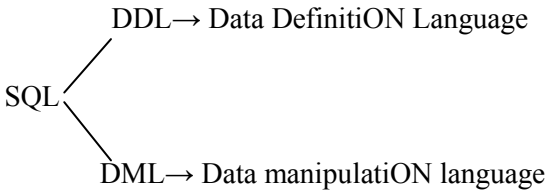
SQL

Structured Query Language

جزوه مهندس حمید رضا نیرومند
ویژه درس آزمایشگاه پایگاه داده
دانشگاه جامع علمی-کاربردی
http://niroomand.ir

زبان میزبان: به زبان برنامه‌نویسی اصلی که با آن کدنویسی می‌کنیم و از آنجا به دیتابیس متصل می‌شویم، Host Language یا زبان میزبان می‌گویند. مثل C# یا VB یا JAVA

تنها رابط بین زبان میزبان و DB، کدنویسی به زبان SQL است. بنابراین، یادگیری زبان SQL از اهمیت بسیار بالایی برخوردار است.



مهم‌ترین نوع داده‌ها:

نوع داده‌های عددی:

bigint	$-2^{32} \rightarrow 2^{32}$	8B
int	$-2^{16} \rightarrow 2^{16}$	4B
smallint	-32768→ 32768	2B
tinyint	0→ 255	

نوع داده‌های رشته‌ای (کاراکتر):

char (n)	1→4000	هر کاراکتر ۱ بایت فضا می‌گیرد
nchar (n)	1→4000	هر کاراکتر ۱ بایت فضا می‌گیرد و Unicode پشتیبانی می‌شود
varchar(n)		برای رشته هایی که طول متغیر دار ند
nvarchar(n)		برای رشته هایی که طول متغیر دارند و Unicode هستند

نکته: Char برای مقادیری که ثابت هستند استفاده می‌شود و اگر رشته به اندازه کل فیلد نباشد، انتهای آن کاراکتر سفید (خالی) ثبت می‌شود.
اگر رشته مورد نظر، انگلیسی بود varchar و اگر فارسی بود از نوع nvarchar استفاده می‌کنیم.

نوع داده‌های تاریخ:

DateTime	ex: 2011/12/22 09:29:30.255 میلی‌ثانیه را نیز ثبت می‌کند
SmallDateTime	ex: 2011/12/22 09:29:30 میلی‌ثانیه را ثبت نمی‌کند
timestamp	با هر بار تغییر یک سطر، لحظه آپدیت به طور خودکار، در این فیلد درج می‌شود

دستور ایجاد بانک:

```
CREATE DATABASE نام بانک
```

انتخاب بانک و اتصال به آن:

```
USE نام بانک;
```

دستور ایجاد Table:

ساختار کلی:

```
CREATE TABLE [database_name.]table_name
(
      <column name>    <datatype>
      [DEFAULT   <constant expresiON>]
      [IDENTITY  [(seed,increment)]]
      [NULL | NOT NULL]
      [<column   constraints>]
);
CREATE TABLE [نام بانک.] نام جدول
(
      <نام ستون> <نوع ستون>
      [DEFAULT          <مقدار پیش فرض>]
      [IDENTITY  [(افزایش در هر سطر , عدد شروع)]]
      [NULL | NOT NULL]
      [<قیود دیگر>]
);
```

تشریح اجزا:

- بعد از عبارت CREATE TABLE نام پایگاه داده‌ای که قرار است جدول ها در آن ایجاد شود نوشته می‌شود و نام دلخواه برای جدول در ادامه آن با یک نقطه از نام پایگاه جدا می‌شود.
- در داخل پرانتز باید لیست فیلد ها یا ستون ها همراه با صفات هر یک (مثلا نوع مقدار پیش فرض، خالی بودن یا نبودن و ....) درج می‌شود.
- ابتدا نام ستون و سپس نوع داده آن با یک فاصله جدا می‌شود.

- اگر لازم است یک ستون در صورت عدم مقداردهی از سوی کاربر یک مقدار پیش فرض به خود بگیرد بعد از کلمه کلیدی DEFAULT آن مقدار را درج می‌کنیم.
- IDENTITY مشخص می‌کند که یک ستون مثل حالت AutoNumber در Access باشد یعنی سطر اول یک عدد به خود بگیرد و سطرهای بعدی به صورت افزایشی (Increment) عددی بیشتر بگیرد در اینجا می‌توان عدد آغازین را به جای Seed و مقدار افزایش در هر سطر را به جای Increment قرار داد.

اگر فیلدی می‌تواند هیچ مقدار (NULL) باشد می‌توان کلمه کلیدی NULL را در بین نوشت و اگر نباید هیچ مقدار باشد NOT NULL درج شود.

قیود مربوط به جدول:

۱- قیود موجودیت Entity constraints

۲- قیود دامنه Domain constraints

۳- قیود جامعیت ارجاعی Referential constraints

۱-منظور از قیود موجودیت همان تعریف کلید اصلی Primary key)) برای جدول است.

۲-منظور از قیود دامنه این است که به طور مثال یک ستون مقداری بین ۲عدد خاص یا از بین چند عبارت خاص داشته باشد.

۳-منظور از قیود جامعیت ارجاعی تعریف کلید خارجی برای جدول است.

نکته: بهتر است قیدهایی که برای یک جدول در نظر می‌گیریم نامگذاری شوند تا بعدا راحت تر قابل تغییر یا حذف باشند.

مهم‌ترین قیود:

- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- CHECK
- DEFAULT

قید PRIMARY:

برای تعیین یک فیلد به عنوان کلید اصلی (وکلید خارجی) دو روش وجود دارد:

- **روش اول:** تعیین یک فیلد به عنوان کلید اصلی در هنگام تعریف و ایجاد جدول، در این صورت در سمت راست تعریف ستون مورد نظر کلمه کلیدی PRIMARY KEY درج می‌شود.

```
CREATE TABLE Books
(
	BID int IDENTITY NOT NULL PRIMARY KEY,
	BName nvarchar(100) NOT NULL
);
```

- **روش دوم:** تعیین یک ستون به عنوان کلید اصلی برای جدول که قبلا ایجاد شده است.

نکته: شاید نیاز باشد جدول یک بار حذف شود و دوباره ایجاد شود.

برای تعریف قید کلید اصلی به صورت زیر عمل می‌کنیم.

مثال)

```
ALTER TABLE Books
ADD CONSTRAINT pk_Booksid
PRIMARY KEY(id);
```

مثال)

جدول BOOKS در بانک Library با ستون های زیر را ایجاد کنید.

BID → int AutoNumber

Bcode → nvarchar (10)

BName

Author

Translator

Publisher

```
CREATE TABLE Books
(
	BID int IDENTITY NOT NULL PRIMARY KEY,
	BCode nvarchar(10) NOT NULL,
	BName nvarchar(100) NOT NULL,
	Author nvarchar(80) NULL,
	Translator nvarchar(80) NULL,
	Publisher nvarchar(255) NULL
);
```

قید FOREIGN KEY:

- روش اول: هنگام تعریف جدول

```
CREATE TABLE Library.lending
      (
            id int IDENTITY PRIMARY KEY,
            BID int NOT NULL
                  FOREIGN KEY REFERENCES Books(BID),
            MID int NOT NULL
                  FOREIGN KEY REFERENCES Member(MID),
            lendingdate smalldatetime,
            returndate smalldatetime
   );
```

- روش دوم:

```
ALTER TABLE Lending
ADD CONSTRAINT fk_lending_bid
FOREIGN KEY (BID)
REFERENCES books(BID);
```

یک نمونه کد پیچیده تر:جدول رابطه بین سفارش و کالاهای داخل سفارش

```
USE order;
CREATE TABLE OrderDetails
(
      OrID int NOT NULL,
      PrID int NOT NULL,
      Qty int NOT NULL,
      CONSTRAINT PKOrder_Details
      PRIMARY KEY (OrID,PrID),
      CONSTRAINT FK_OID
      FOREIGN KEY (OrID)
      REFERENCES Orders(OID)
      ON UPDATE NO ACTION
      ON DELETE CASCADE,
      CONSTRAINT FK_PID
      FOREIGN KEY (PrID)
      REFERENCES Products(PID)
      ON UPDATE NO ACTION
      ON DELETE CASCADE
);
```

ویرایش بانک:

با استفاده از دستور ALTER می‌توان ساختار جدول و بانک را تغییر داد.

مهم ترین ویرایش در مورد بانک تغییر نام آن است، برای این کار از ساختار زیر استفاده می‌کنیم.

```
USE library;
ALTER DATABASE <database_name>
MODIFY NAME=<new_dbname>
```

برخی تغییرات که ممکن است نیاز باشد روی جدول انجام شود عبارتند از:

- تغییر نام جدول
- تغییر نام ستون
- افزودن ستون جدید
- تغییر نوع یک ستون
- افزودن قید

تغییر نام جدول:

```
go
sp_rename <table_old_name>,<table_new_name>;
```

نام جدول Books را به Book تغییر دهید.

```
go
sp_rename Books,Book;
```

تغییر نام یک ستون:

```
go
sp_rename 'table_name[old_column_name]','[new_column_name]','column';
```

فیلد BID از جدول Books را به id تغییر دهید.

```
go
sp_rename 'books.bid','id','column';
```

افزودن یک ستون جدید به جدول:

```
ALTER TABLE <table_name>
ADD
<column_name> <data_type>
[DEFAULT <DEFAULT_value>]
[IDENTITY]
[NULL|NOT NULL];
```

مثال)

فیلدی به نام Place را به جدول Books اضافه کنید.

```
ALTER TABLE books
ADD
place nvarchar(255)
DEFAULT ' قفسه اول '
NOT NULL;
```

تغییر نوع ستون:

```
ALTER TABLE <table_name>
ALTER COLUMN <column_name>
<new_data_type> [NULL|NOT NULL]
```

نوع فیلد BName از جدول Books را که هم اکنون varchar(255) NOT NULL است را به نوع varchar(100) NULL تغییر دهید.

```
ALTER TABLE books
ALTER COLUMN bname
varchar(100) NULL;
```

برای حذف یک یا چند ستون از جدول:

```
ALTER TABLE <table_name>
DROP COLUMN <column_name>,...,n;
```

مثال)

فیلد Place را از جدول Books حذف کنید.

```
ALTER TABLE books
DROP COLUMN place;
```

حذف یک قید از جدول:

```
ALTER TABLE <table_name>
DROP CONSTRAINT <constraint_name>;
```

حذف جدول یا بانک:

برای حذف یک شی (جدول یا بانک) می‌توان از دستور DROP استفاده کرد.

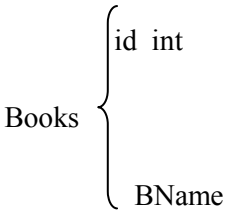
نکته: برای انجام برخی عملیات در روی جدول Books سیستم هشدار می‌دهد که ابتدا جدول Books را DROP کنید، سپس ایجاد کنید و در نهایت عملیات را ادامه دهید.
دستور مربوط به DROP کردن جدول Books را درج کنید.

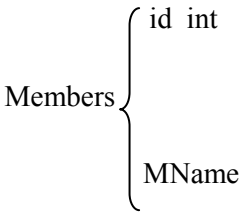
```
USE library;
DROP TABLE books;
```

دیتابیس Library را DROP کنید.

```
DROP DATABASE library;
```

جداول Books و Members با برخی قیود روی فیلدها:






```
CREATE DATABASE library;
USE library;
CREATE TABLE books
  (
     id int IDENTITY NOT NULL PRIMARY KEY,
     bname nvarchar(255) NOT NULL
  );

CREATE TABLE members
 (
     id int IDENTITY NOT NULL PRIMARY KEY,
     bname nvarchar(255) NOT NULL
 );
```

برای حذف کلید اصلی از ساختار زیر استفاده می‌کنیم.

```
ALTER TABLE books
DROP CONSTRAINT pk_books_7c8480ae;
ALTER TABLE books
DROP constraints pk__books__id
```

افزودن رکورد به جدول:

```
INSERT [INTO] <table_name> [(column_list)]
VALUES(data_values);
```

نکته: اگر صفتی از نوع AutoNumber باشد، هنگام درج، صفت مورد نظر را در بین نام ستون ها نمی‌نویسیم.

```
USE library;
INSERT INTO books(bname) VALUES ('leavating c#')
```

مثال) کتابی به نام "Learning C#" با نام نویسنده A.J HogE و مترجم Ali Akbari و انتشارات Ghoghnoos را به جدول Books اضافه کنید.

در مرحله اول باید ستون Author، Translator، Publisher را به جدول Books اضافه کنیم.

```
ALTER TABLE books
ADD
Author nvarchar(100)NOT NULL,
Translator nvarchar(100)NOT NULL,
Publisher nvarchar(255)NOT NULL;
```

بعد رکورد را اضافه می‌کنیم

```
INSERT INTO books(BName, Author, Translator, Publisher)
VALUES ('leavating c#','A.J Hoge','Ali Akbari','Ghoghnoos');
```

دستور Select:

برای پرس و جو از یک جدول از دستور Select استفاده می‌شود که قسمت عمده‌ای از دستورات SQL را تشکیل می‌دهد.

ابتدا بانک Library را با سه جدول زیر در نظر بگیرید.

Books

BID	BName	Author	Publisher	Price	Nums
۱	Internet	Ulman	Berekely	۱۸۰۰۰	۲
۲	Access	Elmarsi	Wrox	۲۰۰۰۰	۳
۳	Database	C.J.Date	MIT	۲۵۰۰۰	۱
۴	Network	Elmarsi	MIT	۱۸۰۰۰	۲
۵	Microsoft Access	C.J.Date	Press	۱۰۰۰۰	۱۰
۶	Uml	Elmarsi	Berekely	۵۰۰۰	۵

Members

MID	MFName	MLName	MFaName	BrithDate	MStatus	MGender
۱	Hamidreza	Niroomand	Mostafa	1986	1	1
۲	Yahya	Ghanbari	Hossain	1980	2	1
۳	Ali	Radmard	Hojjat	1970	2	1
۴	Ali	Akbari	Reza	2000	1	1
۵	Maryam	Rezaee	Mahmod	2003	1	0

Lending

LID	BID	MID	L_Date	R_Date
1	2	3	2010/3/3	2010/3/10
2	1	4	2010/3/10	
3	5	5	2010/3/12	2010/3/17
4	3	3	2010/3/3	
5	2	1	2010/3/20	
6	5	1	2010/3/20	

دستور ایجاد بانک و جداول:

```
CREATE DATABASE Library;

USE Library;
CREATE TABLE Books
	(
		BID int IDENTITY NOT NULL PRIMARY KEY,
		BName nvarchar(255) NOT NULL,
		Author nvarchar(80) NOT NULL,
		Publisher nvarchar(255) NULL,
		Price int NULL,
		Nums int NOT NULL
	);

CREATE TABLE Members
	(
		MID int IDENTITY NOT NULL PRIMARY KEY,
		MFName nvarchar(30) NOT NULL,
		MLName nvarchar(50) NOT NULL,
		MFaName nvarchar(30) NOT NULL,
		MAge smallint NULL,
		MStatus smallint NOT NULL DEFAULT 1,
		MGender smallint NOT NULL
	);

CREATE TABLE Lending
	(
		id int IDENTITY NOT NULL PRIMARY KEY,
		BID int NOT NULL FOREIGN KEY REFERENCES Books(BID),
		MID int NOT NULL FOREIGN KEY REFERENCES Members(MID),
		L_DATE varchar(9) NOT NULL,
		R_Date varchar(9) NULL
	);
```

دستور اضافه کردن رکوردها:

```
INSERT INTO Books (BName, Author, Publisher, Price, Nums) VALUES
('Internet', 'Ulman', 'Berekely', 18000, 2);

INSERT INTO Books (BName, Author, Publisher, Price, Nums) VALUES ('Access',
'Elmarsi', 'Wrox', 20000, 3);

INSERT INTO Books (BName, Author, Publisher, Price, Nums) VALUES
('Database', 'C.J.Date', 'MIT', 25000, 1);

INSERT INTO Books (BName, Author, Publisher, Price, Nums) VALUES
('Network', 'Elmarsi', 'MIT', 18000, 2);

INSERT INTO Books (BName, Author, Publisher, Price, Nums) VALUES
('Microsoft Access', 'C.J.Date', 'Press', 10000, 10);

INSERT INTO Books (BName, Author, Publisher, Price, Nums) VALUES ('UML',
'Elmarsi', 'Berekely', 5000, 5);


INSERT INTO Members (MFName, MLName, MFaName, MAge, MStatus, MGender)
VALUES ('Hamid Reza', 'Niroomand', 'Mostafa', 24, 1, 1);

INSERT INTO Members (MFName, MLName, MFaName, MAge, MStatus, MGender)
VALUES ('Yahya', 'Ghanbari', 'Hossain', 30, 2, 1);

INSERT INTO Members (MFName, MLName, MFaName, MAge, MStatus, MGender)
VALUES ('Ali', 'Radmard', 'Hojjat', 31, 2, 1);

INSERT INTO Members (MFName, MLName, MFaName, MAge, MStatus, MGender)
VALUES ('Ali', 'Akbari', 'Reza', 22, 1, 1);

INSERT INTO Members (MFName, MLName, MFaName, MAge, MStatus, MGender)
VALUES ('Maryam', 'Rezaee', 'Mahmood', 20, 1, 0);


INSERT INTO Lending (BID, MID, L_Date, R_Date) VALUES
(2,3,'2010/3/3','2010/3/10');

INSERT INTO Lending (BID, MID, L_Date, R_Date) VALUES (1,4,'2010/3/10',
'');

INSERT INTO Lending (BID, MID, L_Date, R_Date) VALUES
(5,5,'2010/3/12','2010/3/17');

INSERT INTO Lending (BID, MID, L_Date, R_Date) VALUES (3,3,'2010/3/3','');

INSERT INTO Lending (BID, MID, L_Date, R_Date) VALUES (2,1,'2010/3/20','');

INSERT INTO Lending (BID, MID, L_Date, R_Date) VALUES (5,1,'2010/3/20','');
```

ساختار کلی SELECT:

```
SELECT <column_list>
[FROM <source tabel(s)>]
[WHERE <restrictive cONditiON>]
[GROUP BY <column name>]
[HAVING <restrictive cONditiON>]
[ORDER BY <column list>]
```

قسمت FROM :

این قسمت مشخص می‌کند که دستور SELECT بر روی چه جدول یا جداولی اجرا شود.

نکته:اگر به جای لیست ستون ها علامت(*) درج شود یعنی همه ستون ها و این همان عملگر SELECT (σ) در جبر رابطه‌ای است.

مثال)

لیست تمام کتاب‌های کتابخانه را نمایش دهید.

```
USE library;
SELECT * FROM books;
```

نکته ۱: اگر دو تا جدول بنویسیم همان ضرب دکارتی می‌شود.

نکته ۲:اگر به جای (*) نام چند ستون درج شود (و هیچ شرطی بیان نشود) خروجی این دستور شبیه عملگر Π در جبر رابطه‌ای است.

مثال)

نام کوچک تمام اعضا را لیست کنید.

```
USE library;
SELECT members.mfname FROM members;
```

نام کوچک و نام بزرگ تمام اعضا را لیست کنید.

```
USE library;
SELECT members.mfname, members.mLname FROM members;
```

برای تعیین یک شرط بر روی انتخاب (SELECT) از عبارت WHERE استفاده می‌شود و شرط لازم بعد از آن بیان می‌شود.

عملگرهای قابل استفاده در قسمت WHERE :

عملگر های مقایسه‌ای: =, <>, =<, >=, <, >, = !, <>!,<!

عملگر های منطقی: NOT, AND, OR

عملگر BETWEEN.بر ای انتخاب بین چند گزینه

عملگر N برای نمایش مقادیر از بین چند مقدار

عملگرهای مقایسه‌ای و منطقی:

لیست اعضایی را نمایش دهید که نام کوچک آنها Ali است.

```
USE library;
SELECT * FROM members WHERE members.mfname='ali' AND members.mage>25;
```

لیست کتابهایی را نمایش دهید که قیمت آن ها بیش از ۱۵۰۰۰ تومان است.

```
USE library;
SELECT * FROM Books WHERE Books.Price>15000;
```

نام خانوادگی اعضایی را نمایش دهید که بیش از ۲۵ سال سن دارند.

```
USE library;
SELECT MLName FROM members WHERE members.mage>25;
```

نام خانوادگی اعضایی را که ۲۴ ساله نباشند را لیست کنید.

```
USE library;
SELECT MLName FROM members WHERE members.mage!=25;
```

لیست اعضایی که نامشان برابر Ali و سن آنها بزرگتر از ۲۵ سال باشد را نمایش دهید.

```
USE library;
SELECT * FROM members WHERE members.mfname!='ali' AND members.mage!>25;
```

لیست اعضایی را نمایش دهید که نام آنها حمیدرضا یا علی باشد.

```
USE library;
SELECT * FROM members WHERE members.mfname='ali' AND members.mfname='hamid
reza';
```

لیست اعضایی را نمایش دهید که نام آنها Ali نباشد و سن آنها از ۲۵ بیشتر نباشد.

```
USE library;
SELECT * FROM members WHERE NOT(members.mfname='ali'AND members.mage>25);
```

عملگر BETWEEN :

لیست کتاب هایی که قیمت آنها بین ۲۰۰۰۰ تا ۳۰۰۰۰ باشد را نمایش دهید.

```
USE library;
SELECT * FROM books WHERE books.price BETWEEN 20000 AND 30000;
```

لیست اعضایی که سن آنها ۲۵ تا ۳۵ سال باشد را لیست کنید.

```
USE library;
SELECT * FROM Members WHERE Members.Mage BETWEEN 25 AND 35;
```

نام کتاب هایی که بین ۴ تا ۷ است را لیست کنید.

```
USE library;
SELECT * FROM books WHERE books.nums BETWEEN 4 AND 7;
```

نکته: عملگر BETWEEN مساوی را هم می‌آورد.

عملگر IN:

لیست کتاب هایی را نمایش دهید که انتشارات آنها یکی از انتشارات زیر باشد.

Wrox, Harvard, MIT, Lynda

```
USE library;
SELECT * FROM books WHERE books.publisher
IN('wrox','harvard','mit','lynda');
```

عبارت ORDER BY(مرتب کردن):

اگر بعد از این عبارت نام یک ستون بیایید سطح های خروجی نسبت به آن ستون مرتب می‌شوند .

مثال)

لیست اعضا مرتب شده بر اساس سن آنها نمایش دهید.

```
SELECT *FROM members ORDER BY members.mag;
SELECT *FROM members ORDER BY members.mag desc;
```

لیست اعضا مرتب شده بر اساس سن آنها نمایش دهید.

```
SELECT * FROM members ORDER BY members.mfname,mlname;
```

نام کتاب ها بر اساس حروف الفبا مرتب کنید.

```
SELECT bname FROM books ORDER BY books.bname;
```

عبارت GROUP BY:

این عبارت در حقیقت باعث می‌شود مقادیر تکراری در یک ستون یکبار نوشته شود.

مثال) بانک Lending بر اساس BID گروه بندی کنید.

```
SELECT bid FROM lending GROUP BY  bid;
```

توابع جمعی:

همراه با عملگر Group by می‌توان از توابع جمعی نیز استفاده کرد.

- تابع AVG()، میانگین هر گروه را محاسبه می‌کند

مثال)

فرض کنید می‌خواهیم میانگین قیمت کتاب برای هر نویسنده را در جدول Books بیاوریم.

```
SELECT author,AVG(price)FROM books GROUP BY author;
```

میانگین سن اعضا را نمایش دهید.

```
SELECT AVG(mage)FROM members;
```

- تابع : MAX و MIN

مقدار کمینه و بیشینه را نشان می‌دهد.

لیست نویسندگان کتابخانه همراه با کمترین قیمت کتاب آنها

```
SELECT author.MIN(price)FROM books GROUP BY author;
```

در جدول Books طبیعتا ممکن است یک نویسنده چندین کتاب با قیمت های مختلف داشته باشد نام نویسنده وبیشترین وکمترین قیمت کتابهای که نوشته است را لیست کنید.

```
SELECT author, MIN (price), MAX(price) FROM books GROUP BY author;
```

- Count :

این تابع تعداد سطر ها در یک Query را می‌شمارد .

دو روش استفاده وجود دارد:

۱- Count(*) : مقادیر NULL را می‌شمارد (همه سطرها)

۲- Count(column_name): مقادیر NULL در این ستون را نادیده می‌گیرد

مثال)

تعداد سطر ها ی جدول Books را نمایش دهید.

```
SELECT COUNT(*)FROM books;
```

تعداد اخذ های که برگردانده شده اند را لیست کنید .

```
SELECT COUNT(r_date)FROM lending WHERE r_date!='';
```

- Sum :

مجموعه مقادیر یک ستون را نمایش می‌دهد.

مثال)

تعداد کل کتابهای کتابخانه چند تا است.

```
SELECT SUM(nums)FROM books;
```

تعداد کتابهای Elmarsi در کتابخانه چند تاست.

```
SELECT SUM(nums)FROM books WHERE author='elmarsi';
```

ارزش کل کتابهای کتابخانه چند است .

```
SELECT SUM(price*nums) FROM books;
```

نام ، قیمت، تعداد کتاب نمایش دهید و در یک ستون ضرب قیمت و تعداد را نشان دهید.

```
SELECT bname, Price, nums, price*nums FROM books;
```

قسمت Having:

- حتما با Group by می‌آید.
- پس از گروه بندی روی گروهها اعمال می‌شود.

مثال)

لیست نویسندگانی را نمایش دهید که مجموعه کتابهای آنها بیش از ۱۰ کتاب باشد.

```
SELECT  author  FROM  books  GROUP BY  author  HAVING SUM(nums)>10;
```

اسم نویسندگانی که ارزش مجموعه کتابهای آنها بیش از صدهزار تومان باشد.

```
SELECT  author  FROM  books  GROUP BY  author  HAVING
SUM(nums*price)>100000;
```

گزاره های DISTINCT و ALL :

مقادیر تکراری را نمایش می‌دهد.

گزاره DISTINCT مقادیر تکراری را نمایش نمی دهد.

مثال)

لیست نویسندگان کتابخانه را لیست کنید.

```
SELECT  DISTINCT  author  FROM  books;
```

گزاره ALL به طور پیش فرض برای SELECT در نظر گرفته می‌شود.

JOIN:

بعد از نرمال سازی، جداول تجزیه خواهند شد برای خروجی گرفتن باید جداول با هم در اصطلاح JOIN شوند.

انواع JOIN :

چیزی که در همه JOIN ها مشترک است آنها باید یک رکورد را از یک جدول با یک یا چند رکورد از جدول دیگر تطبیق می‌دهند تا یک رکورد را که در حقیقت ترکیبی از آن رکورد هاست ایجاد کند.

INNER JOIN :

پرکاربرترین نوع JOIN است که بر اساس یک فیلد مشترک معمولا((فیلد خارجی)) با هم ترکیب می‌کند.

ساختار کلی:

```
SELECT <SELECT_list>
FROM <first_table>
[<join type>]<SELECT table>
[ON <join cONditiON>]
```

مثال)

لیست کتاب‌هایی که حداقل یکبار قرض گرفته شده‌اند را نمایش دهید.

```
SELECT*FROM lending INner join books ON lending.bid=books.bid;
```

فقط نام یکتای کتاب هایی که حداقل یکبار قرض گرفته شوند.

```
SELECT DISTINCT books.bname FROM lending INner join books ON
lending.bid=books.bid;
```

افزودن جدول سوم به پیوند

مثال)

مشخص کنید که چه اعضایی (با مشخصات کامل) چه کتابهایی را تا کنون قرض گرفته اند.

```
SELECT*FROM lending JOIN books ON lending.bid=books.bid join members ON
lending.mid=members.mid;
```

لیست کتابهایی که قیمت آنها از میانگین قیمت کل کتابها بیشتر است.

```
SELECT*FROM books WHERE books.price>(SELECT AVG(price)FROM books);
```

لیست اعضایی را نمایش دهید که mid آنها برابر با یکی از تعداد کتابهای (num) باشد.

```
SELECT*FROM members WHERE members.mid IN(SELECT nums FROM books);
```

لیست اعضایی را نمایش دهید که mid آنها از روی شماره تعداد کتابهاnums بدست می‌آید واین تعداد برابر یکی از شماره کتابهاbid است

```
SELECT*FROM members WHERE members.mid IN(SELECT nums FROM books WHERE
books.nums IN(SELECT books.bid FROM books);
```

اسم نویسندگان را نمایش دهید که میانگین قیمت کتابهای آنها بیشتر از میانگین قیمت کل کتابها باشد.

```
SELECT books.author,AVG(books,price)FROM books GROUP BY author HAVING
AVG(price)>(SELECT AVG(price)FROM books);
```

قید UNIQUE:

بر روی هر فیلدی که اعمال شود باعث می‌شود مقدار فیلد در رکوردهای مختلف یکتا(unique) باشد دقیقا همان کار کلید اصلی را انجام می‌دهد.

دو روش برای اعمال این قید وجود دارد :

- روش اول: هنگام تعریف جدول در این حالت کلمهunique در انتهای تعریف فیلد نوشته می‌شود.

```
CREATE TABLE books
(
bid int INENTITY PRIMARY KEY,
bname nvarchar(255)UNIQ UE,
);
```

- روش دوم: بعد از تعریف جدول(بر روی جدولی که قبلا ایجاد شده است)

```
ALTER TABLE boooks
ADD constrant UNIQUE bname
UNIQUE (bname);
```

قید CHECK:

قبل از Insert کردن یک رکورد ابتدا شرط مربوط به check به روی یک فیلد اعمال شده است چک می‌شود اگر شرط برقرار بود رکورد درج می‌شود دو روش برای اعمال این قید وجود دارد.

- روش اول: هنگام تعریف جدول

مثال) با استفاده از قید check طوری تنظیم کنید که تاریخ قرض گرفتن یک کتاب نتواند زودتر از تاریخ جاری باشد

```
CREATE TABLE lending(
l_date smalldatetime
CHECK l_date<=getdate()
```

تابع getdate() در SQL تاریخ فعلی را به دست می‌آورد.

- روش دوم: بعد از ایجاد جدول

```
ALTER TABLE lending
ADD CONSTRAINT check_l_date
CHECK(l_date<=getdate())
```

قیدDEFAULT:

مقدار پیش فرض یک فیلد در صورتی که در هنگام درج مقداری برای آن تعریف نشود را مشخص می‌کند.

- روش اول:هنگام تعریف جدول قبلا توضیح داده شده است
- روش دوم:جدول lending را طوری تنظیم کنید که اگر کتابدار تاریخ اخذ را درج نکرده تاریخ جاری به طور پیش فرض DEFAULT درج شود.

```
ALTER TABLE lending

ADD CONSTRAINT defult_l_date

DEFAULT getdate()for l_date;
```

یکی ازکتابدارهای کتابخانه فقط مسئول ثبت قرض دادن کتاب در بانک است بنابراین نیازی به داشتن سن آدرس، وضعیت تأهل و نام پدر اعضا ندارد یک view از جدول members برای این کتابدار بسازد.

```
CREATE VIEW v1(mid,mfname,mlname,mgender)

AS SELECT(mid,mfname,mlastname,mgender)FROM members;
```

نام ونام خانوادگی یکی از نویسندگانTONebown بوده که به اشتباه در جدول Ulman،books ثبت شده یک قطعه کد آنها را اصلاح کنید؟

```
UPDATE Books
SET Author='TONenbown'
WHERE Author='Ulman';
```