاربرد
GROUP BY	دسته‌بندی ردیف‌ها برای محاسبه جمع، میانگین، و...
HAVING	فیلتر روی نتایج گروه‌بندی‌شده
ترکیب با JOIN	برای نمایش نام کالا، مشتری و... به‌جای فقط شناسه‌ها
توابع تجمیعی در گزارش گروهی	مثل فروش به تفکیک کالا یا مشتری

🟦فصل هشتم: استفاده از ساختار شرطی CASEدرSQL Server

Chapter 8: Conditional Logic with CASE in SQL Server

ساختار CASEدر SQL Server به شما امکان می‌دهد بر اساس شرط‌هایی مشخص، **مقادیر متفاوتی را در خروجی نمایش دهید** .این قابلیت مشابه دستور IF...ELSEدر

زبان‌های برنامه‌نویسی است و برای دسته‌بندی، ایجاد گزارش‌های سفارشی و محاسبات شرطی بسیار کاربردی است.

◆۸.۱ ساختار کلی CASE

```
CASE
    WHEN شرط THEN مقدار
    [WHEN شرط THEN مقدار]
    [ELSE مقدار پیش‌فرض]
END
```

◆۸.۲ مثال: دسته‌بندی قیمت کالاها

```
SELECT Name,
       Price,
       CASE
           WHEN Price < 1000000 THEN N'ارزان'
           WHEN Price BETWEEN 1000000 AND 5000000 THEN N'متوسط'
           ELSE N'گران'
       END AS PriceCategory
```

```
FROM Products;
```

◆۸.۳ استفاده از CASEدر محاسبات فروش

```
--نمایش مالیات بر اساس قیمت فروش هر آیتم
SELECT si.Quantity, si.UnitPrice,
    si.Quantity * si.UnitPrice AS TotalPrice,
    CASE
        WHEN si.UnitPrice > 10000000 THEN N'شامل مالیات'
        ELSE N'بدون مالیات'
    END AS TaxStatus
FROM SalesItems si;
```

◆۸.۴ استفاده از CASEدر مرتب‌سازی با ORDER BY

```
--مرتب‌سازی کالاها به ترتیب اول: گران، دوم: متوسط، سوم: ارزان
SELECT Name, Price
FROM Products
ORDER BY
  CASE
```

```
    WHEN Price >= 5000000 THEN 1
    WHEN Price BETWEEN 1000000 AND 4999999 THEN 2
    ELSE 3     END;
```

🔷۸.۵ استفاده از CASEدر UPDATE

```
--افزایش موجودی کالاها بر اساس قیمت آن‌ها
UPDATE Products
SET Stock = Stock +
  CASE
    WHEN Price < 1000000 THEN 10
    WHEN Price BETWEEN 1000000 AND 5000000 THEN 5
    ELSE 2
  END;
```

✅جمع‌بندی فصل هشتم

کاربرد CASE	مثال
دسته‌بندی داده‌ها	دسته‌بندی کالاها به ارزان/متوسط/گران
شرط در ستون‌های خروجی	نمایش وضعیت مالیات

کاربرد CASE	مثال
مرتب‌سازی سفارشی	مرتب‌سازی بر اساس دسته‌بندی قیمت
شرط در UPDATE	بروزرسانی مقادیر بر اساس شرط

ساختار CASEبسیار منعطف است و در بیشتر بخش‌های) SQL درون SELECT, ORDER BY, UPDATE، و ...) قابل استفاده است.

در فصل بعد، وارد دنیای **اتصال جداول (Joins)** می‌شویم که یکی از مهم‌ترین ابزارها در SQLبرای ساخت گزارش‌های ترکیبی است.

📘فصل نهم: اتصال جداول با JOINدرSQL Server

Chapter 9: Joining Tables in SQL Server

درSQL ، اطلاعات معمولاً در جداول جداگانه ذخیره می‌شوند. برای ساخت گزارش‌های کامل یا بازیابی اطلاعات مرتبط، باید این جداول را به هم **وصل کنیم** .این کار با استفاده از JOINانجام می‌شود.

🔷۹.۱ انواع JOINدرSQL Server

نوع JOIN	توضیح
INNER JOIN	فقط رکوردهای مشترک بین دو جدول را برمی‌گرداند
LEFT JOIN	همه رکوردهای جدول چپ + رکوردهای مرتبط از جدول راست
RIGHT JOIN	همه رکوردهای جدول راست + رکوردهای مرتبط از جدول چپ
FULL JOIN	همه رکوردهای هر دو جدول (مشترک و غیرمشترک)
CROSS JOIN	ضرب کارتزین – ترکیب همه رکوردهای دو جدول با هم

◆۹.۲ مثال با INNER JOIN

```
--نمایش نام کالا و تعداد فروش آن در هر آیتم فاکتور

SELECT si.InvoiceID,

    p.Name AS ProductName,

    si.Quantity

FROM SalesItems si

INNER JOIN Products p ON si.ProductID = p.ProductID;
```

◆۹.۳ مثال با LEFT JOIN

```
--نمایش همه کالاها و تعداد فروش‌شان (در صورت وجود)

SELECT p.Name,
```

```
    si.Quantity
FROM Products p
LEFT JOIN SalesItems si ON p.ProductID = si.ProductID;
```

در این مثال، اگر کالایی تاکنون فروخته نشده باشد، مقدار Quantity برابر با NULL خواهد بود.

◆۹.۴ مثال با RIGHT JOIN

```
--نمایش همه آیتم‌های فروش به همراه اطلاعات کالا (در صورت وجود کالا)
SELECT si.InvoiceID, si.Quantity, p.Name AS ProductName
FROM SalesItems si
RIGHT JOIN Products p ON si.ProductID = p.ProductID;
```

◆۹.۵ مثال با FULL JOIN

```
--نمایش همه کالاها و آیتم‌های فروش، حتی آن‌هایی که ارتباط ندارند
SELECT p.Name AS ProductName, si.Quantity
FROM Products p
FULL JOIN SalesItems si ON p.ProductID = si.ProductID;
```

◆۹.۶ مثال با CROSS JOIN

--ترکیب همه کالاها با همه طرف حساب‌ها (برای تست یا ساخت داده‌های فرضی)

```
SELECT p.Name AS ProductName, a.Name AS AccountName
FROM Products p
CROSS JOIN Accounts a;
```

🔷 ۹.۷ اتصال چند جدول هم‌زمان

--نمایش نام مشتری، نام کالا، تعداد، و مبلغ فروش

```
SELECT a.Name AS CustomerName,
    p.Name AS ProductName,
    si.Quantity,
    si.Quantity * si.UnitPrice AS TotalPrice
FROM SalesItems si
JOIN Products p ON si.ProductID = p.ProductID
JOIN SalesInvoices sih ON si.InvoiceID = sih.InvoiceID
JOIN Accounts a ON sih.AccountID = a.AccountID;
```

✅ جمع‌بندی فصل نهم

نوع JOIN	کاربرد
INNER JOIN	فقط داده‌های مشترک
LEFT JOIN	تمام رکوردهای جدول اول، حتی اگر ارتباطی نباشد
RIGHT JOIN	تمام رکوردهای جدول دوم
FULL JOIN	همه رکوردهای هر دو جدول
CROSS JOIN	ضرب کارتزین برای تولید ترکیب کامل

اتصال جداول بخش اساسی از SQL است و مهارت در JOINها برای تحلیل حرفه‌ای داده‌ها ضروری است.

در فصل بعد، با مفهوم **زیردرخواست‌ها (Subqueries)** آشنا می‌شویم که به شما امکان می‌دهد کوئری‌های تو در تو بسازید و از خروجی یک کوئری به عنوان ورودی کوئری دیگر استفاده کنید.

📘فصل دهم: زیردرخواست‌ها (Subqueries) در SQL Server

Chapter 10: Subqueries in SQL Server

زیردرخواست یا Subquery کوئری‌ای است که **درون کوئری دیگر** قرار می‌گیرد و خروجی آن می‌تواند برای فیلتر کردن، محاسبه یا مقداردهی در کوئری اصلی استفاده شود .
Subqueryها ابزار بسیار قدرتمندی برای ساخت کوئری‌های پیچیده و پویا هستند.

۱۰.۱ انواع Subquery🔷

نوع	کاربرد
در SELECT	برای محاسبه مقدار برای هر ردیف
در WHERE	برای فیلتر کردن نتایج
در FROM	ایجاد جدول موقتی یا View مجازی
هم‌بسته(Correlated)	وابسته به هر ردیف از کوئری بیرونی

۱۰.۲ Subquery در WHERE

--نمایش کالاهایی که قیمت آن‌ها بیشتر از میانگین قیمت همه کالاهاست

```
SELECT Name, Price

FROM Products

WHERE Price > (

   SELECT AVG(Price) FROM Products

);
```

۱۰.۳ Subquery در SELECT

--نمایش کالاها همراه با میانگین قیمت همه کالاها

```
SELECT Name, Price,

    (SELECT AVG(Price) FROM Products) AS AveragePrice
```

```
FROM Products;
```

۱۰.۴ Subquery در FROM

```
--نمایش کالاهایی با قیمت بیشتر از ۲ میلیون با استفاده از جدول موقتی
SELECT *
FROM (
    SELECT Name, Price FROM Products
) AS Temp
WHERE Temp.Price > 2000000;
```

۱۰.۵ Subquery هم‌بسته(Correlated)

```
--نمایش کالاهایی که حداقل یک‌بار فروخته شده‌اند
SELECT Name
FROM Products p
WHERE EXISTS (
    SELECT 1
    FROM SalesItems si
    WHERE si.ProductID = p.ProductID
);
```

🔸این Subquery به مقدار ProductIDاز کوئری بیرونی وابسته است، و برای هر ردیف اجرا می‌شود.

🔷۱۰.۶ استفاده از Subquery با IN, NOT IN, EXISTS

```
--نمایش کالاهایی که هیچ‌گاه فروخته نشده‌اند

SELECT Name

FROM Products

WHERE ProductID NOT IN (

   SELECT ProductID FROM SalesItems

);
```

✅جمع‌بندی فصل دهم

کاربردSubquery	نمونه
در SELECT	افزودن ستون محاسباتی
در WHERE	فیلتر کردن بر اساس نتایج دیگر
در FROM	ساختن جدول موقتی
هم‌بسته	بررسی وجود یا شرط خاص به ازای هر ردیف
با IN, EXISTS, NOT IN	تطبیق با لیستی از نتایج

Subqueryها پایه تحلیل‌های پیشرفته در SQL هستند و با تمرین مداوم می‌توان آن‌ها را به خوبی به‌کار گرفت.

در فصل بعد، با **Viewsنماها** (و **Indexesنمایه‌ها**) (آشنا می‌شویم که برای ساده‌سازی گزارش‌ها و بهینه‌سازی سرعت کوئری‌ها کاربرد دارند.

📘فصل یازدهم: نماها (Views) و نمایه‌ها (Indexes) در SQL Server

Chapter 11: Views and Indexes in SQL Server

در این فصل با دو ابزار بسیار مهم برای سازماندهی بهتر و عملکرد بالاتر پایگاه داده آشنا می‌شویم:

- **View**: یک جدول مجازی که از نتیجه یک SELECT ساخته می‌شود
- **Index**: ساختاری برای افزایش سرعت جستجو و اجرای کوئری‌ها

🟩بخش اول View :(نما)

🔷۱۱.۱ View چیست؟

VIEWیک جدول مجازی است که از نتیجه اجرای یک SELECTساخته می‌شود. اطلاعات آن از جداول واقعی گرفته می‌شود و خودش داده‌ای ذخیره نمی‌کند.

🔷۱۱.۲ مزایای استفاده ازView

✅سادەسازی گزارش‌ها
✅جلوگیری از تکرار کوئری‌های پیچیده
✅افزایش امنیت (محدودسازی دسترسی به ستون‌ها)
✅استفاده آسان در نرم‌افزارهای خارجی

🔷۱۱.۳ ایجاد یک View ساده

```
CREATE VIEW vw_SalesReport AS
SELECT si.InvoiceID,
    p.Name AS ProductName,
    si.Quantity,
    si.UnitPrice,
    si.Quantity * si.UnitPrice AS TotalPrice
FROM SalesItems si
JOIN Products p ON si.ProductID = p.ProductID;
```

۱۱.۴ استفاده ازView

```
SELECT * FROM vw_SalesReport;


-- فیلتر بر روی View
SELECT * FROM vw_SalesReport
WHERE TotalPrice > 10000000;
```

۱۱.۵ حذف یکView

```
DROP VIEW vw_SalesReport;
```

بخش دوم Index :نمایه

۱۱.۶ Index چیست؟

INDEXساختاری است که دسترسی به داده‌ها را سریع‌تر می‌کند. مانند فهرست یک کتاب عمل می‌کند. با ایندکس، SQL Serverمی‌تواند ردیف‌های مرتبط را سریع‌تر پیدا کند.

۱۱.۷ مزایا و معایبIndex

مزایا	معایب
افزایش سرعت جستجو	مصرف فضای بیشتر
بهبود عملکرد JOINو WHERE	کند شدن عملیات INSERT, UPDATE, DELETE

۱۱.۸ ایجاد Index درSQL Server

--ایجاد ایندکس روی نام کالا

```
CREATE INDEX idx_ProductName
ON Products(Name);
```

--ایندکس ترکیبی (برای جستجوهای پیچیده‌تر)

```
CREATE INDEX idx_SalesInvoice_Product
ON SalesItems(InvoiceID, ProductID);
```

۱۱.۹ حذفIndex

```
DROP INDEX idx_ProductName ON Products;
```

✅جمع‌بندی فصل یازدهم

ابزار	کاربرد
VIEW	ساخت گزارش‌های از پیش‌تعریف‌شده باSELECT
INDEX	افزایش سرعت دسترسی به داده‌ها، مخصوصاً در جداول بزرگ

ویو ابزار گزارش‌سازی است.

ایندکس ابزار بهینه‌سازی سرعت است.

در فصل بعد، با **توابع رشته‌ای (String Functions)** آشنا می‌شویم که برای پردازش متن‌ها مثل نام، شماره، ایمیل و... بسیار مفید هستند.

📘فصل دوازدهم: توابع رشته‌ای (String Functions) درSQL Server

Chapter 12: String Functions in SQL Server

توابع رشته‌ای (String Functions) برای **بررسی، اصلاح، ترکیب یا جدا کردن داده‌های متنی** مثل نام کالا، شماره تماس، ایمیل و... استفاده می‌شوند. این توابع در SQL Serverبسیار متنوع و قدرتمند هستند.

◆۱۲.۱ تابع – LEN()طول رشته

```
--تعداد کاراکترهای نام کالا

SELECT Name, LEN(Name) AS Length

FROM Products;
```

۱۲.۲ تابع UPPER() و LOWER() – تبدیل به حروف بزرگ/کوچک

```
--نام کالا با حروف بزرگ

SELECT UPPER(Name) AS UpperName

FROM Products;

--نام مشتری با حروف کوچک

SELECT LOWER(Name) AS LowerName

FROM Accounts;
```

۱۲.۳ تابع LTRIM() و RTRIM() – حذف فاصله از ابتدا و انتهای رشته

```
--حذف فاصله اضافی سمت چپ

SELECT LTRIM('   کیبورد') AS Result;

--حذف فاصله اضافی سمت راست

SELECT RTRIM('ماوس   ') AS Result;
```

۱۲.۴ تابع TRIM() – در SQL Server 2017 به بعد

```
SELECT TRIM('   لپ‌تاپ  ') AS Cleaned;
```

۱۲.۵ تابع REPLACE() – جایگزینی زیررشته

--جایگزینی "لپ‌تاپ" با "نوت‌بوک"

SELECT REPLACE(Name, N'لپ‌تاپ', N'نوت‌بوک') AS NewName

FROM Products;

۱۲.۶ تابع SUBSTRING() – استخراج بخشی از رشته

--گرفتن ۳ کاراکتر اول از نام کالا

SELECT SUBSTRING(Name, 1, 3) AS ShortName

FROM Products;

۱۲.۷ اتصال رشته‌ها با +

--اتصال نام و نوع طرف حساب

SELECT Name + ' - ' + Type AS FullDescription

FROM Accounts;

۱۲.۸ تابع CHARINDEX() – موقعیت یک رشته در رشته دیگر

--پیدا کردن موقعیت کلمه "بی‌سیم" در نام کالا

SELECT Name, CHARINDEX(N'بی‌سیم', Name) AS Position

FROM Products;

✅جمع‌بندی فصل دوازدهم

تابع	کاربرد
LEN()	طول رشته
UPPER() / LOWER()	تبدیل به حروف بزرگ/کوچک
LTRIM() / RTRIM()	حذف فاصله چپ/راست
TRIM()	حذف فاصله ابتدا و انتها(+2017)
REPLACE()	جایگزینی متن
SUBSTRING()	جدا کردن بخش از متن
CHARINDEX()	یافتن محل یک کلمه در رشته
+	اتصال متون

در فصل بعد، به سراغ **توابع عددی و ریاضی** می‌رویم که برای محاسبات قیمت، تخفیف، گرد کردن، و تولید عدد تصادفی کاربرد دارند.

📘فصل سیزدهم: توابع عددی و ریاضی درSQL Server

Chapter 13: Numeric and Mathematical Functions in SQL Server

توابع ریاضی در SQL Server به شما اجازه می‌دهند انواع محاسبات عددی را روی ستون‌ها و مقادیر انجام دهید. این توابع در قیمت‌گذاری، تخفیف، مالیات، و تحلیل‌های آماری کاربرد زیادی دارند.

۱۳.۱ تابع – ROUND() گرد کردن عدد

```
--گرد کردن قیمت به دو رقم اعشار

SELECT Name, ROUND(Price, 2) AS RoundedPrice

FROM Products;
```

۱۳.۲ تابع – CEILING() گرد کردن به بالا

```
SELECT CEILING(3.14) AS CeilValue;  -- خروجی: 4
```

۱۳.۳ تابع – FLOOR() گرد کردن به پایین

```
SELECT FLOOR(3.99) AS FloorValue;  -- خروجی: 3
```

۱۳.۴ تابع – ABS() قدر مطلق

```
SELECT ABS(-2500) AS AbsoluteValue;  -- خروجی: 2500
```

۱۳.۵ تابع – POWER() توان عدد

```
SELECT POWER(2, 3) AS Result;  -- خروجی: 8
```

۱۳.۶ تابع – SQRT() جذر (ریشه دوم)

SELECT SQRT(144) AS SquareRoot; -- خروجی: 12

◆ ۱۳.۷ تابع RAND() – تولید عدد تصادفی

SELECT RAND() AS RandomValue; -- خروجی: عددی بین 0 و 1

◆ ۱۳.۸ محاسبات عددی در SELECT

--محاسبه مبلغ کل فروش هر آیتم

SELECT Quantity, UnitPrice,

    Quantity * UnitPrice AS TotalPrice

FROM SalesItems;

◆ ۱۳.۹ استفاده از توابع ریاضی در UPDATE

--افزایش قیمت کالاها به میزان ۱۰٪

UPDATE Products

SET Price = ROUND(Price * 1.10, 2);

✅ جمع‌بندی فصل سیزدهم

تابع	کاربرد
ROUND()	گرد کردن
CEILING() / FLOOR()	گرد کردن به بالا / پایین
ABS()	قدر مطلق
POWER() / SQRT()	توان / جذر
RAND()	عدد تصادفی بین ۰ تا ۱
محاسبات با +, -, *, /	اعمال مستقیم ریاضی

در فصل بعد، با **توابع تاریخ و زمان** در SQL Server آشنا می‌شویم که در تحلیل‌های زمانی، گزارش‌های دوره‌ای، و مدیریت اسناد تاریخی بسیار کاربرد دارند.

📘 فصل چهاردهم: توابع تاریخ و زمان در SQL Server

Chapter 14: Date and Time Functions in SQL Server

مدیریت زمان در پایگاه داده‌ها بسیار مهم است؛ مخصوصاً در سیستم‌هایی مانند صدور فاکتور، سررسید پرداخت، گزارش‌های ماهانه یا سالانه و غیره SQL Server .مجموعه‌ای از توابع برای کار با **تاریخ و زمان** فراهم کرده است.

۱۴.۱ تابع – GETDATE() تاریخ و زمان جاری سیستم

```
SELECT GETDATE() AS CurrentDateTime;
```

۱۴.۲ تابع – GETUTCDATE() تاریخ و زمان جهانی(UTC)

```
SELECT GETUTCDATE() AS UtcNow;
```

۱۴.۳ استخراج بخش‌هایی از تاریخ با DATEPART()

```
--استخراج سال از تاریخ فاکتور
SELECT InvoiceDate,
    DATEPART(YEAR, InvoiceDate) AS Year,
    DATEPART(MONTH, InvoiceDate) AS Month,
    DATEPART(DAY, InvoiceDate) AS Day
FROM SalesInvoices;
```

۱۴.۴ قالب‌بندی تاریخ با FORMAT() (از SQL Server 2012 به بعد)

```
--نمایش تاریخ به‌صورت سفارشی
SELECT FORMAT(InvoiceDate, 'yyyy/MM/dd') AS FormattedDate
FROM SalesInvoices;
```

--نمایش فقط ساعت و دقیقه

```
SELECT FORMAT(GETDATE(), 'HH:mm') AS CurrentTime;
```

۱۴.۵🔷 محاسبه اختلاف بین دو تاریخ با DATEDIFF()

تعداد روز بین تاریخ فاکتور و امروز

```
SELECT InvoiceDate,
    DATEDIFF(DAY, InvoiceDate, GETDATE()) AS DaysSinceInvoice
FROM SalesInvoices;
```

۱۴.۶🔷 افزودن زمان با DATEADD()

--نمایش تاریخ ۳۰ روز آینده

```
SELECT DATEADD(DAY, 30, GETDATE()) AS DueDate;
```

۱۴.۷🔷 گرفتن فقط بخش تاریخ با CAST یا CONVERT

--حذف زمان و نمایش فقط تاریخ

```
SELECT CAST(GETDATE() AS DATE) AS TodayOnly;
```

یا:

```
SELECT CONVERT(DATE, GETDATE()) AS TodayOnly;
```

🔷۱۴.۸ فیلتر کردن بر اساس تاریخ

--فاکتورهایی که در سال ۲۰۲۴ صادر شده‌اند

```
SELECT * FROM SalesInvoices

WHERE YEAR(InvoiceDate) = 2024;
```

--فاکتورهایی از ۳ ماه گذشته

```
SELECT * FROM SalesInvoices

WHERE InvoiceDate >= DATEADD(MONTH, -3, GETDATE());
```

✅جمع‌بندی فصل چهاردهم

تابع	کاربرد
GETDATE()	تاریخ و زمان فعلی
DATEPART()	استخراج بخش‌های تاریخ (سال، ماه، روز، ساعت و...)
FORMAT()	قالب‌بندی خروجی تاریخ/زمان
DATEDIFF()	محاسبه فاصله بین دو تاریخ
DATEADD()	افزودن فاصله زمانی
CAST / CONVERT	تبدیل نوع به DATE برای حذف زمان

تابع	کاربرد
فیلتر بر اساس تاریخ	گزارش‌گیری‌های زمانی

در فصل بعد، به سراغ **جستجوی الگوها و تطبیق رشته‌ها** با استفاده از LIKE, PATINDEX, CHARINDEXو دیگر ابزارهای جستجوی متنی در SQL Server می‌رویم.

📘فصل پانزدهم: جستجوی الگوها و تطبیق رشته‌ها درSQL Server

Chapter 15: Pattern Matching in SQL Server

در این فصل یاد می‌گیریم چطور با استفاده از عملگرها و توابع متنی مانند LIKE, PATINDEX, CHARINDEXو حتی عبارات باقاعده) در(SQL Server 2022 ، **الگوهای متنی** را جستجو و تحلیل کنیم.

🔷۱۵.۱ جستجو با LIKE

عملگر LIKEبرای **جستجوی تطبیقی ساده روی رشته‌ها** استفاده می‌شود.

نماد	توضیح
%	صفر یا چند کاراکتر دلخواه
_	فقط یک کاراکتر دلخواه

🧪مثال‌ها:

--کالاهایی که با "لپ‌تاپ" شروع می‌شوند

```
SELECT * FROM Products
WHERE Name LIKE N'لپ‌تاپ%';

--کالاهایی که حاوی "بی‌سیم" هستند
SELECT * FROM Products
WHERE Name LIKE N'%بی‌سیم%';

--کالاهایی با نام چهارحرفی
SELECT * FROM Products
WHERE Name LIKE '____';
```

◆۱۵.۲ جستجو با CHARINDEX

تابع CHARINDEX مکان اولین وقوع یک زیررشته را در یک رشته برمی‌گرداند.

```
--پیدا کردن مکان کلمه "USB" در نام کالا
SELECT Name, CHARINDEX('USB', Name) AS USB_Position
FROM Products
WHERE CHARINDEX('USB', Name) > 0;
```

۱۵.۳🔷 جستجو با PATINDEX

تابع PATINDEXمشابه CHARINDEXاست اما از **الگوهای LIKEبا %و** _پشتیبانی می‌کند.

--موقعیت اولین کلمه‌ای که با "کیبورد" شروع می‌شود

```
SELECT Name, PATINDEX(N'%کیبورد%', Name) AS Position
FROM Products
WHERE PATINDEX(N'%کیبورد%', Name) > 0;
```

۱۵.۴🔷 مقایسه LIKE, CHARINDEXو PATINDEX

تابع/عملگر	قدرت تطبیق	حساس به حروف	پشتیبانی از الگو
LIKE	تطبیق کامل	بستگی به collation دارد	بله
CHARINDEX	فقط زیررشته	بله	خیر
PATINDEX	ترکیبی از LIKEو CHARINDEX	بله	بله ✅

۱۵.۵🔷 استفاده از COLLATEبرای تنظیم حساسیت به حروف بزرگ/کوچک

--جستجوی غیر حساس به حروف

```
SELECT * FROM Products

WHERE Name COLLATE Latin1_General_CI_AI LIKE '%usb%';
```

- 🔷 CI = Case Insensitive
- 🔷 AI = Accent Insensitive

🔷۱۵.۶ جستجوی پیچیده‌تر) از (SQL Server 2022 با REGEXP_LIKE

اگر از SQL Server 2022 به بعد استفاده می‌کنید، می‌توانید از Regex نیز بهره ببرید:

--کالاهایی که شامل عدد در نام‌شان هستند

```
SELECT * FROM Products

WHERE REGEXP_LIKE(Name, '[0-9]');
```

💡اگر از نسخه‌های پایین‌تر استفاده می‌کنید، REGEXP_LIKEدر دسترس نخواهد بود.

✅جمع‌بندی فصل پانزدهم

ابزار	کاربرد
LIKE	جستجوی الگو با %و _
CHARINDEX	یافتن مکان یک زیررشته
PATINDEX	یافتن الگو در متن

ابزار	کاربرد
COLLATE	کنترل حساسیت به حروف
REGEXP_LIKE	جستجوی با قاعده) فقط(+SQL Server 2022

در فصل بعد (فصل پایانی این بخش از کتاب)، به سراغ **تبدیل نوع داده‌ها Type)** **Conversion)**می‌رویم، مثل تبدیل رشته به عدد، عدد به تاریخ، یا تاریخ به متن.

📘فصل شانزدهم: تبدیل نوع داده‌ها درSQL Server

Chapter 16: Data Type Conversion in SQL Server

درSQL Server ، گاهی لازم است نوع یک داده را به نوعی دیگر تغییر دهیم for—example، تبدیل رشته به عدد، تاریخ به متن، یا عدد به رشته. این عملیات با توابع CAST(), CONVERT(),و گاهی PARSE()انجام می‌شود.

◆۱۶.۱ تابع – CAST()تبدیل استاندارد نوع داده

--تبدیل رشته به عدد

```
SELECT CAST('12345' AS INT) AS NumberValue;
```

تبدیل عدد به رشته --

```
SELECT CAST(250000 AS VARCHAR(20)) + ' تومان' AS LabeledPrice;
```

۱۶.۲ تابع – CONVERT() تبدیل با فرمت مخصوص

```
--تبدیل تاریخ به رشته با فرمت خاص

SELECT CONVERT(VARCHAR(10), GETDATE(), 120) AS FormattedDate;  --
خروجی: 2025-04-13


--فرمت ۳(dd/mm/yy)

SELECT CONVERT(VARCHAR(10), GETDATE(), 3) AS PersianStyleDate; --
خروجی: 25/04/13
```

◆کدهای فرمت مختلف برای :CONVERT

کد	فرمت تاریخ خروجی
1	mm/dd/yy
3	dd/mm/yy
104	dd.mm.yyyy
120	yyyy-mm-dd hh:mi:ss (ISO)

۱۶.۳ تابع – PARSE() تبدیل هوشمندتر (SQL Server 2012 به بعد)

--تبدیل رشته به تاریخ

```
SELECT PARSE('2025-04-13' AS DATE USING 'en-US') AS ParsedDate;
```

🔶برای استفاده از PARSE() باید USINGرا همراه زبان فرهنگ (culture) وارد کرد.

🔷۱۶.۴ تبدیل خودکار در عملیات ریاضی یا متنی

--جمع عدد و رشته به‌صورت خودکار تبدیل می‌شود

```
SELECT 'مبلغ: ' + CAST(1000000 AS VARCHAR(20)) + ' تومان' AS PriceText;
```

🔷۱۶.۵ بررسی نوع داده با SQL_VARIANT_PROPERTY() پیشرفته

--بررسی نوع داده واقعی یک مقدار

```
SELECT SQL_VARIANT_PROPERTY(CAST(123.45 AS DECIMAL(10,2)),
'BaseType') AS DataType;
```

✅جمع‌بندی فصل شانزدهم

تابع	کاربرد
CAST()	تبدیل ساده بین انواع داده
CONVERT()	تبدیل همراه با فرمت مخصوص، مخصوصاً برای تاریخ

تابع	کاربرد
PARSE()	تبدیل پیشرفته‌تر با فرهنگ زبانی
تبدیل ضمنی	در ترکیب رشته و عدد، SQL Serverخودش تبدیل می‌کند

جادوی فکر یک شرکت پیشرو در زمینه توسعه راهکارهای نوآورانه‌ی دیجیتال، هوش مصنوعی و تحول سازمانی است. ما با تمرکز بر آموزش، تحقیق، و مشاوره، به سازمان‌ها کمک می‌کنیم تا از فناوری‌های نوظهور برای رشد، کارایی بیشتر و تصمیم‌گیری هوشمندانه بهره‌مند شوند.

تیم ما متشکل از متخصصان فناوری، تحلیل‌گران کسب‌وکار و مشاوران تحول دیجیتال است که با ترکیبی از دانش فنی و درک عمیق از نیازهای بازار، راهکارهایی جامع و عملی ارائه می‌دهند.

این کتاب با هدف ارائه‌ی بینش‌های کاربردی و آینده‌نگر درباره‌ی هوش مصنوعی عاملی گردآوری و تألیف شده است تا راهنمایی مؤثر برای مدیران، کارآفرینان، و علاقه‌مندان به فناوری باشد.

برای آشنایی بیشتر با خدمات ما، به وب‌سایت جادوی فکر سر بزنید:

www.jadoosoft.com

ساری-خیابان پیروزی- نبش پیروزی ۱۴ -ساختمان پارک علم و فناوری استان مازندران-طبقه سوم-کد پستی:۴۸۱۸۷۶۵۷۶۷

۰۱۱۳۳۲۵۹۶۵۹ – ۰۲۱۸۸۱۷۸۶۰۵ (پشتیبانی عصرها از ساعت ۱۶ تا ۲۰، ۰۹۱۰۱۰۰۹۰۰۷)

اگر به دنبال یادگیری SQL به زبانی ساده، کاربردی و پروژه‌محور هستید، این کتاب دقیقا همان چیزی‌ست که نیاز دارید. با تکیه بر مثال‌های واقعی از سیستم‌های خرید و فروش، قدم‌به‌قدم با مفاهیم اصلی طراحی پایگاه داده، دستورات پایه‌ای SQL Server، فیلتر کردن و مرتب‌سازی داده‌ها، توابع کاربردی، گزارش‌گیری، اتصال جداول، زیرپرس‌وجوها (Subqueries)، نماها (Views)، ایندکس‌ها و بسیاری ابزارهای حرفه‌ای دیگر آشنا می‌شوید.

شرکت نرم افزاری جادوی فکر - (مجری پروژه نرم افزاری حوزه هوش مصنوعی و بلاکچین)

www.jadoosoft.com