برای خرید، فروش و پخش محصولات مختلف به کار می‌رود. برخی از مراحل بیان شده طراحی پایگاه داده در درس در زیر آمده است:

۱. تعیین جداول مورد نیاز، این بانک اطلاعاتی اطلاعات مشتریان، حساب‌ها و غیره را نگهداری می‌کند. جداول در نظر گرفته شده برای این بانک در زیر آمده‌اند:

- جدول **Customer**، اطلاعات مشتریان از قبیل کد مشتری، نام مشتری، نام خانوادگی مشتری، کد ملی و غیره را نگهداری می‌کند (جدول ۳ - ۱).
- جدول **City**، اطلاعات شهرها از قبیل کد شهر، نام شهر و کد کاربر را نگهداری می‌نماید (جدول ۳-۱).

جدول ۳ - ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات

نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
Customer (مشتریان)	ID	کد مشتری	Varchar(6)	بله	نه	نه
	firstName	نام مشتری	Varchar(30)	نه	نه	نه
	lastName	نام خانوادگی مشتری	Varchar(30)	نه	نه	نه
	nationalCode	کد ملی	Varchar(10)	نه	نه	نه
	Tel	شماره تلفن	Varchar(15)	نه	بله	بله
	Mobile	شماره موبایل	Varchar(15)	نه	بله	نه
	Address	آدرس	Varchar(50)	نه	بله	نه
	acivityType	نوع فعالیت	Varchar(50)	نه	بله	نه
	cityID	کد شهر	Varchar(6)	نه	بله	بله
	stateID	کد استان	Varchar(6)	نه	بله	نه
	Email	ایمیل	Varchar(50)	نه	بله	نه
	goodAccount	خوش حسابی	Bit	نه	بله	نه
	Credit	میزان اعتبار	Decimal(18,0)	نه	بله	نه
	userCode	کد کاربر	Varchar(6)	نه	بله	بله
Login1 (کاربران)	userCode	کد کاربر	Varchar(6)	بله	نه	نه
	usernName	نام کاربر	Varchar(50)	نه	نه	نه
	password	کلمه عبور	Varchar(50)	نه	بله	نه
City (شهرها)	cityID	کد شهر	Varchar(6)	بله	نه	نه
	cityName	نام شهر	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله

ادامه جدول ۳ - ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات

نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
State (استان‌ها)	stateID	کد استان	Varchar(6)	بله	نه	نه
	stateName	نام استان	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Store (انبار)	storeID	کد انبار	Varchar(6)	بله	نه	نه
	storeName	نام انبار	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	نه
Merchandise (کالا)	merchandiseID	کد کالا	Varchar(6)	بله	نه	نه
	merchandiseGroupID	کد گروه کالا	Varchar(6)	نه	نه	بله
	merchandiseName	نام کالا	Varchar(6)	نه	نه	نه
	consumerPrice	قیمت مصرف کننده	Numeric(18, 0)	نه	بله	نه
	buyPrice	قیمت خرید	Numeric(18, 0)	نه	بله	نه
	poketSellingPrice	قیمت فروش بسته‌ای	Numeric(18, 0)	نه	بله	نه
	changueSelling Price	قیمت تعویض	Numeric(18, 0)	نه	بله	نه
	hav	موجودی	Numeric(18, 0)	نه	بله	نه
	peritionAmount	حد سفارش	Numeric(18, 0)	نه	بله	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
merchandiseGroup (گروه کالا)	merchandiseGroupID	کد گروه کالا	Varchar(6)	بله	نه	نه
	merchandiseGroupName	نام گروه کالا	Varchar(50)	نه	نه	نه
	unit	واحد	Varchar(50)	نه	بله	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
BankSave (بانک)	fishNo	شماره فیش	Varchar(15)	بله	نه	نه
	hesabNo	شماره حساب	Varchar(15)	نه	نه	نه
	Date	تاریخ	Varchar(8)	نه	نه	نه
	mablagTake	مبلغ برداشتی	Decimal(18, 0)	نه	نه	نه
	mablagSave	مبلغ واریزی	Decimal(18, 0)	نه	نه	نه
	sellerID	کد فروشنده	Varchar(6)	نه	نه	بله
	comment	توضیحات	Varchar(6)	نه	بله	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
BuyInvoice (فاکتور خرید)	invoiceNumeral	شماره فاکتور	Decimal(18, 0)	بله	نه	نه
	Date	تاریخ فاکتور	Varchar(8)	نه	نه	نه
	ID	شماره مشتری	Varchar(6)	نه	نه	بله

ادامه جدول ۳ - ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات

نام جدول	نام فیلد	هدف	نوع	کلید اصلی	تهی	کلید خارجی
BuyInvoice (فاکتور خرید)	buyType	نوع خرید	Varchar(50)	نه	بله	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
buyInvoice1 (ریز فاکتور خرید)	invoiceNumeral	شماره فاکتور	Decimal(18, 0)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(6)	بله	نه	بله
	Price	قیمت	Numeric(18, 0)	نه	نه	نه
	Discount	تخفیف	Numeric(18, 0)	نه	نه	نه
	Amount	تعداد	Numeric(18, 0)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
sellInvoice (فاکتور فروش)	invoiceNumeral	شماره فاکتور	Decimal(18, 0)	بله	نه	نه
	Date	تاریخ فاکتور	Varchar(8)	نه	نه	نه
	ID	شماره مشتری	Varchar(6)	نه	نه	بله
	sellType	نوع فروش	Varchar(50)	نه	بله	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
sellInvoice1 (ریز فاکتور فروش)	invoiceNumeral	شماره فاکتور	Decimal(18, 0)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(6)	بله	نه	بله
	Price	قیمت	Numeric(18, 0)	نه	نه	نه
	Discount	تخفیف	Numeric(18, 0)	نه	نه	نه
	Amount	تعداد	Numeric(18, 0)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Hessab (حساب بانکی)	hessabNo	شماره حساب	Varchar(15)	بله	نه	نه
	hessabName	نام حساب	Varchar(50)	نه	نه	نه
	bankName	نام بانک	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Seller (فروشندگان)	sellerID	شماره فروشنده	Varchar(6)	بله	نه	نه
	sellerName	نام فروشنده	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Recipt (رسید)	recepitID	شماره رسید	Varchar(6)	بله	نه	نه
	storeID	کد انبار	Varchar(6)	نه	نه	بله
	Date	تاریخ	Varchar(8)	نه	نه	نه
	InvoiceNumeral	شماره فاکتور	Decimal(18, 0)	نه	نه	بله
	Amount	تعداد	Numeric(18, 0)	نه	نه	نه

ادامه جدول ۳ - ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات

نام جدول	نام فیلد	هدف	نوع	کلید اصلی	تهی	کلید خارجی
Recipt (رسید)	Render	تحویل گیرنده	Varchar(50)	نه	نه	نه
	Teller	تحویل دهنده	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Receipt2 (ریز کالا رسید شده)	receiptID	شماره رسید	Varchar(6)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(6)	بله	نه	بله
	Amount	تعداد کالا	Numeric(18,0)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Order (حواله)	receiptID	شماره حواله	Varchar(6)	بله	نه	نه
	storeID	کد انبار	Varchar(6)	نه	نه	بله
	Date	تاریخ	Varchar(8)	نه	نه	نه
	invoiceNumeral	شماره فاکتور	Decimal(18,0)	نه	نه	بله
	Amount	تعداد	Numeric(18,0)	نه	نه	نه
	Render	تحویل گیرنده	Varchar(50)	نه	نه	نه
	Teller	تحویل دهنده	Varchar(50)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله
Order2 (ریز کالای حواله شده)	receiptID	شماره حواله	Varchar(6)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(6)	بله	نه	بله
	Amount	تعداد کالا	Numeric(18,0)	نه	نه	نه
	userCode	کد کاربر	Varchar(6)	نه	نه	بله

- **جدول Login**، اطلاعات کاربران از قبیل کد کاربر، نام کاربر و کلمه عبور را نگهداری می‌ک (جدول ۳-۱).
- **جدول State**، اطلاعات استان‌ها از قبیل کد استان، نام استان و کد کاربر را نگهداری می‌کند (ج ۳ - ۱).
- **جدول Store**، اطلاعات انبار از قبیل کد انبار، نام انبار و کد کاربر را نگهداری می‌کند (جدول۳ - ۱
- **جدول Merchandise**، اطلاعات کالا از قبیل کدکالا، نام کالا، کد گروه کالا، قیمت مصرف ک قیمت خرید و غیره را نگهداری می‌کند (جدول ۳ - ۱).

- **جدول BankSave**، اطلاعات برداشت و واریزها به بانک از قبیل شماره فیش، شماره حساب، تاریخ، مبلغ برداشت، مبلغ واریز و غیره را نگهداری می‌نماید (جدول ۳ – ۱).
- **جدول buyInvoice**، اطلاعات فاکتور از قبیل شماره فاکتور، تاریخ فاکتور، شماره مشتری، نوع خرید و کد کاربر را ذخیره می‌کند (جدول۳ – ۱).
- **جدول buyInvoice1**، اطلاعات ریز فاکتورها از قبیل شماره فاکتور، کد کالا، قیمت، تخفیف، تعداد و کد کاربر را ذخیره می‌کند (جدول ۳ – ۱).
- **جدول SellInvoice**، اطلاعات فاکتور فروش از قبیل شماره فاکتور، تاریخ، کد مشتری و غیره را نگهداری می‌کند (جدول ۳ – ۱).
- **جدول SellInvoice1**، اطلاعات ریز فاکتور فروش از قبیل شماره فاکتور، کد کالا، قیمت، تخفیف، تعداد و کد کاربر را نگهداری می‌نماید (جدول ۳ – ۱).
- **جدول Hessab**، اطلاعات حساب‌ها از قبیل شماره حساب، نام بانک، نام حساب و کد کاربر را ذخیره می‌نماید (جدول ۳ – ۱).
- **جدول Seller**، اطلاعات فروشندگان از قبیل شماره فروشنده، نام فروشنده و کد کاربر را ذخیره می‌نماید (جدول ۳ – ۱).
- **جدول Receipt**، اطلاعات رسیدها از قبیل شماره سند، کد انبار، تاریخ، شماره فاکتور و غیره را ذخیره می‌نماید (جدول ۳ – ۱).
- **جدول Receipt2**، اطلاعات ریز رسیدها از قبیل شماره رسید، کد کالا، تعداد و کد کاربر را ذخیره می‌کند (جدول ۳ – ۱).
- **جدول Order**، اطلاعات حواله‌ها از قبیل شماره حواله، کد انبار، تاریخ و غیره را نگهداری می‌کند (جدول ۳ – ۱).
- **جدول Order2**، اطلاعات ریز حواله‌ها از قبیل شماره حواله، کد کالا، تعداد و کد کاربر را نگهداری می‌کند (جدول ۳ – ۱).

۲. تعیین فیلدهای کلید اولیه

همان‌طور که بیان گردید، در هر جدول باید یک یا چند فیلد آن به عنوان کلید اولیه باشد. کلیدهای اولیه بانک اطلاعاتی خرید، فروش و پخش محصولات در زیر آمده است:

- **در جدول Customer**، فیلد ID کلید اولیه است. زیرا، هیچ دو مشتری نمی‌توانند شماره مشتری یکسان داشته باشند.
- **در جدول Login**، فیلد userCode کلید اولیه است. چون، هیچ دو کاربری کد یکسان ندارند.
- **در جدول City**، فیلد cityID کلید اولیه است. چون هیچ دو شهری کد یکسان ندارند.
- **در جدول State**، فیلد stateID کلید اولیه است. زیرا، کد استان نمی‌تواند تکراری باشد.

- در جدول **Store**، فیلد storeID کلید اولیه است. چون، کد انبار یکتا است.
- در جدول **Merchandise**، فیلد merchandiseID کلید اولیه است. چون، هیچ دو کالایی نمی‌توانند کد یکسان داشته باشند.
- در جدول **MerchandiseGroup**، فیلد merchandiseGroupID کلید اولیه است. زیرا، کد گروه کالا نمی‌تواند تکراری باشد.
- در جدول **BankSave**، فیلد fishNo کلید اولیه است. زیرا، هیچ دو فیشی نمی‌توانند شماره تکراری داشته باشند.
- در جداول **buyInvoice** و **SellInvoice**، فیلد invoiceNumeral کلید اولیه است. زیرا، هیچ دو فاکتور خرید و فروشی نباید شماره تکراری داشته باشند.
- در جداول **buyInvoice1** و **SellInvoice1**، ترکیب فیلدهای invoiceNumeral، merchandiseID کلید اولیه است.
- در جدول **Hesab**، فیلد hesabNo کلید اولیه است، چون هیچ دو حسابی نمی‌توانند شماره تکراری داشته باشند.
- در جدول **Seller**، فیلد sellerID کلید اولیه است. چون کد هیچ دو فروشنده نمی‌تواند یکی باشد.
- در جدول **Receipt** و **Order**، به ترتیب receiptID و orderID کلید اولیه هستند.
- در جدول **Receipt2**، ترکیب فیلدهای receiptID و merchandiseID کلید اولیه است.
- در جدول **Order2**، ترکیب فیلدهای orderID و merchandiseID کلید اولیه است.

۳. تعیین ارتباط بین جداول

یکی از بخش‌های بسیار مهم بانک اطلاعاتی، تعیین فیلد ارتباطی بین جداول و نوع ارتباط است. ارتباط بین جداول در بانک اطلاعاتی خرید، فروش و پخش محصولات در زیر آمده است:

- بین جداول Customer و Login1، فیلد ارتباط userCode است. این فیلد در جدول Customer کلید خارجی، ولی در جدول Login کلید اولیه است.
- بین جدول Customer و City، فیلد ارتباط cityID است. این فیلد در جدول Customer کلید خارجی است.
- بین جدول Customer و State، فیلد ارتباط stateID است که در جدول Customer کلید خارجی است. چون در جدول State کلید اولیه می‌باشد.
- بین جدول Login1 و تمام جدول، userCode ارتباط را برقرار می‌کند که در جدول Login1، فیلد userCode کلید اولیه است، اما، در بقیه جداول کلید خارجی می‌باشد.
- بین جداول Store، جداول Receipt و Order فیلد ارتباط storeID است. این فیلد در جدول Store کلید اولیه است. اما در جدول Receipt و Order کلید خارجی می‌باشد.

- بین جدول Merchandise و جداول Receipt1، Order1، buyInvoice1 و SellInvoice1 فیلد merchandiseID ارتباط را برقرار می‌کند. این فیلد در جدول Merchandise کلید اولیه است. ولی، در بقیه جداول کلید خارجی می‌باشد.
- بین جداول Merchandise و MerchandiseGroup فیلد merchandiseGroupID ارتباط را برقرار می‌کند. این فیلد در جدول Merchandise کلید خارجی است.
- بین جداول BankSave و Hesab فیلد hesabNo ارتباط را برقرار می‌کند. فیلد hesabNo در جدول BankSave کلید خارجی است.
- بین جدول Customer و جداول buyInvoice1 و SellInvoice1 فیلد ID ارتباط را برقرار می‌کند. این فیلد در جدول Customer کلید اولیه است.

مثال ۱ – ۱. پرس‌وجویی که بانک اطلاعاتی Buy_Sell را ایجاد می‌کند.

```
CREATE DATABASE Buy_Sell
ON PRIMARY
(NAME=buy_sell_data,FILENAME='C:\classDatabase\buy_sell_data.mdf
',SIZE = 5MB, MAXSIZE = 10MB,FILEGROWTH = 5%)
LOG ON
(NAME=buy_sell_log,FILENAME='C:\classDatabase\buy_sell_log.ldf',
  SIZE = 4MB, MAXSIZE = 12MB,FILEGROWTH = 2MB)
  COLLATE Arabic_CS_AS_KS_WS
GO
```

فایل‌های زیر به این بانک اضافه می‌شود:

- **فایل داده buy_sell_data.mdf**، در پوشه classDatabase درایو C ایجاد می‌شود که دارای اندازه 5MB است و حداکثر تا 10MB رشد می‌کند (درصد رشد آن ۵ درصد است).
- **فایل کارنامه buy_sell_log.ldf**، در پوشه classDatabase درایو C ایجاد می‌گردد. این فایل اندازه 4MB را دارد که حداکثر تا 12MB می‌تواند رشد کند (مقدار رشد این فایل 2MB در هر مرتبه است).
- دقت کنید که برای COLLATE مقدار Arabic_CS_AS_KS_WS انتخاب گردید تا بتوان داده‌های فارسی را در جداول بانک ذخیره نمود.

مثال ۲ – ۱. پرس‌وجویی که جدول Login1 را در بانک اطلاعاتی Buy_Sell ایجاد می‌کند.

```
USE Buy_Sell
GO
CREATE TABLE Login1
 ( userCode varchar(6) PRIMARY KEY,
   userName varchar(50) NOT NULL,
   Password varchar(50) NOT NULL )
 GO
```

در این جدول، فیلد userCode به عنوان کلید اولیه تعریف شده است. فیلد userCode در تمام جداول بانک اطلاعاتی در نظر گرفته شده است تا تعیین گردد، کدام کاربر رکوردهای هر جدول را اضافه کرده یا تغییر داده است.

مثال ۳ – ۱. پرس‌وجویی که جدول City را در بانک اطلاعاتی فعلی (Buy_Sell) ایجاد می‌کند.

```
CREATE TABLE City
```

```
( cityID varchar(6) PRIMARY KEY,
  cityName varchar(50) NOT NULL,
  userCode varchar(6) REFERENCES Login1(userCode) )
GO
```

این دستور جدول City را با کلید اولیه cityID ایجاد می‌کند. ارتباط این جدول از طریق فیلد userCode با جدول Login1 برقرار می‌گردد (فیلد userCode در این جدول کلید خارجی است).

مثال ۴ – ۱. پرس‌وجویی که جدول State را در بانک اطلاعاتی Buy_Sell ایجاد می‌کند.

```
CREATE TABLE State
( stateID varchar(6) PRIMARY KEY,
  stateName varchar(50) NOT NULL,
  userCode varchar(6) REFERENCES Login1(userCode) )
GO
```

مثال ۵ – ۱. پرس‌وجویی که جدول Customer را ایجاد می‌کند.

```
CREATE TABLE Customer
( ID varchar(6) PRIMARY KEY,
  firstName varchar(30) NOT NULL,
  lastName varchar(30) NOT NULL,
  nationalCode varchar(10) NULL,
  Tel varchar(15) NULL,
  Mobile varchar(15) NULL,
  Addres varchar(150),
  activityType varchar(50),
  cityID varchar(6) REFERENCES City(cityID),
  stateID varchar(6) REFERENCES State(stateID),
  Email varchar(40),
  goodAcount  bit,
  credit  decimal(18, 0),
  userCode varchar(6) REFERENCES Login1(userCode)
 )
GO
```

همان‌طور که می‌بینید، این جدول با جداول Login1 (از طریق فیلد userCode)، City (از طریق فیلد cityID) و State (از طریق فیلد stateID) ارتباط دارد.

مثال ۶ – ۱. پرس‌وجویی که اندازه فیلد Email از جدول Customer را به ۵۰ تغییر می‌دهد.

```
ALTER TABLE Customer
 ALTER COLUMN Email varchar(50)
GO
```

مثال ۷ – ۱. پرس‌وجویی که فیلد Picture را به جدول Customer اضافه می‌کند.

```
ALTER TABLE Customer
 ADD  Picture  varbinary(MAX)
GO
```

نکته: نوع varbinary(MAX) تصویر را به صورت بایت به بایت در جدول نگهداری می‌کند. اما، نوع Image مسیر تصویر را در جدول نگهداری می‌نماید.

مثال ۸ – ۱. پرس‌وجویی که جدول Store را ایجاد می‌کند.

```
CREATE TABLE Store
( storeID varchar(6) PRIMARY KEY,
  userCode varchar(6) REFERENCES Login1(userCode) )
GO
```

همان‌طور که در این دستور می‌بینید، فیلد storeName در جدول ایجاد نشد، پرس‌وجوی ۹ این فیلد را به جدول اضافه می‌کند.

مثال ۹ – ۱. پرس‌وجویی که فیلد storeName را به جدول Store اضافه می‌کند.

```
ALTER TABLE Store
 ADD storeName varchar(50) NOT NULL
GO
```

مثال ۱۰ – ۱. پرس‌وجویی که جدول merchandiseGroup (گروه کالا) را اضافه می‌کند.

```
CREATE TABLE merchandiseGroup
 ( merchandiseGroupID varchar(6) PRIMARY KEY,
   merchandiseGroupName varchar(50) NOT NULL,
   unit varchar(50) )
 GO
```

همان‌طور که در این دستور می‌بینید، فیلد userCode ایجاد نگردید و ارتباط آن با جدول Login1 برقرار نشد. در پرس‌وجوی ۱۱ این فیلد ایجاد شده ارتباط آن با جدول Login1 برقرار می‌گردد.

مثال ۱۱ – ۱. پرس‌وجویی که فیلد userCode را به جدول merchandiseGroup اضافه کرده و ارتباط بین این جدول و جدول Login1 را از طریق آن برقرار می‌کند.

```
ALTER TABLE merchandiseGroup
 ADD  userCode varchar(6) REFERENCES Login1(userCode)
GO
```

مثال ۱۲ – ۱. پرس‌وجویی که جدول Merchandise (کالا) را ایجاد می‌کند.

```
CREATE TABLE  Merchandise
 ( merchandiseID varchar(6) PRIMARY KEY,
   merchandiseGroupID varchar(6)REFERENCES  merchandiseGroup(
   merchandiseGroupID), merchandiseName varchar(50)NOT NULL,
   consumerPrice  decimal(18,0),
   buyPrice  decimal(18,0),
   poketSellingPrice  decimal(18,0),
   chequeSellingPrice  decimal(18,0),
   Hav decimal(18,0) ,
   userCode varchar(6) REFERENCES Login1(userCode)  )
 GO
```

مثال ۱۳ – ۱. پرس‌وجویی که جدول Seller (فروشنده) را ایجاد می‌کند.

```
CREATE TABLE Seller
 ( sellerID varchar(6) PRIMARY KEY,
   sellerName varchar(50) NOT NULL,
   userCode varchar(6) REFERENCES Login1(userCode) )
 GO
```

مثال ۱۴ – ۱. پرس‌وجویی که جدول BuyInvoice (فاکتور خرید) را ایجاد می‌کند.

```
CREATE TABLE BuyInvoice
 ( invoiceNumeral decimal(18, 0) PRIMARY KEY,
   date varchar(8) NOT NULL,
   ID varchar(6) REFERENCES Customer(ID),
   buyType  varchar(50),
   userCode varchar(6) REFERENCES Login1(userCode) )
 GO
```

مثال ۱۵ – ۱. پرس‌وجویی که جدول BuyInvoice1 (ریز اطلاعات فاکتور) را ایجاد می‌کند.

```
CREATE TABLE BuyInvoice1
 ( invoiceNumeral decimal(18, 0),
   merchandiseID varchar(6),
   ID varchar(6) REFERENCES Customer(ID),
   Price  decimal(18, 0),
   Discount  decimal(18, 0),
   Amount decimal(18, 0),
   userCode varchar(6) REFERENCES Login1(userCode),
   PRIMARY KEY(invoiceNumeral,merchandiseID),
   CONSTRAINT FK_I_ID FOREIGN KEY(invoiceNumeral) REFERENCES
BuyInvoice(invoiceNumeral),
   CONSTRAINT FK_M_ID FOREIGN KEY( merchandiseID) REFERENCES
Merchandise( merchandiseID)
  )
 GO
```

همان‌طور که در این دستور می‌بینید، کلید این جدول ترکیب فیلدهای invoiceNumeral و merchandiseID است. ترکیب این فیلدها با واژه PRIMARY KEY به عنوان کلید اولیه تعریف گردیدند.

در این دستور دو محدودیت زیر تعریف شده است:

- **محدودیت FK_I_ID**، برای تعریف کلید خارجی که ارتباط بین جداول BuyInvoice1 و BuyInvoice را از طریق فیلد invoiceNumeral برقرار می‌کند، به کار می‌رود.
- **محدودیت FK_M_ID**، برای تعریف کلید خارجی که ارتباط بین جداول BuyInvoice1 و Merchandise را از طریق فیلد merchandiseID برقرار می‌کند، به کار می‌رود.

مثال ۱۶ – ۱. پرس‌وجویی که فیلد Picture را از جدول Customer حذف می‌کند.

```
ALTER TABLE Customer
  DROP COLUMN Picture
GO
```

برای تمرین بیشتر بقیه جداول بانک اطلاعاتی Buy_Sell (جداول موجود در جدول ۳– ۱ که ایجاد نشده‌اند) را ایجاد کنید.

۶ – ۱. دستور کار آزمایشگاه

قبل این که دستور کار آزمایشگاه را شروع کنیم، بانک اطلاعاتی آژانس حمل و نقل را در نظر بگیرید.

در این بانک اطلاعاتی، مشتریان (Customers) زنگ می‌زنند، سفارشی (انتقال وسائل یا خودشان از یک مکان به مکان دیگر) را می‌دهند. سپس خودرو یا خودروهایی که هر یک از آن خودروها یک راننده دارند، این سفارشات را انجام می‌دهند. جداول و فیلدهای این بانک اطلاعاتی در جدول ۴ – ۱ آمده است. اکنون به سوالات زیر پاسخ دهید.

۱. ارتباط بین جداول رانندگان (Drivers) و جدول ماشین‌ها (Cars) از چه نوعی است؟

۲. ارتباط بین مشتریان (Customers) و سفارشات (Orders) از چه نوعی است؟

۳. ارتباط بین جداول رانندگان (Drivers) و سفارشات (Orders) از چه نوعی است؟

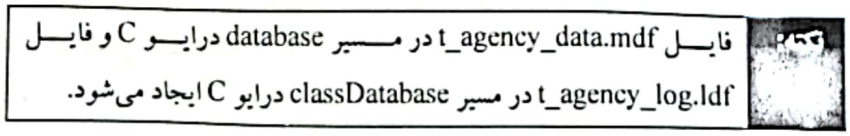
۴. ارتباط بین جداول Orders و Cars از چه نوعی است؟

جدول ۴ - ۱ بانک اطلاعاتی آژانس حمل و نقل

نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
Cars (ماشین‌ها)	carID	شماره ماشین	Varchar(10)	بله	نه	نه
	carType	نوع ماشین	Varchar(30)	نه	نه	نه
	capicity	ظرفیت	Decimal(18, 0)	نه	نه	نه
	unit	واحد	Varchar(30)	نه	نه	نه
	drID	شماره راننده	Varchar(10)	نه	نه	بله
Drivers (رانندگان)	drID	شماره راننده	Varchar(10)	بله	نه	نه
	drFname	نام راننده	Varchar(20)	نه	نه	نه
	drLname	نام خانوادگی راننده	Varchar(20)	نه	نه	نه
	degree	درجه گواهی نامه	Varchar(20)	نه	بله	نه
Customers (مشتریان)	cID	شماره مشتری	Varchar(10)	بله	نه	نه
	cFname	نام مشتری	Varchar(20)	نه	نه	نه
	cLname	نام خانوادگی مشتری	Varchar(20)	نه	نه	نه
	Tel	شماره تلفن	Varchar(15)	نه	بله	نه
	City	شهر مشتری	Varchar(30)	نه	بله	نه
Orders (سفارشات)	oID	شماره سفارش	Varchar(10)	بله	نه	نه
	drID	شماره راننده	Varchar(10)	نه	بله	بله
	cID	شماره مشتری	Varchar(10)	نه	بله	بله
	desCity	شهر مقصد	Varchar(30)	نه	نه	نه
	Freight	کرایه	Decimal(18, 0)	نه	نه	نه

۵. بانک اطلاعاتی آژانس حمل و نقل را با نام TransportationAgency ایجاد کنید. این بانک دارای فایل‌های زیر باشد:

- فایل **t_agency_data.mdf**، دارای حجم سه مگابایت و حداکثر تا ۱۰ مگابایت رشد می‌کند که درصد رشد آن ۲۰ است.
- فایل **t_agency_log.ldf**، دارای حجم سه مگابایت و حداکثر تا ۱۰ مگابایت رشد می‌کند که درصد رشد آن ۵۰ است.



فایل t_agency_data.mdf در مسیر database درایو C و فایل t_agency_log.ldf در مسیر classDatabase درایو C ایجاد می‌شود.

۶. فایل کارنامه t_agency1_log.ldf را با اندازه 5MB به بانک اطلاعاتی TransportationAgency در پوشه Database درایو C اضافه کنید، به طوری که حداکثر تا 100MB بتواند رشد داشته باشد و رشد فایل 5MB در هر مرتبه باشد.

۷. فایل داده t_agency1_data.mdf را با اندازه 5MB به بانک اطلاعاتی TransportationAgency در پوشه classDatabase درایو C اضافه کنید، به طوری که حداکثر اندازه آن 50MB شود و رشد آن 5MB در هر مرحله باشد.

۸. فایل t_agency1_data.mdf را از بانک اطلاعاتی TransportationAgency حذف کنید.

۹. جدول Drivers را ایجاد کنید.

۱۰. فیلدهای Email (از نوع (varchar(20) و Photo (از نوع image) را به جدول Drivers اضافه کنید.

۱۱. فیلد Email را از جدول Drivers حذف کنید.

۱۲. جدول Cars را ایجاد کنید.

۱۳. ارتباط بین جداول Drivers و Cars را ایجاد کنید.

۱۴. فیلد Photo را از جدول Drivers حذف کنید.

۱۵. جدول Customers را ایجاد کنید.

۱۶. فایل t_agency1_log.ldf را از بانک اطلاعاتی TransportationAgency حذف کنید.

۱۷. جدول Orders را ایجاد کنید.

۱۸. ارتباط بین Orders با جداول Drivers و Customers را برقرار کنید.

۱۹. در بانک اطلاعاتی که اطلاعات را به زبان فارسی ذخیره می‌کند، برای ذخیره اطلاعات فیلدهای زیر از چه انواعی استفاده می‌شوند:

الف: شماره دانشجویی ب: نمره ج: مدرک استاد د: کد ملی

پ: عکس دانشجو ت: اطلاعات XML خ: تاریخ فارغ‌التحصیلی چ: قیمت کالا

۲۰. جدولی به نام Test با فیلدهای x1 و x2 از نوع int به بانک TransportationAgency اضافه کنید.

۲۱. فیلد x2 را در جدول Test کلید اولیه تعیین کنید.

۲۲. جدول Test را حذف کنید.

۷ – ۱. جواب دستور کار آزمایشگاه

۱. ارتباط بین Cars و Drivers یک به یک است. چون، هر ماشین فقط یک راننده دارد و هر راننده می‌تواند فقط یک ماشین داشته باشد (البته در این سیستم). برای برقراری ارتباط بین این دو جدول فیلد drID (شماره راننده) به کار می‌رود. این فیلد در جدول Cars کلید خارجی است، چون در جدول Drivers کلید اولیه (اصلی) می‌باشد.

۲. ارتباط بین Customers و Orders چند به یک است. زیرا، هر مشتری می‌تواند چند سفارش داشته باشد. فیلد ارتباط بین این دو جدول clD است که در جدول Orders کلید خارجی است. چون در جدول Customers کلید اصلی می‌باشد.

۳. ارتباط بین جداول Drivers و Orders چند بر چند است. زیرا، یک راننده می‌تواند چند سفارش را انجام دهد و چند راننده نیز می‌توانند یک سفارش را اجرا کنند (فیلد ارتباط drID است که در جدول Orders کلید خارجی است).

۴. ارتباط بین جداول Cars و Orders به طور مستقیم وجود ندارد. اما، از طریق جدول Drivers می‌توان ارتباط بین جداول Cars و Orders را برقرار گردد.

۵. ایجاد بانک اطلاعاتی TransportationAgency با اجرای دستور زیر انجام می‌شود:

```
CREATE DATABASE TransportationAgency
ON(NAME=t_agency_data,FILENAME='C:\Database\t_agency_data.mdf',
     SIZE = 3, MAXSIZE = 10MB, FILEGROWTH = 20%)
LOG ON( NAME = t_agency_log, FILENAME = 'C:\classDatabase
\t_agency_log.ldf',
     SIZE = 3, MAXSIZE = 10MB, FILEGROWTH = 50%)
 COLLATE Arabic_CS_AS_KS_WS
GO
```

همان‌طور که در این دستور می‌بینید، مقدار Arabic_CS_AS_KS_WS برای پارامتر COLLATE انتخاب گردید تا بتوان مقادیر فارسی را در جداول آن ذخیره نمود.

۶. برای اضافه کردن فایل کارنامه t_agency1_log.ldf دستور زیر را اجرا کنید:

```
ALTER DATABASE TransportationAgency
ADD LOG FILE( NAME = t_agency1_log,FILENAME = 'c:\Database\
t_agency1_log.ldf',SIZE = 5MB,MAXSIZE=100MB,FILEGROWTH = 5MB)
GO
```

۷. برای اضافه کردن فایل t_agency_data.mdf دستور زیر را اجرا کنید:

```
ALTER DATABASE TransportationAgency
ADD FILE( NAME = t_agency1_data,FILENAME = 'c:\classDatabase
\t_agency1_data.mdf',SIZE = 5MB,MAXSIZE=50MB,FILEGROWTH=5MB)
GO
```

۸. برای حذف فایل t_agency1_data.mdf از بانک دستور زیر را اجرا کنید:

```
ALTER DATABASE TransportationAgency
 REMOVE FILE t_agency1_data
```

۹. برای ایجاد جدول Drivers دستور زیر را اجرا کنید:

```
USE TransportationAgency
GO
CREATE TABLE Drivers
(
  drID    varchar(10) PRIMARY KEY,
  drFname    varchar(20) NOT NULL,
  drLname    varchar(20) NOT NULL,
  dgeree    varchar(20)  NULL
 )
 GO
```

۱۰. برای افزودن فیلدهای Email و Photo دستورات زیر را اجرا کنید:

```
ALTER TABLE Drivers
 ADD Email varchar(20), Photo image
```

```
GO
```

۱۱. برای حذف فیلد Email از جدول Drivers، دستورات زیر را اجرا نمایید:

```
ALTER TABLE Drivers
 DROP COLUMN Email
GO
```

۱۲. برای ایجاد جدول Cars دستورات زیر را انجام دهید:

```
CREATE TABLE Cars
(
  carID    varchar(10) PRIMARY KEY,
  carType  varchar(30) NOT NULL,
  capicity decimal(18, 0),
  unit     varchar(30),
  drID     varchar(10)
)
GO
```

۱۳. برای ایجاد ارتباط بین جداول Cars و Drivers محدودیتی به نام FK_CAR_DR (جهت تعریف کلید خارجی) با دستور زیر ایجاد کنید:

```
ALTER TABLE Cars
 ADD CONSTRAINT FK_CAR_DR FOREIGN KEY (drID) REFERENCES
      Drivers(drID)
 GO
```

۱۴. برای حذف فیلد Photo از جدول Drivers، دستورات زیر را اجرا کنید:

```
ALTER TABLE Drivers
 DROP COLUMN Photo
GO
```

۱۵. برای ایجاد جدول Customers، دستورات زیر را تایپ کنید:

```
CREATE TABLE Customers
(
  cID    varchar(10) PRIMARY KEY,
  cFname    varchar(20) NOT NULL,
  cLname    varchar(20) NOT NULL,
  Tel    varchar(15)  NULL,
  City    varchar(30)  NULL
)
GO
```

۱۶. برای حذف فایل t_agency1_log.ldf از بانک اطلاعاتی TransportationAgency، دستورات زیر را اجرا کنید:

```
ALTER DATABASE TransportationAgency
 REMOVE FILE t_agency1_log
```

۱۷. برای ایجاد جدول Orders، دستورات زیر را اجرا نمایید:

```
CREATE TABLE Orders
(
  oID      varchar(10) PRIMARY KEY,
  drID     varchar(10),
  cID      varchar(10),
  desCity varchar(30) NOT NULL,
  Freight decimal(18, 0)
)
```

```
GO
```

۱۸. برای ایجاد ارتباط بین جدول Orders با جداول Drivers و Customers دستورات زیر را تایپ کنید:

```
ALTER TABLE Orders
 ADD CONSTRAINT FK_ORDER_DR FOREIGN KEY (cID) REFERENCES
      Customers(cID),
 CONSTRAINT FK_ORDER_CST FOREIGN KEY (drID) REFERENCES
       Drivers(drID)
GO
```

۱۹. فیلدهایی که محاسبات روی آن‌ها انجام نمی‌شود، بهتر است از نوع varchar در نظر گرفته شوند. بنابراین فیلدهای شماره دانشجوئی، مدرک استاد و کد ملی را از نوع varchar در نظر می‌گیریم. اما فیلدهایی که محاسبات بر روی آن‌ها انجام می‌شود، بهتر است عددی (real، int، Decimal و ...) در نظر گرفته شوند، بنابراین فیلدهای نمره و قیمت کالا با یکی از انواع تعیین شده در نظر گرفته می‌شوند. فیلد عکس دانشجو باید یکی از انواع varbinary(MAX) یا Image در نظر گرفته شود. اگر این فیلد را به صورت Image در نظر بگیرید، مسیر تصویر در جدول ذخیره خواهد شد. اما، اگر این فیلد را از نوع varbinary(MAX) در نظر بگیرید، فایل تصویر به صورت بایت به بایت در جدول ذخیره خواهد شد. اطلاعات Xml را از نوع xml در نظر می‌گیریم. فیلدهای تاریخ فارسی با فرمت صورت varchar در نظر گرفته می‌شود.

۲۰. دستور زیر جدول Test را اضافه می‌کند:

```
CREATE TABLE Test (x1 int, x2 int)
```

۲۱. دستورات زیر را اجرا کنید تا x2 به عنوان کلید اولیه جدول Test تعریف شود:

```
ALTER TABLE Test
 ALTER COLUMN x2 int NOT NULL
 GO
ALTER  TABLE Test
 ADD CONSTRAINT PK_X2 PRIMARY KEY(x2)
```

همان‌طور که در این دستورات می‌بینید، ابتدا با دستور ALTER محدودیت NOT NULL را برای فیلد x2 تعریف کردیم. زیرا، فیلدهایی که محدودیت NULL داشته باشند، نمی‌توانند به عنوان کلید اولیه تعریف شوند. سپس، با دستور ALTER TABLE، فیلد x2 را به عنوان کلید اولیه معرفی نمودیم.

۲۲. برای حذف جدول Test دستور DROP TABLE به صورت زیر به کار می‌رود:

```
DROP TABLE Test
```

۸ - ۱. تمرین‌ها

جداول موجود در جدول ۵ - ۱ را در نظر بگیرید. اکنون به سوالات زیر پاسخ دهید:

۱. بانک اطلاعاتی به نام Accounting ایجاد کنید که دارای یک فایل داده و یک فایل کارنامه باشد. مسیر فایل داده و کارنامه تفاوت داشته باشد.

۲. یک فایل داده جدید به بانک Accounting اضافه کنید.

۳. کلیدهای خارجی جداول موجود در جدول ۵ - ۱ را تعیین کنید.

۴. ارتباط بین جداول را تعیین کرده و نوع ارتباط را مشخص نمایید.

۵. جداول را به همراه ارتباط بین آن‌ها در بانک اطلاعاتی Accounting ایجاد کنید.

جدول ۵ـ۲ توزیع فیلدهای جداول سیستم حسابداری

نام جدول	نام فیلد	نوع فیلد	اندازه	کلید اولیه
جدول کل	کد کل	varchar	3	بله
	نام کل	varchar	50	نه
	کد نوع حساب	varchar	1	نه
جدول معین	کد کل	varchar	3	بله
	کد معین	varchar	3	بله
	نام معین	varchar	50	نه
جدول تفضیلی	کد کل	varchar	3	بله
	کد معین	varchar	3	بله
	کد تفضیلی	varchar	6	بله
	نام تفضیلی	varchar	50	نه
سربرگ سند	شماره سند	decimal	5	بله
	تاریخ سند	varchar	10	نه
	شرح سند	varchar	255	نه
	تعداد ضمایم سند	decimal	3	نه
	کد نوع سند	varchar	1	نه
جدول ریز سند	شماره سند	decimal	5	بله
	کد کل	varchar	3	بله
	کد معین	varchar	3	بله
	کد تفضیلی	varchar	6	بله
	شرح ردیف سند	varchar	255	نه
	مبلغ بدهکار	decimal	18	نه
	مبلغ بستانکار	decimal	18	نه
	کد هزینه	varchar	6	نه
	کد پروژه	varchar	6	نه
جدول هزینه	کد هزینه	varchar	6	بله
	نام هزینه	varchar	50	نه
جدول پروژه	کد پروژه	varchar	6	بله
	نام پروژه	varchar	50	نه
جدول نوع حساب	کد نوع حساب	varchar	1	بله
	نام نوع حساب	varchar	50	نه
جدول نوع سند	کد نوع سند	varchar	1	بله
	نام نوع سند	varchar	50	نه
جدول گروه حساب	کد گروه حساب	varchar	2	بله
	نام گروه حساب	varchar	50	نه

فصل ۲

ورود، ویرایش، حذف و بازیابی اطلاعات

در فصل اول، با چند مثال ساده روش ایجاد بانک اطلاعات و جداول آن را با استفاده از دستور SQL دیدید. همان طور که بیان گردید، بعد از ایجاد جداول در بانک اطلاعات باید بتوان داده‌ها را در آن وارد نمود، آن‌ها را ویرایش کرد، یا داده‌های اضافی را حذف نمود و در صورت نیاز به اطلاعات خاصی، آن‌ها را بازیابی کرد. بنابراین، در این فصل دستورات INSERT، UPDATE، DELETE و SELECT که برای دست‌کاری داده‌های جدول به کار می‌روند را می‌آموزیم.

قبل از این که به ورود، ویرایش و بازیابی اطلاعات بپردازیم، به جداول بانک اطلاعات PublishDB اطلاعات اولیه می‌دهیم تا خروجی‌هایی که از دستورات گرفته می‌شود، همان خروجی باشد که شما با اجرای دستورات می‌گیرید. بنابراین، ابتدا اطلاعات جداول بانک اطلاعاتی PublishDB را با توجه به شکل ۱-۲ در نظر بگیرید. این اطلاعات را در ادامه وارد جداول خواهیم کرد. سپس بازیابی اطلاعات را از این جداول بیان می‌کنیم.

اطلاعات جدول Publishers

countBookPrint	bLname	bFname	cityName	URL	Tel	pName	publD
15	عباس‌نژاد	رمضان	بابل	www.fanavarienovin.net	۰۱۱۱۲۲۵۶۶۸۷	فناوری نوین	01
100	حاتمی	محمد رضا	تهران	www.danshiran.ir	۰۲۱۶۶۴۰۰۲۲۰	دانش نگار	02
	امینی	رضا	بابلسر			دانشگاه مازندران	03

اطلاعات جدول Authors

SumPayment	Mobile	Email	Ranking	Age	atLname	atFname	atlD
28000000	09111116212	fanavarienovin@yahoo.com	فوق لیسانس	42	عباس‌نژاد	رمضان	01
	09112185426	yousef_dssht@yahoo.com	لیسانس	24	عباس‌نژاد	یوسف	02
10000000			فوق لیسانس	32	رحیم‌پور	باقر	03
			فوق لیسانس	27	پویان	مرتضی	04
6000000			لیسانس	35	جلیلی	سیدحجت اله	05

اطلاعات کتاب‌ها (Books)

ISBN	Title	page	Price	editNO	printNO	groupID
۶۰۰۹۱۴۱۳۹۵	اصول طراحی پایگاه داده	۲۵۶	۷۳۰۰۰	۱	۱	۰۲
۶۰۰۹۲۲۵۴۲۲	طراحی سیستم‌های شی گرا	۴۳۲	۱۳۰۰۰۰	۱	۱	۰۱
۶۰۰۹۱۴۱۳۰۲	حل مسائل C++	۲۱۶	۵۷۰۰	۱	۲	۰۱
۶۰۰۹۱۴۱۳۴۰	حل مسائل C#	۲۰۰	۶۲۰۰۰	۱	۲	
۶۰۰۹۱۴۱۳۸۸	امنیت شبکه	۱۹۲	۶۵۰۰۰	۱	۱	۰۳
۶۰۰۹۲۲۵۴۳۹	مدیریت استراتژیک	۲۷۲	۷۵۰۰۰	۱	۱	۰۴
۷۰۰۹۲۲۵۴۳۱	رهیافت C++	۳۰۴	۴۰۰۰۰	۱	۱	۰۱

اطلاعات جدول GroupBook

groupID	groupName
۰۱	برنامه‌نویسی
۰۲	پایگاه داده
۰۳	شبکه‌های رایانه‌ای
۰۴	فناوری اطلاعات
۰۵	گرافیک رایانه‌ای

اطلاعات جدول PubBook

ISBN	PubID	Payment
6009141395	01	6000000
6009225422	01	8000000
6009141340	01	7000000
6009141388	01	6000000
6009225439	01	5000000
7009225431	01	3500000
7009225431	02	3500000
6009141302	01	6000000

اطلاعات جدول AutBook

ISBN	atID	Payment
6009141395	01	5000000
6009141395	03	5000000
6009225422	01	5000000
6009225422	03	5000000
6009141302	01	6000000
6009141340	01	6000000
6009141388	01	6000000
6009225439	05	6000000
7009225431	01	6000000

شکل ۱ – ۲ اطلاعات جدول بانک اطلاعاتی PublishDB.

۱–۲. دستور INSERT

این دستور برای اضافه کردن رکورد به جداول بانک اطلاعات به کار می‌رود و به صورت زیر استفاده می‌شود:

```
INSERT [INTO]   {table_name|view_name}
 {[(column_list)] {VALUES({DEFAULT|NULL|expression} [,...]
 |derived_table
 |execute_statement
 } }
 |DEFAULT VALUES
```

پارامترهای این دستور در زیر آمده‌اند:

- **table_name**، نام جدولی را تعیین می‌کند که اطلاعات باید وارد آن شود.

- **view_name**، نام دیدی را تعیین می‌کند که اطلاعات باید وارد آن شود.
- **column_list**، لیست فیلدهایی را تعیین می‌کند که اطلاعات بخش VALUES باید وارد آن‌ها شود.
- **DEFAULT**، مقدار پیش‌فرض فیلد جدول یا دید را مشخص می‌نماید. اگر مقدار فیلد از طریق دستور INSERT وارد نگردد، این مقدار در فیلد قرار می‌گیرد.
- **NULL**، مقدار تهی را وارد فیلد جدول یا دید می‌نماید.
- **expression**، حاصل یک عبارت را وارد فیلد جدول یا دید می‌نماید (برای مقدار دادن به فیلدهای محاسباتی مفید است).
- **derived_table**، خروجی یک دستور SELECT را وارد جدول می‌نماید. در ادامه دستور SELECT را می‌آموزیم.
- **execute_statement**، خروجی یک دستور SELECT یا READTEXT را به جدول یا دید اضافه می‌کند.

مثال ۱-۲. پرس‌وجوهایی که دو رکورد به جدول GroupBook اضافه می‌کنند.

```
INSERT INTO GroupBook  VALUES('01', 'برنامه نویسی')
INSERT INTO GroupBook  VALUES('02', 'پایگاه داده')
```

مثال ۲-۲. پرس‌وجویی که سه رکورد به جدول Authors اضافه می‌کنند.

```
INSERT INTO Authors VALUES('01','رمضان','عباس‌نژاد',42,
    'فوق لیسانس', 'fanavarienoin@yahoo.com', Null, 28000000)
INSERT INTO Authors VALUES('02','یوسف','عباس‌نژاد',26,'لیسانس',
      'yousf_dssht@yahoo.com', Null, Null)
INSERT INTO Authors VALUES('03','باقر','رحیم پور',32,
   'فوق لیسانس', Null, Null, 10000000)
```

نکته: در هنگام اضافه کردن رکورد با دستور INSERT باید نکات زیر را رعایت کنید:

۱. باید نوع داده‌ها نظیر به نظیر با نوع فیلدها، یکی باشند و طول مقادیر باید کوچک‌تر یا مساوی طول فیلدها باشند.

۲. برای فیلدهایی که کلید اولیه هستند، مقدار تکراری وارد نکنید. اگر کاربر مقادیر تکراری برای این فیلدها وارد کند، نه تنها رکورد اضافه نمی‌شود، بلکه پیام خطا ظاهر خواهد شد.

۳. برای فیلدهایی که قید NOT NULL دارند، نمی‌توانید مقادیر تهی وارد کنید. اگر مقدار تهی برای این فیلدها وارد کنید، پیام خطا ظاهر می‌شود.

۴. اگر دو جدول دارای ارتباط باشند، در موقع اضافه کردن مقدار فیلد ارتباط در جدول فرزند، چنانچه این مقدار برای فیلد ارتباط در جدول پدر (همان جدولی که کلید اولیه یا کلید فرعی در آن تعریف شده است)، وارد نشده باشد، پیام خطا ظاهر می‌شود.

۵. دستور زیر رکوردی با مقادیر پیش‌فرض برای فیلدهای آن اضافه می‌کند:

INSERT INTO نام جدول VALUES DFAULTS

در این دستور چنانچه، فیلدی از رکورد برای آن مقدار پیش‌فرض تعریف نشده باشد، مقدار NULL در آن فیلد قرار می‌گیرد.

۲-۲. ویرایش رکوردهای جدول

برای این که رکوردها به جداول اضافه شوند و خروجی‌ها یکنواخت گردند، دستورات زیر را تایپ کرده، اجرا نمایید:

```
USE PublishDB
GO
INSERT INTO Authors VALUES('04','مرتضی','پویان', 27, NULL,
            NULL, NULL, NULL)
INSERT INTO Authors VALUES('05','سید حجت‌اله','جلیلی',35,
            NULL, NULL, NULL,6000000 )
INSERT INTO GroupBook VALUES('03', 'شبکه رایانه ای')
INSERT INTO GroupBook VALUES('04', 'فناوری اطلاعات')
INSERT INTO GroupBook VALUES('05', 'گرافیک رایانه ای')
INSERT INTO Publishers VALUES('01','فناوری نوین',
'۰۱۱۱۲۲۵۶۶۶۸۷', 'fanavarienovin.net', 'رمضان', 'عباس نژاد',
'بابل',15)
INSERT INTO Publishers VALUES('02', 'دانش نگار',
   '02166400220', 'محمدرضا', 'حاتمی', 'تهران', 100)
INSERT INTO Publishers VALUES('03', 'دانشگاه مازندران',
   '01125226686', 'رضا', 'امینی', 'بابلسر', 10)
INSERT INTO Books VALUES('6009141395', 'اصول طراحی پایگاه
داده', 256, 73000, 1, 1, '01')
INSERT INTO Books VALUES('6009225422', 'اصول طراحی سیستم های
شی‌گرا', 432, 130000, 1, 1, '01')
INSERT INTO Books VALUES('6009141302', 'حل مسائل C++', 216,
    57000, 2, 1, '01')
INSERT INTO Books VALUES('6009141340', 'حل مسائل C# ', 256,
    62000, 2, 1, '01')
INSERT INTO Books VALUES('6009141388', 'امنیت شبکه', 192,
    65000, 1, 1, '03')
INSERT INTO Books VALUES('6009225439', 'مدیریت استراتژیک',
    272, 75000, 1, 1, '04')
INSERT INTO Books VALUES('7009225431', 'رهیافت C++', 320,
    43000, 1, 1, '01')
INSERT INTO Books VALUES('6009141328', 'ریاضی ۱', 192, 65000,
    1, 1, '03')
INSERT INTO AutBook VALUES ('6009141395', '01', 5000000)
INSERT INTO AutBook VALUES ('6009141395','03', 5000000)
INSERT INTO AutBook VALUES ('6009225422','01', 5000000)
INSERT INTO AutBook VALUES ('6009225422','03', 5000000)
INSERT INTO AutBook VALUES ('6009141302','01', 6000000)
INSERT INTO AutBook VALUES ('6009141340','01', 6000000)
INSERT INTO AutBook VALUES ('6009141388','01', 6000000)
INSERT INTO AutBook VALUES ('6009225439','05', 6000000)
INSERT INTO AutBook VALUES ('7009225431','01', 6000000)
INSERT INTO PubBook VALUES ('6009141395', '01', 6000000)
INSERT INTO PubBook VALUES ('6009225422','01', 8000000)
INSERT INTO PubBook VALUES ('6009141302','01', 6000000)
INSERT INTO PubBook VALUES ('6009141340','01', 7000000)
INSERT INTO PubBook VALUES ('6009141388','01', 6000000)
INSERT INTO PubBook VALUES ('6009225439','01', 5000000)
INSERT INTO PubBook VALUES ('7009225431','01', 3500000)
INSERT INTO PubBook VALUES ('7009225431','02', 3500000)
```

برای ویرایش رکوردهای جدول می‌توانید از دستور UPDATE به صورت زیر استفاده کنید:

```
UPDATE table_name  [TOP(exp) [PERCENT]
  SET field_list= value_list|exp_list|NULL
  WHERE condition
```

در این دستور، پارامتر table_name، نام جدولی را تعیین می‌کند که می‌خواهید رکوردهای آن را تغییر دهید. گزینه TOP، تعیین می‌کند چند رکورد اول تغییر کند و تعداد رکوردها با exp تعیین می‌شوند. گزینه PERCENT تعیین می‌کند، چند درصد رکوردهای جدول تغییر یابند. field_list، لیست فیلدهایی را تعیین می‌کند که مقادیر آن‌ها باید به یکی از مقادیر value_list، exp_list یا NULL تغییر کند و condition، شرطی را تعیین می‌کند که رکوردهای دارای آن شرط مقدار آن‌ها باید تغییر کند.

نکته: در هنگام استفاده از دستور UPDATE به نکات زیر توجه کنید:

۱. اگر در دستور UPDATE بخش WHERE ذکر نگردد، کلیه رکوردهای جدول تغییر می‌یابند. بنابراین، در هنگام استفاده از دستور UPDATE باید نهایت دقت را داشته باشید تا ناخواسته داده‌ها را تغییر ندهید.

۲. اگر با دستور UPDATE مقدار فیلد کلید اولیه یا فیلد یکتا را طوری تغییر دهید که مقدار تکراری برای آن فیلد ایجاد شود، پیام خطا ظاهر خواهد شد.

مثال ۳-۲. پرس‌وجویی که موجب نمایش خطا خواهد شد. زیرا، مقدار کد گروه کتاب 01 برای گروه برنامه‌نویسی وجود دارد و نمی‌تواند به گروه کتاب پایگاه داده تخصیص یابد. چون، فیلد groupID کلید اولیه جدول GroupBook، است.

```
UPDATE GroupBook  SET groupID ='01'
 WHERE groupName = 'پایگاه داده'
```

مثال ۴-۲. پرس‌وجویی که کد گروه کتاب "اصول طراحی پایگاه داده" را به '02' تغییر می‌دهد.

```
UPDATE Books  SET groupID='02'
WHERE Title = 'اصول طراحی پایگاه داده'
```

۳. اگر جدولی با جدول دیگر ارتباط داشته باشد و بخواهیم در جدول دوم، مقدار فیلدی که ارتباط بین دو جدول را برقرار می‌کند، تغییر دهیم (چنانچه این فیلد در جدول پدر دارای همان مقداری باشد که فعلاً مقدار آن در جدول فرزند وجود دارد)، پیام خطا ظاهر خواهد شد.

۴. اگر دو جدول با هم ارتباط داشته باشند و بخواهیم مقدار فیلد ارتباط را در جدول اولیه طوری تغییر دهیم که این مقدار در جدول دوم موجود نباشد، پیام خطایی ظاهر می‌شود.

۵. چنانچه دستور UPDATE هر یک از قیود تعریف شده در جدول را نقض کند، پیغام خطا صادر می‌گردد.

۳-۲. حذف رکوردهای جدول

برای حذف رکوردهای اضافی جدول می‌توانید از دستور DELETE به صورت زیر استفاده کنید:


```
DELETE [TOP(exp) [PERCENT]]  FROM table_name WHERE condition
```

با این پارامترها در دستور UPDATE آشنا شدید.

مثال ۵-۲. پرس‌وجویی که رکوردی از جدول Books (کتاب‌ها) را حذف می‌کند که نام کتاب آن برابر "ریاضی ۱" باشد.

```
DELETE  FROM Books
  WHERE  title='ریاضی1'
```

نکته: در هنگام استفاده از دستور DELETE باید به نکات زیر توجه داشته باشید:

۱. اگر دستور DELETE را بدون شرط استفاده کنید، کلیه رکوردهای جدول حذف خواهند شد.

۲. با دستور DELETE نمی‌توانید جدول را حذف کنید. بلکه، فقط می‌توانید رکوردهای جدول را حذف نمایید.

۳. اگر دو جدول با هم ارتباط داشته باشند و بخواهیم رکوردی را از جدول پدر حذف کنیم که مقدار فیلد ارتباط در جدول فرزند موجود باشد، پیام خطا ظاهر می‌شود (رکورد حذف نمی‌گردد).

مثال ۶-۲. پرس‌وجویی که می‌خواهد رکوردی با کد گروه 01 را از جدول GroupBook حذف کند، اما با خطا مواجه خواهد شد.

```
DELETE FROM  GroupBook   WHERE groupID = '01'
```

۴-۲. پرس‌وجوی بازیابی اطلاعات

پرس‌وجو[1]، یکی از ابزارهای بسیار مهم بازیابی داده‌ها در بانک اطلاعات است. با استفاده از پرس‌وجو می‌توانید اعمال زیر را انجام دهید:

۱. داده‌های جدول را بازیابی کنید.

۲. اطلاعات را بر اساس چند فیلد مرتب نمایید.

۳. رکوردهای تکراری را جست‌وجو نمایید.

۴. خلاصه‌سازی اطلاعات را انجام دهید.

۵. داده‌ها را گروه‌بندی نمایید.

قبل از این که پرس‌وجو را بی‌آموزیم باید عملگرها[2] و انواع آن‌ها را در SQL بی‌آموزیم. زیرا، عملگرها در تولید پرس‌وجو نقش بسزایی دارند.

۱-۴-۲. عملگرها در SQL

عملگر، نمادی است که عمل خاصی را انجام می‌دهد. به عنوان مثال، عملگر * دو مقدار را با هم ضرب کرده، نتیجه را برمی‌گرداند. عملگرها، معمولاً در عبارات[3] به کار می‌روند. هر عبارت، ترکیبی از شناسه‌ها[4]، نام فیلد، توابع[5]، لیترال‌ها، ثوابت و عملگرها است. انواع عملگرها در SQL عبارت‌اند از:

۱. عملگرهای محاسباتی
۲. عملگر انتساب
۳. عملگرهای بیتی
۴. عملگرهای رابطه‌ای
۵. عملگر یکانی
۶. عملگر اتصال رشته‌ای
۷. عملگرهای ویژه

---

[1].Query [2].Operators [3].Expressions [4].Identifiers [5].Functions

جدول ۲-۱ عملگرهای محاسباتی در SQL

عملگر	نام	مثال	نتیجه	هدف
+	جمع	۵ + ۲	۷	مجموع عملوند اول و دوم را محاسبه می‌کند.
-	تفریق	۳ - ۶	۳	تفاضل عملوند دوم از عملوند اول را محاسبه می‌کند.
*	ضرب	۶ * ۳	۱۸	حاصل ضرب دو عملوند را محاسبه می‌کند.
/	تقسیم	۳۳ / ۳	۱۱	حاصل تقسیم عملوند اول بر عملوند دوم را محاسبه می‌کند.
%	باقی‌مانده تقسیم صحیح	۳۰ % ۷	۲	باقی‌مانده تقسیم صحیح عملوند اول بر عملوند دوم را محاسبه می‌کند.

عملگرهای محاسباتی

این عملگرها برای انجام محاسبات بر روی داده‌ها به کار می‌روند. جدول ۱- ۲ عملگرهای محاسباتی را نشان می‌دهد. با اغلب این عملگرها آشنا هستید. ممکن است با عملگر % آشنایی نداشته باشید. این عملگر باقی‌مانده تقسیم صحیح عملوند اول بر عملوند دوم را برمی‌گرداند (تقسیم به صورت صحیح انجام می‌شود نه به صورت اعشاری). به عنوان مثال، نتیجه ۴ % ۱۱ برابر ۳ است.

عملگر انتساب

عملگر =، نتیجه ارزیابی یک عبارت را به متغیر نسبت می‌دهد. به عنوان مثال، دستورات زیر را مشاهده نمایید:

```
SET @k = 10
SET @Fname = 'Ali'
```

دستور اول، مقدار ۱۰ را در k و دستور دوم، رشته 'Ali' را در Fname قرار می‌دهد.

عملگرهای بیتی

عملگرهایی هستند که بین دو عملوند از نوع صحیح قرار گرفته، عملیات بیتی از قبیل «و» بیتی، «یا» بیتی و «یا انحصاری» بیتی را انجام می‌دهند (جدول۲- ۲).

عملگرهای رابطه‌ای

عملگرهایی هستند که دو عبارت (عملوند) را با هم مقایسه کرده، نتیجه «مقایسه را به صورت منطقی برمی‌گردانند (جدول۳- ۲).

جدول ۲-۲ عملگرهای بیتی در SQL

عملگر	نام	مثال	نتیجه	هدف
&	«و» بیتی	۷۵ & ۱۷۰	۱۰	عملوند اول و دوم را بیت به بیت «و» منطقی می‌نماید.
|	«یا» بیتی	۷۵ | ۱۷۰	۲۳۵	عملوند اول و دوم را بیت به بیت «یا» منطقی می‌نماید.
^	«یا بیتی انحصاری» (XOR)	۷۵ ^ ۱۷۰	۲۲۵	عملوند اول و دوم را بیت به بیت با هم XOR می‌نماید.

عملگرهای منطقی

عملگرهایی هستند که بین دو عبارت منطقی قرار گرفته، نتیجه دو عبارت منطقی را بررسی می‌کنند. عبارات منطقی دارای یکی از سه ارزش درستی[1]، نادرستی[2] و یا تهی[3] هستند. این عملگرها در جدول ۴- ۲ آمده‌اند.

جدول ۲- ۲ عملگرهای رابطه‌ای در SQL

عملگر	نام	مثال	نتیجه	هدف
=	تساوی	۵ = ۱۰	نادرست	دو عملوند را با یکدیگر مقایسه می‌کند، در صورتی که با هم برابر باشند، نتیجه درست است.
>	بزرگ‌تر	۵ > ۱۰	درست	اگر عملوند اول بزرگ‌تر از عملوند دوم باشد، نتیجه درست است.
<	کوچک‌تر	۵ < ۱۰	نادرست	اگر عملوند اول کوچک‌تر از عملوند دوم باشد، نتیجه درست است.
>=	بزرگ‌تر مساوی	۷ =< ۱۰	درست	اگر عملوند اول بزرگ‌تر یا مساوی عملوند دوم باشد، نتیجه درست است.
<=	کوچک‌تر مساوی	۱۵=<۱۰	درست	اگر عملوند اول کوچک‌تر یا مساوی عملوند دوم باشد، نتیجه درست است.
< >	نامساوی	۱۵<>۱۰	درست	اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است.
=!	نامساوی	۱۰=!۱۰	نادرست	اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است.
!<	کوچک‌تر نیست	۱۲!<۱۰	نادرست	اگر عملوند اول کوچک‌تر از عملوند دوم نباشد، نتیجه درست است.
!>	بزرگ‌تر نیست	۱۲ !>۱۰	درست	اگر عملوند اول بزرگ‌تر از عملوند دوم نباشد، نتیجه درست است.

جدول ۲- ۳ عملگرهای منطقی در SQL

X	Y	X AND Y	X OR Y	Not X
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

جدول ۲- ۵ عملگرهای یکانی در SQL

عملگر	نام	مثال	نتیجه	هدف
+	مثبت	+۵	۵	مقدار مثبت عملوند را برمی‌گرداند.
-	منفی	-۱۵	-۱۵	مقدار منفی عملوند را برمی‌گرداند.
~	متمم یک	~۱۰۱	۰۱۰	متمم یک عملوند را برمی‌گرداند.

عملگرهای یکانی

بر روی یک عملوند صحیح عمل می‌کنند. جدول ۵- ۲ عملکرد این عملگرها را نمایش می‌دهد.

---

[1].True [2].False [3].Null

عملگر اتصال رشته‌ای

عملگر است که بین دو عملوند از نوع رشته‌ای قرار گرفته آن‌ها را به هم الحاق می‌کند.

مثال ۷ – ۲. دستوراتی که رشته «Database with SQL» را در متغیر @S3 قرار می‌دهند.

```
SET @S1 = "Database with "
SET @S2 = "SQL"
SET @S3 = @S1 + @S2
```

عملگرهای ویژه

عملگرهایی هستند که در SQL کاربردهای خاصی دارند. بعضی از این عملگرها در زیر آمده‌اند:

عملگر IN: تعیین می‌کند که آیا مقداری در مجموعه‌ای از مقادیر وجود دارد یا خیر. دستور زیر را ببینید:

```
Title IN('حل مسائل#C','حل مسائل ++C','اصول طراحی پایگاه داده')
```

این دستور مشخص می‌کند آیا فیلد نام کتاب (Title) برابر یکی از مقادیر «اصول طراحی پایگاه داده»، حل مسائل ++C یا حل مسائل #C» است یا خیر. این دستور، معادل دستور زیر است:

```
Title = 'اصول طراحی پایگاه داده' OR Title = 'حل مسائل++C'
    OR Title = 'حل مسائل#C'
```

عملگر BETWEEN ... AND: تعیین می‌کند که نتیجه یک عبارت یا مقدار یک فیلد، بین دو مقدار قرار دارد یا خیر. دستور زیر را در نظر بگیرید:

```
Score BETWEEN 0 AND 20
```

این دستور تعیین می‌نماید آیا نمره (Score) بین ۰ تا ۲۰ است یا خیر.

عملگر ALL: یک مقدار را با کلیه مقادیر یک ستون یا چند مقدار مقایسه می‌کند. اگر شرط مقایسه مقدار با تمام مقادیر ستون، دارای ارزش درستی باشد، عملگر ALL، نتیجه True، وگرنه، نتیجه False را برمی‌گرداند. دستور زیر را ببینید:

```
3 <= ALL SELECT Score FROM Score
```

این دستور تعیین می‌کند آیا تمام نمره‌ها بزرگ‌تر یا مساوی ۳ هستند یا خیر.

عملگر SOME یا ANY: مقداری را با مجموعه‌ای از مقادیر مقایسه می‌کند. اگر حداقل یکی از مقادیر در شرط صدق کند، نتیجه True، وگرنه نتیجه False برگشت داده می‌شود.

عملگر LIKE : تعیین می‌کند رشته‌ای در بخشی از فیلد قرار دارد یا خیر. این عملگر به صورت زیر به کار می‌رود:

```
match_expr [NOT] LIKE Pattern[ESCAPE excap_Char]
```

match_expr، عبارت یا رشته‌ای از کاراکترها است.

Pattern، الگویی است که باید در match_expr جست‌وجو گردد که می‌تواند حاوی کاراکترهای جدول ۶– ۲ باشد.

جدول ۶-۲ کاراکترهای الگوی عملگر LIKE

کاراکتر	مثال	عبارت مورد جست‌وجو	عملکرد
%	LIKE '%Computer%'	Computer Computer NetWork Net Computer Computers ...	به جای کاراکتر % می‌تواند تعدادی کاراکتر قرار گیرند یا هیچ کاراکتری قرار نگیرد.
-	LIKE '-ean'	Sean Dean ...	به جای این کاراکتر فقط یک کاراکتر قرار می‌گیرد یا هیچ کاراکتری قرار نمی‌گیرد.
$[a_1-a_2]$	LIKE '[a-o]ook'	aook book cook	یک سری کاراکترها که بین a_1 تا a_2 باشند، می‌توانند به جای $[a_1-a_2]$ قرار بگیرند.
$[^a_1-a_2]$	LIKE '[^d-f]ell'	Cell Mell ...	کاراکترهایی که بین a_1 تا a_2 نباشند می‌توانند به جای $[^a_1-a_2]$ قرار بگیرند.

۲-۴-۲. بازیابی اطلاعات از جدول با دستور SELECT

دستور SELECT برای بازیابی اطلاعات از جدول به کار می‌رود. ساختار اصلی دستور SELECT به صورت زیر است:

```
SELECT  A1, A2,A3,... An  یا  *
FROM    r1, r2, ..., rn
WHERE   p
```

همان‌طور که در این ساختار می‌بینید، دستور SELECT از سه بخش اصلی زیر تشکیل شده است:

- **بخش SELECT**، مانند عملگر پرتو در جبر رابطه‌ای است. این بخش صفاتی (ستون‌ها یا فیلدها) که باید نمایش داده شوند را تعیین می‌کند. $A_1, A_2, \ldots, A_n$ صفاتی هستند که باید در نتیجه دستور SELECT نمایش داده شوند. اگر به جای $A_1, A_2, \ldots, A_n$ از نماد * استفاده شود، کلیه ستون‌های (فیلدهای) جدول رابطه نمایش داده می‌شوند.
- **بخش FROM**، معادل عملگر ضرب دکارتی در جبر رابطه‌ای است. یعنی، لیست رابطه‌هایی $(r_1,r_2,\ldots,r_n)$ را مشخص می‌کند که در بازیابی اطلاعات شرکت می‌کنند (در این جا لیست جداول را تعیین می‌کنند).
- **بخش WHERE**، معادل عملگر گزینش (σ) در جبر رابطه‌ای است. در این بخش می‌توان شرط انتخاب رکوردها یا شرط پیوند جداول را مشخص کرد.

مثال ۸-۲. پرس‌وجویی که نام و نام خانوادگی مؤلفین را نمایش می‌دهد.

```
SELECT aFname, aLname
FROM Authors
```

همان‌طور که در این خروجی می‌بینید، فقط فیلدهای نام و نام خانوادگی مؤلفین نمایش داده شده‌اند.

	aFname	aLname
1	رمضان	عباس نژاد
2	یوسف	عباس نژاد
3	باقر	رحیم پور
4	مرتضی	پویان
5	سید حجت اله	جلیلی

مثال ۹-۲. پرس‌وجویی که نام کتاب و قیمت آن را نمایش می‌دهد.

```
SELECT Title, Price
FROM Books
```

	Title	Price
1	حل مسائل C++	57000
2	حل مسائل C#	62000
3	امنیت شبکه	65000
4	اصول طراحی پایگاه داده	73000
5	اصول طراحی سیستم های شی گرا	130000
6	مدیریت استراتژیک	75000
7	رهافت C++	43000

مثال ۱۰-۲. پرس‌وجویی که شابک کتاب‌های تألیف شده که حق تألیف برای آن‌ها دریافت شده، را نمایش می‌دهد.

```
SELECT ISBN
FROM AutBook
```

	ISBN
1	6009141302
2	6009141340
3	6009141388
4	6009141395
5	6009141395
6	6009225422
7	6009225422
8	6009225439
9	7009225431

همان‌طور که در این خروجی می‌بینید، برخی از شماره‌های دانشجویی تکرار شده‌اند. اگر بخواهید مقادیر تکراری را از خروجی دستور SELECT حذف کنید، باید بعد از SELECT از کلمه DISTINCT استفاده کنید (مثال زیر را ببینید):

مثال ۱۱-۲. پرس‌وجویی که شابک کتاب‌های تألیف شده را نمایش می‌دهد که حق تألیف برای آن‌ها دریافت شد (شماره‌های تکراری را فقط یک بار نمایش دهد).

```
SELECT DISTINCT ISBN
FROM AutBook
```

	ISBN
1	6009141302
2	6009141340
3	6009141388
4	6009141395
5	6009225422
6	6009225439
7	7009225431

همان‌طور که در این خروجی مشاهده می‌کنید، شابک‌ها فقط یک بار نمایش داده شده‌اند.

نکته: همان‌طور که بیان گردید، بخش WHERE برای بازیابی رکوردهایی که دارای شرط خاص هستند، به کار می‌رود. در بخش WHERE می‌توانید از عملگرهای مقایسه‌ای (<، >، =<، =>، =، <>)، منطقی (AND، OR و NOT) و عملگرهای ویژه (BETWEEN، IN، LIKE، ALL، SOME، ANY و غیره) استفاده کنید. چند نمونه از این پرس‌وجوها در زیر آمده‌اند.

مثال ۱۲-۲. پرس‌وجویی که کتاب‌های را نمایش می‌دهد که نام آن‌ها " امنیت شبکه " است.

```
SELECT * FROM Books
WHERE Title= 'امنیت شبکه'
```

	ISBN	Title	Page	Price	editNo	printNo	groupID
1	6009141338	امنیت شبکه	192	65000	1	1	03

مثال ۱۳-۲. پرس‌وجویی که لیست حق تألیف بین ۵۰۰۰۰۰۰ تا ۶۰۰۰۰۰۰ را نمایش می‌دهد.

```
SELECT * FROM AutBook
WHERE Payment BETWEEN 5000000AND6000000
```

	ISBN	atID	Payment
1	6009141302	01	6000000
2	6009141340	01	6000000
3	6009141388	01	6000000
4	6009141395	01	5000000
5	6009141395	03	5000000
6	6009225422	01	5000000
7	6009225422	03	5000000
8	6009225439	05	6000000
9	7009225431	01	6000000

مثال ۱۴ - ۲. پرس‌وجویی که نام، نام خانوادگی و مدرک مؤلفین را نمایش می‌دهد که در مدرک آن‌ها کلمه "فوق" وجود دارد.

```
SELECT aFname, aLname, Ranking FROM Authors
WHERE Ranking LIKE '%فوق%'
```

	aFname	aLname	Ranking
1	رمضان	عباس نژاد	فوق لیسانس
2	باقر	رحیم پور	فوق لیسانس

همان‌طور که در این خروجی می‌بینید، فیلدهای نام، نام خانوادگی و مدرک مؤلفین را بازیابی کرد که عبارت "فوق" در مدرک آن‌ها وجود دارد.

مثال ۱۵ - ۲. پرس‌وجویی که نام و نام خانوادگی مؤلفین را نمایش می‌دهد که نام آن‌ها "علی"، "باقر" یا "یوسف" باشد.

```
SELECT aFname, aLname FROM Authors
WHERE aFname IN ('علی', 'یوسف', 'باقر')
```

	aFname	aLname
1	یوسف	عباس نژاد
2	باقر	رحیم پور

مثال ۱۶ - ۲. پرس‌وجویی که نام و نام خانوادگی مؤلفین را نمایش می‌دهد که نام خانوادگی آن‌ها با حرف "د" خاتمه یافتند.

```
SELECT aFname, aLname FROM Authors
WHERE aLname LIKE '%د'
```

	aFname	aLname
1	رمضان	عباس نژاد
2	یوسف	عباس نژاد

مثال ۱۷ - ۲. پرس‌وجویی که نام و نام‌خانوادگی مؤلفین را نمایش می‌دهد که نام آن‌ها دقیقاً چهار کاراکتر باشد.

```
SELECT aFname, aLname FROM Authors
WHERE aFname LIKE '____'
```

	aFname	aLname
1	یوسف	عباس نژاد
2	باقر	رحیم پور

مثال ۱۸ - ۲. پرس‌وجویی که نام و نام خانوادگی تمام مؤلفین را نمایش می‌دهد که نام و نام خانوادگی آن‌ها حداقل پنج حرف دارد.

```
SELECT aFname, aLname FROM Authors
WHERE aFname LIKE '___%' and aLname LIKE '__%'
```

	aFname	aLname
1	رمضان	عباس نژاد
2	مرتضی	پویان
3	سید حجت اله	جنیدی

نکته: در هنگام استفاده از دستور SELECT می‌توان عنوان ستون‌ها را تغییر داد. برای این منظور می‌توانید از کلمه AS استفاده کرده، بعد از عبارت AS نام جدید ستون را تایپ نمایید. چنانچه نام ستون بیش از یک کلمه باشد و بین کلمات جای خالی[1] وجود دارد باید نام ستون را در بین تک گیومه (') قرار دهید.

مثال ۱۹ - ۲. پرس‌وجویی که شابک و نام کتاب‌هایی که چاپ دوم هستند را نمایش می‌دهد (در عنوان ستون به جای ISBN، شابک و به جای Title، نام کتاب را نمایش می‌دهد).

---

[1] .Blank

	aFname	aLname
1	رمضان	عباس نژاد
2	یوسف	عباس نژاد
3	باقر	رحیم پور
4	مرتضی	پویان
5	سید حجت اله	جلیلی
6	رمضان	عباس نژاد
7	محمد رضا	حاتمی
8	رضا	امینی

همان‌طور که در این خروجی می‌بینید، برخی از نام و نام خانوادگی‌ها تکرار شده‌اند.

۳-۶-۲. عملگر INTERSECT

این عملگر از نتیجه چند پرس‌وجو اشتراک می‌گیرد (مانند عملگر ∩ جبر رابطه‌ای) و نتیجه اشتراک را برمی‌گرداند.

مثال ۲۵-۲. پرس‌وجویی که کد مؤلفینی که کتاب تألیف کرده‌اند را نمایش می‌دهد.

```
SELECT atID  FROM Authors
INTERSECT
SELECT atID FROM AutBook
```

	atID
1	01
2	03
3	05

مثال ۲۶-۲. پرس‌وجویی که نام و نام خانوادگی افرادی که هم مؤلف و هم مدیر انتشارات هستند را نمایش می‌دهد.

```
SELECT aFname, aLname FROM Authors
INTERSECT
SELECT bFname, bLname FROM Publishers
```

	aFname	aLname
1	رمضان	عباس نژاد

۴-۶-۲. عملگر EXCEPT

این عملگر تفاضل بین نتیجه پرس‌وجوی اول و دوم را برمی‌گرداند (مانند عملگر – در جبر رابطه‌ای). یعنی، رکوردهایی را برمی‌گرداند که در پرس‌وجوی اول باشد، ولی در نتیجه پرس‌وجوی دوم نباشد.

مثال ۲۷-۲. پرس‌وجویی که کد مؤلفینی که کتاب چاپ شده در این بانک ندارند، را نمایش می‌دهد.

```
SELECT atID FROM Authors
EXCEPT
SELECT atID FROM AutBook
```

	atID
1	02
2	04

مثال ۲۸-۲. پرس‌وجویی که نام و نام خانوادگی رواسای انتشارات را نمایش می‌دهد که مؤلف نیستند.

```
SELECT bFname, bLname FROM Publishers
EXCEPT
SELECT aFname, aLname FROM Authors
```

	bFname	bLname
1	رضا	امینی
2	محمدرضا	حاتمی

نکته: عملگرهای UNION، INTERSECT و EXCEPT، رکوردهای تکراری را یک بار نمایش می‌دهند. برای نمایش رکوردهای تکراری می‌توان از عملگرهای UNION ALL، INTERSECT ALL و EXCEPT ALL استفاده کرد.

نکته: همان‌طور که بیان گردید، عبارت ORDER BY برای مرتب‌سازی رکوردها به کار می‌رود. در هنگام به کارگیری ORDER BY در پرس‌وجوهای مرکب باید دقت کنید که این عبارت را فقط در یک پرس‌وجو تایپ کنید.

مثال ۲۹-۲. پرس‌وجویی که شماره مؤلفینی را نمایش می‌دهد که در جدول AutBook حق تألیف گرفته‌اند (اطلاعات را بر اساس شماره مؤلف به صورت نزولی مرتب کرده، نمایش می‌دهد).

```
SELECT atID FROM Authors
EXCEPT
SELECT atID FROM AutBook ORDER BY atID DESC
```

	atID
1	04
2	02

جدول ۷-۲ توابع تجمعی (خلاصه‌سازی)

تابع	ورودی	هدف
AVG	عددی	میانگین مقادیر را برمی‌گرداند.
SUM	عددی	مجموع مقادیر را برمی‌گرداند.
MAX	عددی یا غیر عددی	بزرگ‌ترین مقدار را برمی‌گرداند.
MIN	عددی یا غیر عددی	کوچک‌ترین مقدار را برمی‌گرداند.
COUNT	عددی یا غیر عددی	تعداد رکوردها یا مقادیر غیر تهی را برمی‌گرداند.
STDEV	عددی	انحراف معیار یک فیلد یا چند مقدار را برمی‌گرداند.
STDEVP	عددی	انحراف معیار توزیعی یک فیلد یا چند مقدار را برمی‌گرداند
VAR	عددی	واریانس مقادیر یک یا چند فیلد را برمی‌گرداند.
VARP	عددی	واریانس توزیعی مقادیر یک فیلد یا چند مقدار را برمی‌گرداند

۷-۲. توابع تجمعی

توابع تجمعی برای خلاصه‌سازی اطلاعات به کار می‌روند (یعنی، مجموعه‌ای (کلکسیونی) از مقادیر را به عنوان ورودی دریافت کرده، یک مقدار را برمی‌گرداند). با توابع تجمعی می‌توانید مجموع مقادیر، بزرگ‌ترین مقدار، کوچک‌ترین مقدار، میانگین مقادیر یک یا چند فیلد را برگردانید. نمونه‌ای از این توابع در جدول ۷-۲ آمده‌اند.

مثال ۳۰-۲. پرس‌وجویی که تعداد کتاب‌ها را برمی‌گرداند.

```
SELECT COUNT(*) FROM Authors
```

	(No column name)
1	5

۸-۲. گروه‌بندی اطلاعات

گروه‌بندی اطلاعات به عملی گفته می‌شود که فیلدهای با مقادیر تکراری با یک ترتیب منطقی ترکیب گردند. به عنوان مثال، در جدول AutBook ممکن است مجموع، میانگین، بزرگ‌ترین، کوچک‌ترین حق تألیف هر مؤلف را بخواهید. برای این منظور، باید از واژه GROUP BY استفاده کرده، پس از آن atID (شماره مؤلف) را بی‌آورید.

مثال ۳۱-۲. پرس‌وجویی که میانگین، مجموع، حداکثر و حداقل حق تألیف هر مؤلف را نمایش می‌دهد.

```
SELECT  atID 'شماره مولف',AVG(Payment) 'میانگین حق تالیف',SUM
        (Payment) 'مجموع حق تالیف',MAX(Payment) 'بیشترین حق تالیف'
        ,MIN(Payment) 'کمترین حق تالیف'
FROM AutBook
GROUP BY atID
```

	شماره مولف	میانگین حق تالیف	مجموع حق تالیف	بیشترین حق تالیف	کمترین حق تالیف
1	01	5666666.666666	34000000	6000000	5000000
2	03	5000000.000000	10000000	5000000	5000000
3	05	6000000.000000	6000000	6000000	6000000

نکته: در بعضی موارد، قبل از اجرای توابع تجمعی باید رکوردهای تکراری را یک بار نمایش داد. برای این منظور می‌توان از کلمه DISTINCT استفاده نمود.

مثال ۳۲ – ۲. پرس‌وجویی که تعداد ناشرانی که حق نشر گرفته‌اند را نمایش می‌دهد.

```
SELECT COUNT(DISTINCT pubID) FROM PubBook
```

	(No column name)
1	2

نکته: در دستور SQL در هنگام استفاده (*) COUNT نمی‌توان از کلمه DISTINCT استفاده کرد. چنانچه DISTINCT را با توابع MAX، MIN به کار ببرید، نتیجه تغییر نمی‌کند.

استفاده از HAVING در دستور SELECT

در بخش WHERE دستور SELECT می‌توانید شرط‌های معمولی را تست کنید. گاهی نیاز است شرط‌هایی را استفاده کنید که نتیجه یک تابع تجمعی است. در این مواقع باید از بخش HAVING استفاده کنید.

مثال ۳۳ – ۲. پرس‌وجویی که تعداد ناشرانی را نمایش می‌دهد که برای بیش از ۲ کتاب حق نشر دریافت کرده‌اند.

```
SELECT pubID ,COUNT(*) FROM PubBook
WHERE COUNT(*) > 2
GROUP BY pubID
```

با اجرای این دستور خروجی زیر ظاهر می‌گردد:

```
Msg 147, Level 15, State 1, Line 2
An aggregate may not appear IN the WHERE clause unless it
is IN a subquery cOntained IN a HAVING clause or a SELECT
list, AND the column being aggregated is an outer
reference.
```

این خروجی بیان می‌کند که توابع تجمعی را نمی‌توان در شرط WHERE استفاده کرد. به جای WHERE باید از HAVING استفاده نمود. اکنون دستور زیر را تایپ کنید:

```
SELECT pubID ,COUNT(*) FROM PubBook
GROUP BY pubID
HAVING COUNT(*) > 2
```

	pubID	(No column name)
1	01	7

نکته: اگر بخواهید بخش‌های WHERE و HAVING را در یک پرس‌وجو استفاده کنید، ابتدا بخش WHERE، سپس بخش GROUP BY و در پایان، بخش HAVING را بی‌آورید.

مثال ۳۴ – ۲. پرس‌وجویی که تعداد کتاب‌های با کدهای گروه کتاب "۰۱"، "۰۲" یا "۰۳" اگر برابر ۱ باشد را نمایش می‌دهد.

```
SELECT groupID, COUNT(*)  FROM Books
WHERE groupID IN ('01', '02', '03')
GROUP BY groupID
HAVING COUNT(*) =1
```

	groupID	(No column name)
1	02	1
2	03	1

جدول ۸ – ۲ نتیجه عملگرهای منطقی وقتی مقدار Unknown در عملوند باشد

p	q	p AND q	p OR q	NOT q
Unknown	False	False	Unknown	True
Unknown	True	Unknown	True	False
Unknown	Unknown	True	Unknown	Unknown

۹– ۲. تست مقدار تهی فیلد

همان‌طور که بیان گردید، مقدار ستون (فیلد) می‌تواند تهی باشد. بنابراین، باید بتوان رکوردهایی (تاپل‌هایی) را بازیابی کرد که مقدار فیلد آن تهی باشد. برای این منظور، از کلمه NULL استفاده می‌گردد.

مثال ۳۵ – ۲. پرس‌وجویی که نام و نام خانوادگی مؤلفانی را نمایش می‌دهد که مقدار فیلد پست الکترونیکی آن وارد نشده باشد.

```
SELECT aFname as نام,aLname AS'نام خانوادگی'
FROM Authors WHERE Email IS NULL
```

	نام	نام خانوادگی
1	مرتضی	یوبان
2	سیدحجت اله	جلیلی

برای تست تهی نبودن از عبارت IS NOT NULL استفاده می‌گردد.

اگر مقادیر ورودی عملگرهای محاسباتی (+، -، * و /) تهی باشد، نتیجه عبارت تهی خواهد شد. اما، اگر مقادیر ورودی عملگرهای مقایسه‌ای (<، =<، >، =>، = و <>) تهی باشند، نتیجه عبارت **ناشناخته**[1] خواهد شد. مقدار Unknown هیچ یک از مقادیر NULL و NOT NULL نیست. اگر یکی از عملوندهای عملگرهای AND، OR و NOT ناشناخته باشد، نتیجه عبارت منطقی مطابق جدول ۸ – ۲ خواهد بود.

۱۰ – ۲. پرس‌وجوی فرعی

پرس‌وجویی که نتیجه آن توسط پرس‌وجوی دیگر مورد استفاده قرار می‌گیرد، **پرس‌وجوی فرعی**[2] (**متداخل**[3]) نام دارد. در پرس‌وجوهای فرعی، ممکن است چندین پرس‌وجو به صورت پشت سرهم قرار گیرند. در این صورت، نتیجه پرس‌وجوی سطح بعدی در سطح قبلی استفاده می‌شود و این روند تا اولین سطح پرس‌وجو ادامه می‌یابد. پرس‌وجوهای فرعی را می‌توان در عبارت WHERE و HAVING مربوط به پرس‌وجوی اصلی استفاده کرد. در هنگام استفاده از پرس‌وجوهای فرعی باید نکات زیر را رعایت کنید:

- از عملگر BETWEEN نمی‌توان در پرس‌وجوی فرعی استفاده کرد. ولی، در پرس‌وجوی اصلی می‌توان از این عملگر استفاده نمود.
- پرس‌وجوهای فرعی باید در داخل پرانتز قرار گیرند.
- پرس‌وجوهای فرعی که بیش از یک رکورد (تاپل) را برمی‌گردانند، می‌توانند با عملگرهایی مانند IN مورد استفاده قرار گیرند.
- در یک پرس‌وجوی فرعی نمی‌توان از گزینه ORDER BY استفاده کرد. برای این منظور، می‌توان از گزینه GROUP BY استفاده نمود تا عمل ORDER BY نیز انجام شود.

---

[1].Unknown [2].Sub Query [3].Nested Query

پرس‌وجوی فرعی به صورت زیر به کار می‌رود:

دستور پرس‌وجوی اصلی
...
WHERE **[لیست فیلدها] عملگر (دستور پرس‌وجوی فرعی)**

برخی از مثال‌های پرس‌وجوی فرعی را در ادامه می‌بینید.

مثال ۳۶ – ۲. پرس‌وجویی که ناشرانی را نمایش می‌دهد که حداقل برای یک کتاب حق نشر دریافت کرده‌اند.

برای حل این مثال، ابتدا شماره ناشرانی که در جدول PubBook وجود دارند را مشخص می‌کنیم، یعنی:

```
SELECT pubID FROM PubBook
```

سپس، از جدول Publishers، ناشرانی را نمایش می‌دهیم که شماره آن‌ها در نتیجه پرس‌وجوی قبلی موجود باشد. یعنی:

```
SELECT * FROM Publishers
WHERE pubID IN (SELECT pubID FROM PubBook)
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	01	فناوری نوین	011122566687	fanavanenovin.net	بابل	رمضان	عباس نژاد	15
2	02	دانش نگار	02166400220		تهران	محمد رضا	حاتمی	100

مثال ۳۷ – ۲. پرس‌وجویی که ناشرانی را نمایش می‌دهد که هیچ حق نشری دریافت نکردند (برای انجام این پرس‌وجو از عملگر NOT IN استفاده می‌شود).

```
SELECT * FROM Publishers
WHERE pubID NOT IN (SELECT pubID FROM PubBook)
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	03	دانشگاه مازندران	01125226686		بابلسر	رضا	امینی	10

مثال ۳۸ – ۲. پرس‌وجویی که لیست گروه کتاب‌های را بازیابی می‌کند که حداقل یک کتاب از آن گروه در جدول Books وجود دارد.

```
SELECT * FROM GroupBook g
WHERE EXISTS (SELECT * FROM Books WHERE groupID = g.groupID)
```

	groupID	groupName
1	01	برنامه نویسی
2	02	پایگاه داده
3	03	شبکه رایانه ای
4	04	فناوری اطلاعات

همان‌طور که در این پرس‌وجو می‌بینید، از عملگر EXISTS استفاده گردید. این عملگر معنی **شامل شدن** را می‌دهد که به جای این دستور می‌توان از عملگر IN به صورت زیر استفاده کرد:

```
SELECT * FROM GroupBook
WHERE groupID IN (SELECT groupID FROM Books)
```

مثال ۳۹ – ۲.پرس‌وجویی که لیست مؤلفانی را نمایش می‌دهد که تاکنون حق تألیف دریافت نکردند.

```
SELECT * FROM Authors a
WHERE NOT EXISTS (SELECT * FROM AutBook  WHERE atID = a.atID )
```

	atID	aFname	aLname	Age	Ranking	Email	Mobile	sumPayment
1	02	یوسف	عباس نژاد	26	لیسانس	yousef_dsslt@yahoo.com	NULL	NULL
2	04	مرتضی	پویان	27	NULL	NULL	NULL	NULL

همان‌طور که در این پرس‌وجو می‌بینید، عملگر NOT EXISTS مولفینی که هیچ حق تالیفی نگرفته‌اند را تعیین می‌کند. به جای این دستور می‌توانید از عملگر NOT IN به صورت زیر استفاده کنید:

```
SELECT * FROM Authors
WHERE atID NOT IN (SELECT atID  FROM AutBook)
```

مثال ۴۰ – ۲. پرس‌وجویی که گروه‌هایی که اصلاً کتاب ندارند را نمایش می‌دهد (برای حل این پرس‌وجو از عملگر EXCEPT استفاده کنید).

```
SELECT * FROM GroupBook
WHERE groupID IN
(SELECT groupID  FROM GroupBook
 EXCEPT
 SELECT groupID  FROM Books )
```

	groupID	groupName
1	05	گرافیک رایانه ای

پرس‌وجوی فرعی کد گروه‌هایی را برمی‌گرداند که کتاب ندارند (با عملگر EXCEPT) و پرس‌وجوی اصلی اطلاعات این گروه‌ها را نمایش می‌دهد.

مثال ۴۱ – ۲. پرس‌وجویی که کتاب‌های را نمایش می‌دهد که حق تألیف برای آن‌ها دریافت گردید (با عملگر INTERSECT).

```
SELECT * FROM Books
WHERE ISBN IN
(SELECT ISBN  FROM Books
 INTERSECT
 SELECT ISBN  FROM AutBook)
```

	ISBN	Title	Page	Price	edtNo	prntNo	groupID
1	6009141302	حل مسائل C++	216	57000	1	2	01
2	6009141340	حل مسائل C#	256	62000	1	2	01
3	6009141388	امنیت شبکه	192	65000	1	1	03
4	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
5	6009225422	اصول طراحی سیستم های شی گرا	432	133000	1	1	01
6	6009225439	مدیریت استراتژیک	272	75000	1	1	04
7	7009225431	رهیافت C++	320	43000	1	1	01

Query executed successfully. | 1-VAIO\YOUSEF (10.0 RTM) | 1-VAIO\1 (54) | PublishDB | 00:00:00 | 7 ro

پرس‌وجوی فرعی، کد کتاب‌هایی را برمی‌گرداند که در جدول AutBook هستند. سپس، پرس‌وجوی اصلی، اطلاعات کتاب‌ها را نمایش می‌دهد.

مثال ۴۲ – ۲. پرس‌وجویی که کتاب‌های با کد گروه کتاب '02' را نمایش می‌دهد که تاکنون برای آن‌ها حق تألیف دریافت نکردید (با عملگر EXCEPT).

```
SELECT * FROM Books
WHERE ISBN IN
(SELECT ISBN  FROM Books
WHERE groupID ='01'
 EXCEPT
 SELECT ISBN  FROM AutBook)
```

ISBN	Title	Page	Price	edtNo	prntNo	groupID

پرس‌وجوی فرعی شابک کتاب‌های با کد گروه '01' که هیچ حق تألیفی برای آن‌ها دریافت نگردید را برمی‌گرداند و پرس‌وجوی اصلی، کلیه اطلاعات این کتاب‌ها را نمایش می‌دهد.

مثال ۴۳ – ۲. پرس‌وجویی که لیست ناشرانی که برای کتاب‌های با کد گروه '02' حق تالیف داده‌اند را نمایش می‌دهد.

```
SELECT * FROM Publishers p WHERE EXISTS
(SELECT * FROM PubBook
WHERE ISBN IN
(SELECT ISBN  FROM Books WHERE groupID ='02'
 INTERSECT
 SELECT ISBN  FROM AutBook)
AND p.pubID = pubID)
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	01	فناوری نوین	011122566687	fanavarienovin.net	بابل	رمضان	عباس نژاد	15

این پرس‌وجو از دو پرس‌وجوی فرعی و یک پرس‌وجوی اصلی تشکیل شده است. پرس‌وجو فرعی اول، شماره کتاب‌های با کد گروه '02' را برمی‌گرداند. پرس‌وجوی فرعی دوم، شماره کتاب‌های که برای آن‌ها حق تألیف داده شده است را برمی‌گرداند و پرس‌وجوی اصلی، اطلاعات ناشران را بازیابی می‌کند.

۱۱ – ۲. پیوند جداول (رابطه)

تاکنون پرس‌وجوهایی که دیدید، برای بازیابی اطلاعات از چند جدول از پرس‌وجوی فرعی استفاده می‌کردند. برای بازیابی اطلاعات از چند جدول (اطلاعات در چند جدول توزیع شده باشند)، نمی‌توان از پرس‌وجوی فرعی استفاده کرد. به عنوان مثال، فرض کنید پرس‌وجویی بخواهید که بتواند اطلاعات نام، نام خانوادگی مؤلف، نام کتاب و حق تألیف را بازیابی کند. برای این منظور باید جداول کتاب، مؤلف و حق تألیف را با هم پیوند زده، این اطلاعات را از آن‌ها بازیابی کنید. همان‌طور که بیان گردید، برای پیوند یا الحاق بین دو جدول، باید حداقل یک فیلد مشترک بین آن‌ها وجود داشته باشد. فیلدهای مشترک دو جدول باید نوع یکسان داشته باشند (نام فیلدهای مشترک می‌تواند یکی نباشد. ولی، بهتر است نام آن‌ها یکی باشد). به عنوان مثال، فیلد atID پیوند بین جداول Publishers و PubBook را برقرار می‌کند.

پیوند انواع مختلف دارد که عبارت‌اند از:

۱. پیوند ضربدری (Cross Join)
۲. پیوند متعادل (Equal Join)
۳. پیوند نامتعادل (NonEqual Join)
۴. پیوند درونی (Inner Join)
۵. پیوند بیرونی (Outer Join)

۱ – ۱۱ – ۲. پیوند ضربدری

در پیوند ضربدری بین دو جدول، هر رکورد جدول اول، با رکوردهای جدول دوم تکرار می‌شود. یعنی، تعداد رکوردهای خروجی برابر با تعداد رکوردهای جدول اول ضرب در تعداد رکوردهای جدول دوم است. این پیوند همان ضرب دکارتی بین دو جدول می‌باشد.

مثال ۴۴ – ۲. پرس‌وجویی که ضرب دکارتی جدول GroupBook و PubBook را نشان می‌دهد.

```
SELECT * FROM GroupBook, PubBook
```

	groupID	groupName	ISBN	pubID	Payment
1	01	برنامه نویسی	6009141302	01	6000000
2	01	برنامه نویسی	6009141340	01	7000000
3	01	برنامه نویسی	6009141388	01	6000000
4	01	برنامه نویسی	6009141395	01	6000000
5	01	برنامه نویسی	6009225422	01	8000000
6	01	برنامه نویسی	6009225439	01	5000000
7	01	برنامه نویسی	7009225431	01	3500000

Query executed successfully. | 1-VAIO\YOUSEF (10.0 RTM) | 1-VAIO\1 (54) | PublishDB | 00:00:00 | 40 rows

همان‌طور که در خروجی می‌بینید، ضرب دکارتی بین دو جدول GroupBook و PubBook انجام شده است.

منال ۴۵-۲. پرس‌وجویی که کتاب‌های با کد گروه '02' را در گروه کتاب ضرب می‌کند.

```
SELECT * FROM GroupBook AS g ,Books AS b
WHERE b.groupID = '02'
```

	groupID	groupName	ISBN	Title	Page	Price	editNo	printNo	groupID
1	01	برنامه نویسی	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
2	02	پایگاه داده	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
3	03	شبکه رایانه ای	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
4	04	فناوری اطلاعات	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
5	05	گرافیک رایانه ای	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02

این پرس‌وجو ضرب دکارتی با شرط θ را پیاده‌سازی کرده است.

۲-۱۱-۲. پیوند متعادل

پیوند متعادل، رایج‌ترین پیوند است که با استفاده از فیلد مشترک بین دو جدول انجام می‌شود. این پیوند همان ضرب دکارتی بین دو جدول است، به طوری که شرط برابری فیلد مشترک آن‌ها باید در ضرب دکارتی ذکر گردد. الحاق بین دو جدول به صورت زیر انجام می‌شود:

```
SELECT * FROM t1, t2
WHERE t1.relationField = t2.relationField
```

منال ۴۶-۲. پرس‌وجویی که اطلاعات نام کتاب و نام گروه را نمایش می‌دهد.

```
SELECT Title, groupName
FROM GroupBook AS g ,Books AS b
WHERE b.groupID = g.groupID
```

	Title	groupName
1	حل مسائل C++	برنامه نویسی
2	حل مسائل C#	برنامه نویسی
3	امنیت شبکه	شبکه رایانه ای
4	اصول طراحی پایگاه داده	پایگاه داده
5	اصول طراحی سیستم های شی گرا	برنامه نویسی
6	مدیریت استراتژیک	فناوری اطلاعات
7	رهیافت C++	برنامه نویسی

همان‌طور که در این پرس‌وجو می‌بینید، شرط WHERE، عبارت b.groupID = g.groupID است. (فیلد groupID، فیلد مشترک بین جداول Books و GroupBook می‌باشد).

منال ۴۷-۲. پرس‌وجویی که نام ناشر، نام خانوادگی مدیر انتشارات، نام کتاب و میزان حق نشر دریافت شده برای هر کتاب را نمایش می‌دهد.

```
SELECT pName,bFname, bLname,  Title, Payment
FROM Publishers p ,Books AS b, PubBook AS pu
WHERE b.ISBN  = pu.ISBN AND p.pubID = pu.pubID
```

	pName	bFname	bLname	Title	Payment
1	فناوری نوین	رمضان	عباس نژاد	حل مسائل C++	6000000
2	فناوری نوین	رمضان	عباس نژاد	حل مسائل C#	7000000
3	فناوری نوین	رمضان	عباس نژاد	امنیت شبکه	6000000
4	فناوری نوین	رمضان	عباس نژاد	اصول طراحی پایگاه داده	6000000
5	فناوری نوین	رمضان	عباس نژاد	اصول طراحی سیستم های شی گرا	8000000
6	فناوری نوین	رمضان	عباس نژاد	مدیریت استراتژیک	5000000
7	فناوری نوین	رمضان	عباس نژاد	رهیافت C++	3500000
8	دانش نگار	محمدرضا	حاتمی	رهیافت C++	3500000

همان‌طور که در این پرس‌وجو می‌بینید، برای برقراری ارتباط بین سه جدول، ارتباط بین جداول دو به دو با هم برقرار گردید. یعنی، ارتباط جداول Publishers و PubBook از طریق فیلد pubID برقرار گردید و پیوند جدول PubBook و Books، از طریق فیلد ISBN برقرار شد.

۳ – ۱۱ – ۲. پیوند نامتعادل

در پیوند متعادل دیدید که برای برقراری ارتباط از عملگر = در شرط WHERE استفاده شده است. اگر به جای عملگر =، از عملگرهای دیگر (نظیر =>، =<، >، < و <>) در شرط WHERE استفاده کنید، پیوند نامتعادل ایجاد خواهد شد.

مثال ۴۸ – ۲. پرس‌وجویی که دو جدول Books و GroupBook را از طریق عملگر > پیوند می‌زند.

```
SELECT * FROM Books AS b, GroupBook
WHERE b.groupID > GroupBook.groupID
```

	ISBN	Title	Page	Price	editNo	printNo	groupID	groupID	groupName
1	6009141388	امنیت شبکه	192	65000	1	1	03	01	برنامه نویسی
2	6009141388	امنیت شبکه	192	65000	1	1	03	02	پایگاه داده
3	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02	01	برنامه نویسی
4	6009225439	مدیریت استراتژیک	272	75000	1	1	04	01	برنامه نویسی
5	6009225439	مدیریت استراتژیک	272	75000	1	1	04	02	پایگاه داده
6	6009225439	مدیریت استراتژیک	272	75000	1	1	04	03	شبکه رایانه ای

۴ – ۱۱ – ۲. پیوند درونی

الحاق‌هایی که تاکنون ایجاد شده‌اند، شرط الحاق در بخش WHERE دستور SELECT آمده است. اگر در هنگام ایجاد پیوند، شرط معمولی نیز وجود داشته باشد، شرط پیوند با شرط معمولی، "**و**" منطقی (AND) می‌گردد. در این حالت، اگر تعداد شرط‌ها زیاد شود، تشخیص شرط الحاق (پیوند) و شرط معمولی مشکل خواهد بود. برای رفع این مشکل، پیوند درونی مطرح گردید. پیوند درونی به صورت زیر به کار می‌رود:

```
SELECT لیست فیلدها   FROM t1
INNER   JOIN  t2 ON شرط پیوند ۱
INNER   JOIN  t3 ON شرط پیوند ۲
...
WHERE شرط معمولی
```

مثال ۴۹ – ۲. پرس‌وجویی که نام، نام خانوادگی مؤلف، نام کتاب و میزان حق تألیف دریافتی برای هر کتاب را نمایش می‌دهد.

```
SELECT aFname, aLname, Title, Payment
FROM Authors AS a
INNER JOIN AutBook au ON  a.atID = au.atID
```

```
INNER JOIN Books b  ON au.ISBN = b.ISBN
```

	aFname	aLname	Title	Payment
1	رمضان	عباس نژاد	حل مسائل C++	6000000
2	رمضان	عباس نژاد	حل مسائل C#	6000000
3	رمضان	عباس نژاد	امنیت شبکه	6000000
4	رمضان	عباس نژاد	اصول طراحی پایگاه داده	5000000
5	باقر	رحیم پور	اصول طراحی پایگاه داده	5000000
6	رمضان	عباس نژاد	اصول طراحی سیستم های شی گرا	5000000
7	باقر	رحیم پور	اصول طراحی سیستم های شی گرا	5000000
8	سید حجت اله	جلیلی	مدیریت استراتژیک	6000000
9	رمضان	عباس نژاد	رهیافت C++	6000000

مثال ۵۰-۲. پرس‌وجویی که نام ناشر، نام کتاب و مبلغ حق نشر کتابی که حق نشر آن زیر ۴۰۰۰۰۰۰ ریال باشد را نمایش می‌دهد.

```
SELECT pName, Title, Payment
FROM Publishers  AS p
INNER JOIN PubBook  pu
   ON  p.pubID  = pu.pubID
INNER JOIN Books b
   ON pu.ISBN = b.ISBN
WHERE Payment < 4000000
```

	pName	Title	Payment
1	فناوری نوین	رهیافت C++	3500000
2	دانش نگار	رهیافت C++	3500000

مثال ۵۱-۲. پرس‌وجویی که لیست ناشرانی را بازیابی می‌کند که برای هیچ کتابی حق نشر بیش از ۴۰۰۰۰۰۰ ریال دریافت نکردند.

```
SELECT p.*
FROM Publishers  AS p
INNER JOIN PubBook  pu ON  p.pubID  = pu.pubID
WHERE p.pubID  NOT IN (SELECT pubID FROM PubBook  WHERE
        Payment > 4000000)
```

	pubID	pName	Tel	URL	ctyName	bFname	bLname	countBookPrint
1	02	دانش نگار	02166400220		تهران	محمد رضا	حاتمی	100

مثال ۵۲-۲. پرس‌وجویی که لیست مؤلفانی که برای هیچ کتابی حق تألیف بیش از ۵۰۰۰۰۰۰ ریال دریافت نکردند، را نمایش می‌دهد.

```
SELECT DISTINCT a.*
FROM Authors   AS a
INNER JOIN AutBook   at ON  a.atID  = at.atID
WHERE a.atID NOT IN (SELECT atID  FROM AutBook WHERE
       Payment > 5000000 )
```

	atID	aFname	aLname	Age	Ranking	Email	Mobile	sumPayment
1	03	باقر	رحیم پور	32	فوق لیسانس		NULL	10000000

۵-۱۱-۲. پیوند بیرونی

این نوع پیوند، کلیه رکوردهای موجود در یک جدول (حتی رکوردهایی که در جدول دیگر وجود ندارند) را بازیابی کرده، در خروجی می‌آورد. **الحاق بیرونی**[1] به صورت زیر به کار می‌رود:

```
SELECT لیست فیلدها FROM جدول ۱
{LEFT | RIGHT | FULL) OUTER JOIN جدول ۲ ON شرط پیوند
```

این پیوند سه نوع دارد که در زیر می‌بینید:

---

[1] .Outer Join

الحاق LEFT OUTER JOIN، کلیه رکوردهای جدول اول (جدول سمت چپ) را آورده و رکوردهایی از جدول دوم (جدول سمت راست) را نمایش می‌دهد که در شرط جلوی ON صدق کنند. اگر رکوردهایی در جدول اول باشند، ولی در شرط جلوی ON صدق نکنند، به جای فیلدهای جدول دوم NULL نمایش داده می‌شود.

مثال ۵۳-۲. پرس‌وجویی که پیوند بیرونی سمت چپ جدول گروه کتاب و کتاب را نمایش می‌دهد

```
SELECT *    FROM GroupBook AS g
LEFT OUTER JOIN Books b    ON  g.groupID = b.groupID
```

	groupID	groupName	ISBN	Title	Page	Price	editNo	printNo	gr
1	01	برنامه نویسی	6009141302	حل مسائل C++	216	57000	1	2	0
2	01	برنامه نویسی	6009141340	حل مسائل C#	256	62000	1	2	0
3	01	برنامه نویسی	6009225422	اصول طراحی سیستم های شی گرا	432	130000	1	1	0
4	01	برنامه نویسی	7009225431	رهیافت C++	320	43000	1	1	0
5	02	پایگاه داده	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	0
6	03	شبکه رایانه ای	6009141388	امنیت شبکه	192	65000	1	1	0
7	04	فناوری اطلاعات	6009225439	مدیریت استراتژیک	272	75000	1	1	0
8	05	گرافیک رایانه ای	NULL	NULL	NULL	NULL	NULL	NULL	N

همان‌طور که در خروجی می‌بینید، گروه کتابی که برای آن، هیچ کتابی در جدول Books وجود نداشته باشد، به جای فیلدهای آن (فیلدهای جدول Books) NULL آمده است.

الحاق RIGHT OUTER JOIN، کلیه رکوردهای جدول دوم (جدول سمت راست) و رکوردهایی از جدول اول را نمایش می‌دهد که در شرط ON صدق کنند. اگر رکوردهایی در جدول اول وجود داشته باشند که در شرط ON صدق نکنند، به جای فیلدهای جدول اول NULL قرار می‌گیرد.

مثال ۵۴-۲. پرس‌وجویی که پیوند بیرونی سمت راست جدول گروه کتاب و کتاب را نمایش می‌دهد (در این جا شرط پیوند > است).

```
SELECT *    FROM GroupBook AS g
RIGHT OUTER JOIN Books b    ON  g.groupID < b.groupID
```

	groupID	groupName	ISBN	Title	Page	Price	editNo	printNo	groupID
1	NULL	NULL	6009141302	حل مسائل C++	216	57000	1	2	01
2	NULL	NULL	6009141340	حل مسائل C#	256	62000	1	2	01
3	01	برنامه نویسی	6009141388	امنیت شبکه	192	65000	1	1	03
4	02	پایگاه داده	6009141388	امنیت شبکه	192	65000	1	1	03
5	01	برنامه نویسی	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
6	NULL	NULL	6009225422	اصول طراحی سیستم های شی گرا	432	130000	1	1	01
7	01	برنامه نویسی	6009225439	مدیریت استراتژیک	272	75000	1	1	04
8	02	پایگاه داده	6009225439	مدیریت استراتژیک	272	75000	1	1	04
9	03	شبکه رایانه ای	6009225439	مدیریت استراتژیک	272	75000	1	1	04
10	NULL	NULL	7009225431	رهیافت C++	320	43000	1	1	01

همان‌طور که در این خروجی می‌بینید، به جای فیلدهای جدول اول که در شرط صدق نمی‌کنند، NULL آمده است.

الحاق FULL OUTER JOIN، کلیه رکوردهای جدول اول و همه رکوردهای جدول دوم را با هم الحاق می‌کند. به جای فیلدهایی از جدول اول و دوم که در شرط صدق نمی‌کنند، مقدار NULL قرار می‌گیرد.

```
SELECT ISBN شابک , Title AS 'نام کتاب'
FROM Books
WHERE printNo = 2
```

	شابک	نام کتاب
1	6009141302	حل مسائل C++
2	6009141340	حل مسائل C#

۵ – ۲. مرتب سازی رکوردها

یکی از ویژگی‌های جالب دستور SELECT، امکان مرتب‌سازی رکوردهای جدول است. برای مرتب‌سازی رکوردها در دستور SELECT می‌توانید از گزینه ORDER BY استفاده کنید. برای تعیین نوع مرتب‌سازی در بخش ORDER BY می‌توانید از گزینه‌های DESC، برای مرتب سازی نزولی یا ASC، برای مرتب‌سازی صعودی استفاده کنید. اگر نوع مرتب‌سازی ذکر نگردد، به طور پیش‌فرض صعودی (ASC) در نظر گرفته می‌شود.

مثال ۲۰ – ۲. پرس‌وجویی که لیست مؤلفینی را نمایش می‌دهد که سن آن‌ها بیش از ۳۲ سال است (اطلاعات را بر اساس سن مرتب می‌کند و نمایش می‌دهد).

```
SELECT * FROM Authors WHERE Age > 32
ORDER BY Age
```

	atID	aFname	aLname	Age	Ranking	Email	Mobile	sumPayment
1	05	سید حجت اله	جلیلی	35	NULL	NULL	NULL	6000000
2	01	رمضان	عباس نژاد	42	فوق لیسانس	fanavarienoin@yahoo.com	NULL	28000000

همان‌طور که در این خروجی می‌بینید، نتیجه بر اساس سن به صورت صعودی مرتب گردید (نوع مرتب‌سازی ذکر نشده، پس صعودی انتخاب می‌شود).

مثال ۲۱ – ۲. پرس‌وجویی که لیست ناشرین را نمایش می‌دهد (اطلاعات را بر اساس تعداد کتاب‌های چاپ شده توسط آن ناشر به صورت نزولی مرتب می‌کند).

```
SELECT * FROM Publishers
ORDER BY countBookPrint DESC
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	02	دانش نگار	02166400220		تهران	محمدرضا	حاتمی	100
2	01	فناوری نوین	011122566687	fanavarienovin.net	بابل	رمضان	عباس نژاد	15
3	03	دانشگاه مازندران	01125226686		بابلسر	رضا	امینی	10

نکته: علاوه بر این به جای یک فیلد مرتب‌سازی می‌توان چند فیلد را برای مرتب‌سازی انتخاب نمود. برای این منظور بعد از گزینه ORDER BY فیلدها را با کاما از یکدیگر جدا کنید. در این صورت، چنانچه رکوردهایی وجود داشته باشند، که مقدار فیلد اول آن‌ها برابر باشد، بر اساس فیلد دوم مرتب‌سازی را انجام می‌دهد. اگر رکوردهایی وجود داشته باشند که مقدار فیلد اول و دوم مرتب‌سازی برابر داشته باشند، بر اساس فیلد سوم مرتب می‌نماید و این روند را ادامه می‌دهد.

مثال ۲۲ – ۲. پرس‌وجویی که لیست کتاب‌ها را بر اساس نوبت چاپ به صورت نزولی مرتب می‌کند. اگر نوبت چاپ چند کتاب یکی باشد، بر اساس نام کتاب به صورت صعودی مرتب می‌کند.

```
SELECT * FROM Books ORDER BY printNo DESC, Title ASC
```

	ISBN	Title	Page	Price	editNo	printNo	groupID
1	6009141340	حل مسائل C#	256	62000	1	2	01
2	6009141302	حل مسائل C++	216	57000	1	2	01
3	7009225431	رهیافت C++	320	43000	1	1	01
4	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
5	6009225422	اصول طراحی سیستم های شی گرا	432	130000	1	1	01
6	6009141388	امنیت شبکه	192	65000	1	1	03
7	6009225439	مدیریت استراتژیک	272	75000	1	1	04

۶- ۲. پرس‌وجوهای مرکب

این پرس‌وجوها از چندین پرس‌وجو تشکیل شده‌اند. عملگرهایی که برای ایجاد پرس‌وجوهای مرکب به کار می‌روند، عبارت‌اند از :

۵. عملگر UNION (همان عملگر $\cup$ جبر رابطه‌ای).

۶. عملگر INTERSECT (همان عملگر $\cap$ جبر رابطه‌ای).

۷. عملگر EXCEPT (همان عملگر $-$ جبر رابطه‌ای).

۱ - ۶ - ۲. عملگر UNION

این عملگر نتیجه دو یا چند دستور SQL را با هم ترکیب کرده، رکوردهای تکراری را فقط یک بار نمایش می‌دهد.

مثال ۲۳ - ۲. پرس‌وجویی که تمام افراد موجود در بانک اطلاعات انتشارات را نمایش می‌دهد.

در این پرس‌وجو، ابتدا مؤلفین را بازیابی می‌کنیم. سپس، رواسای انتشارات را بازیابی کرده، آن‌ها را با هم ترکیب می‌کنیم. یعنی:

```
SELECT aFname, aLname FROM Authors
UNION
SELECT bFname, bLname FROM Publishers
```

	aFname	aLname
1	باقر	رحیم پور
2	رضا	امینی
3	رمضان	عباس نژاد
4	سید حجت اله	جلیلی
5	محمد رضا	حاتمی
6	مرتضی	پویان
7	یوسف	عباس نژاد

همان‌طور که در این خروجی می‌بینید، نام و نام خانوادگی مؤلفین و رواسای انتشارات نمایش داده شده است.

۲ - ۶ - ۲. عملگر UNION ALL

این عملگر همانند عملگر UNION عمل می‌کند. با این تفاوت که تکرار رکوردها را حذف نمی‌کند.

مثال ۲۴ - ۲. پرس‌وجویی که کلیه افراد موجود در بانک اطلاعاتی انتشارات را نمایش می‌دهد (این پرس‌وجو، نام و نام خانوادگی تکراری را چند بار نمایش می‌دهد).

```
SELECT aFname, aLname FROM Authors
UNION ALL
SELECT bFname, bLname FROM Publishers
```