پایگاه داده‌ها

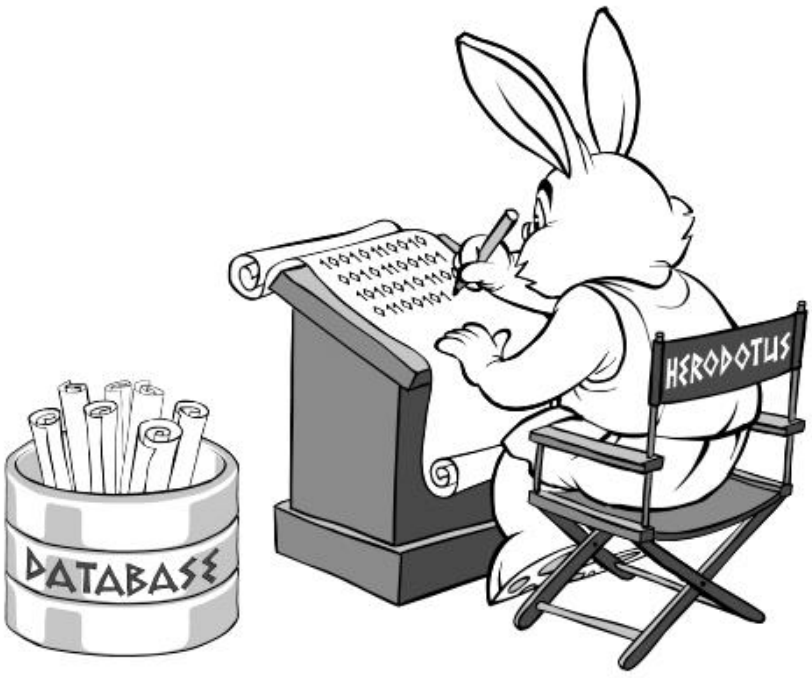
مدرس: میلاد وزان

دانشگاه شهید بهشتی - دانشکده ریاضی - گروه آموزشی آمار

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

https://dbsbu.github.io

فصل ۴: زبان پرسمان ساختار یافته



زبان پرسمان ساختار یافته (Structured Query Language)

زبان پُرسمان (پرس‌وجو یا کوئری) ساختاریافته که به اختصار SQL خوانده می‌شود، برای تعامل داده‌ها با پایگاه داده رابطه‌ای طراحی شده است و به توسعه‌دهندگان کمک می‌کند تا با انجام کارهایی مانند **کوئری**، **به‌روزرسانی**، **حذف** و **مدیریت داده‌ها**، کار را به شیوه‌ای کارآمد با داده‌ها انجام دهند. **داده‌ها** در پایگاه داده رابطه‌ای در **جداول ساختاریافته** با روابط تعریف شده ذخیره می‌شوند.

اهداف SQL

SQL زبان استانداردی است که برای تعامل با پایگاه‌های داده رابطه‌ای استفاده می‌شود. این یک رویکرد سیستماتیک و سازمان یافته برای ایجاد، مدیریت و دستکاری داده‌ها از طریق یک قالب ساختار یافته ارائه می‌دهد.

۱. ذخیره‌سازی و سازماندهی داده‌ها

SQL در تعریف و مدیریتِ ذخیره‌سازیِ داده‌ها در پایگاه داده‌های رابطه‌ای استفاده می‌شود. **داده‌ها** در قالب **جدولِ ساختاریافته** (یا ردیف‌ها و ستون‌ها) برای **مدیریت**، **بازیابی** و **دستکاریِ آسان**، ذخیره می‌شوند.

۲. بازیابی داده‌ها

SQL عمدتا برای بازیابی و استخراج داده‌هایِ خاص، بر اساس شرایط تعریف شده توسط کاربر در مجموعه داده‌های بزرگ در گزارش‌گیری برای هوش تجاری استفاده می‌شود. این زبان قدرتمند به سازمان‌ها امکان می‌دهد تا داده‌های پیچیده را از پایگاه‌های داده عظیم استخراج کنند و به صورت گزارش‌های تحلیلی درآورند. در ادامه، به برخی از جنبه‌های مهم استفاده از SQL در هوش تجاری اشاره می‌کنیم.

استفاده از SQL در هوش تجاری:

- **ایجاد گزارش‌های تحلیلی:** SQL به شرکت‌ها امکان می‌دهد تا گزارش‌های سفارشی‌سازی ایجاد کنند که به بررسی عملکرد فروش، تحلیل رفتار مشتریان، و ارزیابی موفقیت کمپین‌های بازاریابی کمک می‌کند. این گزارش‌ها با استفاده از دستورات SQL مانند SELECT و JOIN ایجاد می‌شوند که امکان استخراج داده‌های مرتبط از جداول مختلف را فراهم می‌کنند. این گزارش‌ها به مدیران کمک می‌کنند تا به سرعت به داده‌های مورد نیاز دسترسی پیدا کنند و تصمیمات آگاهانه بگیرند.
- **پردازش داده‌های بزرگ:** ابزارهای پیشرفته‌ای مانند Hadoop و Spark با SQL سازگارند و امکان تحلیل داده‌های بزرگ را فراهم می‌کنند. این ابزارها به سازمان‌ها کمک می‌کنند تا داده‌های حجیم را پردازش کرده و بینش‌های ارزشمندی از داده‌ها استخراج کنند. با استفاده از این ابزارها، سازمان‌ها می‌توانند داده‌های پیچیده را تجزیه و تحلیل کنند و روندهای پنهان را شناسایی کنند.
- **استفاده از ابزارهای هوش تجاری:** ابزارهای هوش تجاری مانند Tableau و Power BI از SQL برای تحلیل و مصورسازی داده‌ها استفاده می‌کنند. این ابزارها به کاربران امکان می‌دهند تا داده‌ها را تحلیل کرده، روندها را

شناسایی کنند، و بینش‌های ارزشمند ایجاد کنند. با استفاده از این ابزارها، سازمان‌ها می‌توانند داده‌های خود را به صورت داشبوردهای تعاملی و گزارش‌های تحلیلی درآورند و به تصمیم‌گیری بهتر کمک کنند.

۳. دستکاری داده‌ها

SQL قابلیت‌های قوی برای دستکاری داده‌ها در پایگاه‌های داده رابطه‌ای فراهم می‌کند. عملیات اولیه برای دستکاری داده‌ها شامل **درج**، **به‌روزرسانی** و **حذف رکوردها** از جدول است.

مزایای دستکاری داده‌ها در SQL

- **انعطاف‌پذیری:** SQL به کاربران اجازه می‌دهد تا حجم زیادی از داده‌ها را به طور موثر دستکاری کنند، خواه به ردیف‌های جدید اضافه کنند یا ردیف‌های موجود را اصلاح کنند.
- **دقت:** استفاده از عبارت‌هایی مانند WHERE تضمین می‌کند که عملیات فقط برای سوابق مربوط اعمال می‌شود.
- **سازگاری:** SQL یکپارچگی داده‌ها را با رعایت قوانین پایگاه داده رابطه‌ای در حین دستکاری حفظ می‌کند.

۴. یکپارچگی داده‌ها

SQL از محدودیت‌های یکپارچگی داده‌ها برای مدیریت دقت و صحت داده‌ها استفاده می‌کند. این محدودیت‌ها از **افزودن داده‌های نامعتبر** و **تکراری** جلوگیری می‌کنند. در ادامه، به توضیح هر یک از این محدودیت‌ها می‌پردازیم:

- **کلید اصلی (PRIMARY KEY)** اطمینان می‌دهد که مقادیر در یک جدول منحصر به فرد باشند و هیچ‌یک از آنها نمی‌توانند NULL باشند. **این محدودیت به شناسایی یکتا هر رکورد در جدول کمک می‌کند**. برای مثال، اگر در یک جدول از کارمندان، ستون EmployeeID به عنوان PRIMARY KEY تعریف شود، هیچ دو کارمند نمی‌توانند EmployeeID یکسانی داشته باشند و این ستون نمی‌تواند خالی باشد.
- **کلید خارجی (FOREIGN KEY)** به ایجاد رابطه بین دو جدول کمک می‌کند. این کلید در یک جدول به کلید اصلی جدول دیگر اشاره دارد. کلید خارجی از ورود داده‌های نامعتبر به ستون کلید خارجی جلوگیری می‌کند، زیرا داده‌ها باید یکی از مقادیر موجود در جدول والد باشند. برای مثال، اگر در یک جدول سفارشات، ستون CustomerID به عنوان کلیدخارجی به ستون CustomerID در جدول مشتریان اشاره کند، هر مقدار در ستون CustomerID در جدول سفارشات باید در جدول مشتریان وجود داشته باشد.
- **محدودیت NOT NULL** اطمینان می‌دهد که یک ستون خاص در جدول نمی‌تواند مقادیر NULL (تهی) داشته باشد. این به معنای آن است که هر رکورد در آن ستون باید یک مقدار معتبر داشته باشد. برای مثال، اگر ستون Email در یک جدول با محدودیت NOT NULL تعریف شود، هر کارمند باید یک آدرس ایمیل داشته باشد و این ستون نمی‌تواند خالی باشد.

چرا SQL را یاد بگیریم؟

در دنیای امروز که بر اساس داده‌ها پیش می‌رود، یکی از مهارت‌های اصلی مورد نیاز SQL است. در ادامه، دلایل مهم یادگیری SQL را بررسی می‌کنیم:

- **پایه علم داده و تحلیل‌ها:** SQL دروازه‌ای به سمت تحلیل‌هایِ پیشرفته است. بنابراین، برای کسی که به علم داده یا یادگیری ماشین علاقه دارد، SQL یک مهارت ضروری است.
- **زبان استاندارد برای پایگاه داده‌هایِ رابطه‌ای:** SQL زبان استاندارد برای مدیریت و کوئری پایگاه داده‌هایِ رابطه‌ای مانند PostgreSQL، MySQL و SQL Server است. این زبان به‌طور گسترده در صنایع مختلف استفاده می‌شود.
- **مدیریت موثر داده‌ها:** SQL به شما امکان می‌دهد تا به‌راحتی با مجموعه‌های داده بزرگ کار کنید و به‌طور دقیق کوئری‌هایی را برای بازیابی یا دستکاری داده‌ها اجرا کنید. این ویژگی SQL را برای گزارش‌گیری، شناسایی روندها و تصمیم‌گیری‌های مبتنی بر داده ضروری می‌کند.
- **فرصت‌های شغلی:** تسلط بر SQL به شما کمک می‌کند تا در نقش‌های شغلی مختلف از یک تحلیل‌گر داده تا یک مدیر پایگاه داده و توسعه‌دهنده هوش تجاری کار کنید. این مهارت یکی از پرطرفدارترین مهارت‌ها در صنایع متنوع از فناوری تا مالی و بهداشت است.

SQL

SQL یک **زبان اعلامی**[1] است. به این معنی که هنگام کدنویسی علاقه‌ای به نحوهِ انجام کار ندارید و تمرکز روی نتیجه‌ای است که می‌خواهید بدست آورید. این ویژگی یادگیری SQL را در مقایسه با سایر زبان‌های برنامه‌نویسی بسیار آسان می‌کند.

زبان‌های رویه‌ای[2]

در زبان‌های رویه‌ای، برنامه‌نویس باید مراحل دقیق انجام یک وظیفه را مشخص کند. برای مثال، اگر بخواهید یک لیست از مشتریان را از یک فایل بخوانید، باید مراحل زیر را کدنویسی کنید:

- فایل را باز کن.
- داده‌ها را بخوان.
- داده‌ها را پردازش کن.
- نتایج را نمایش بده.

زبان‌های اعلامی

در زبان اعلامی، تنها کافی است هدف خود را بیان کنید. مثلا در SQL کافی است بگویید: "**لیست مشتریان را نمایش بده**". این کار را می‌توانید با دستور زیر انجام دهید:

```
SELECT*FROMcustomers;
```

در این دستور، SQL وظیفه دارد که داده‌های مورد نظر را از جدول "customers" بازیابی کند و نمایش دهد، بدون اینکه نیاز باشد مراحل پردازش را به صورت صریح توضیح دهید.

شروع کار با PostgreSQL

در این بخش، با یادگیری نحوهِ نوشتن دستورات SQL برای تعامل با پایگاه داده، کار با PostgreSQL را آغاز خواهیم کرد.

PostgreSQL چیست؟

PostgreSQL که اغلب Postgres نامیده می‌شود، یک سیستم مدیریت پایگاه داده رابطه‌ای[3] منبع‌باز است. postgreSQL کوئری استاندارد SQL را پشتیبانی می‌کند و **نحو**[4] آن مشابه است.

---

[1] declarative language
[2] Procedural Language
[3] RDBMS
[4] syntax

انواع داده‌ها

در SQL، داده‌ها انواع مختلفی دارند و هرکدام برای ذخیره‌سازی نوع خاصی استفاده می‌شوند. به عبارت ساده‌تر، داده‌ها همان چیزی هستند که در پایگاه داده ذخیره می‌کنیم (مثلا نام، تاریخ تولد، شماره تلفن، و غیره). برای اینکه SQL بتواند این داده‌ها رو به درستی ذخیره و مدیریت کند، باید نوع داده‌ای که قرار هست ذخیره شود رو مشخص کنیم. به این ترتیب، SQL می‌داند که چطور با آن داده‌ها کار کند.

دسته‌بندی انواع داده‌ها در PostgreSQL:

- **عددی** ‹Numeric›

PostgreSQ دو نوع مجزا از اعداد را ارائه می‌دهد:

▪ اعداد صحیح:

نام	اندازه ذخیره‌سازی	شرح	محدوده
SMALLINT	۲ بایت	عدد صحیح بازه کوچک	-۳۲۷۶۸ تا +۳۲۷۶۷
INTEGER	۴ بایت	انتخاب معمول برای عدد صحیح	-۲۱۴۷۴۸۳۶۴۸ تا +۲۱۴۷۴۸۳۶۴۷
BIGINT	۸ بایت	عدد صحیح بازه بزرگ	-۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۸ تا +۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷
SMALLSERIAL	۲ بایت	عدد صحیح خودافزاینده کوچک	۱ تا ۳۲۷۶۷
SERIAL	۴ بایت	عدد صحیح خودافزاینده	۱ تا ۲۱۴۷۴۸۳۶۴۷
BIGSERIAL	۸ بایت	عدد صحیح خودافزاینده بزرگ	۱ تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷

- **اگر بخواهید مقداری را فراتر از محدوده‌های مجاز، وارد یا به‌روز کنید، PostgreSQL خطایی ایجاد می‌کند.**
- **INT مترادف INTEGER است و می‌توانید آن‌ها را به جای یکدیگر استفاده کنید.**

▪ اعداد ممیز شناور:

نام	اندازه ذخیره‌سازی	شرح
NUMERIC(p, s) DECIMAL(p, s) **DEC(p,s)**	متغیر	p تعداد کل ارقام را نشان می‌دهد. s تعداد ارقام بعد از نقطه اعشار را نشان می‌دهد. نوع NUMERIC می‌تواند مقداری تا **۱۳۱۰۷۲** رقم قبل از نقطه اعشار، **۱۶۳۸۳** رقم بعد از نقطه اعشار را در خود جای دهد.
REAL	۴ بایت	دقت ۶ رقم اعشاری
DOUBLE PRECISION	۸ بایت	دقت ۱۵ رقم اعشاری

- **متنی** ‹String›

نام	شرح
CHARACTER VARYING(N) **VARCHAR(N)**	رشته کاراکتری با طول متغیر، حداکثر طول ثابت
CHARACTER(N) **CHAR(N)**	رشته کاراکتر با طول ثابت
TEXT VARCHAR	طول نامحدود

- **هر دو نوع CHAR(n) و VARCHAR(n) می‌توانند تا n کاراکتر را ذخیره کنند. اگر بخواهید رشته‌ای را ذخیره کنید که بیش از n کاراکتر دارد، PostgreSQL یک خطا صادر می‌کند.**
- **اگر n عدد صحیح را برای نوع داده VARCHAR مشخص نکنید، مانند نوع داده TEXT عمل می‌کند. عملکرد VARCHAR (بدون n) و TEXT یکسان است.**
- **در بیشتر موارد، بهتر است از TEXT یا VARCHAR استفاده کنید و از VARCHAR(n) فقط زمانی استفاده کنید که بخواهید PostgreSQL طول را بررسی کند.**

رشته کاراکتر با طول ثابت:

- **SBU**
- **dbsbu.github.io**
- **PostgreSQL**

S	B	U	blank	blank	blank	blank	blank	blank	blank	blank	blank	blank	blank	blank
d	b	s	b	u	.	g	i	t	h	u	b	.	i	o
P	o	s	t	g	r	e	S	Q	L	blank	blank	blank	blank	blank

رشته کاراکتر با طول متغیر:

- **SBU**
- **dbsbu.github.io**
- **PostgreSQL**

S	B	U												
d	b	s	b	u	.	g	i	t	h	u	b	.	i	o
P	o	s	t	g	r	e	S	Q	L					

- **منطقی** ‹Boolean›

نام	اندازه ذخیره‌سازی	شرح
BOOLEAN	۱ بایت	نوع بولی سه مقدار دارد:
BOOL		true
		false
		NULL

- علاوه بر True و False، PostgreSQL از نمایش‌های مختلفی برای true و false در کوئری‌های SQL استفاده می‌کند:

true	false
't'	'f'
'true'	'false'
'y'	'n'
'yes'	'no'
'1'	'0'

- **تاریخی و زمانی** ‹Date & Time›

نوع داده	شرح	قالب و مثال
DATE	فقط تاریخ	'2025-03-30' : YYYY-MM-DD
TIME	فقط زمان	'14:30:00' : HH:MI:SS
TIMESTAMP	تاریخ و ساعت با دقت ثانیه	'2025-03-30 14:30:00' : YYYY-MM-DD HH:MI:SS

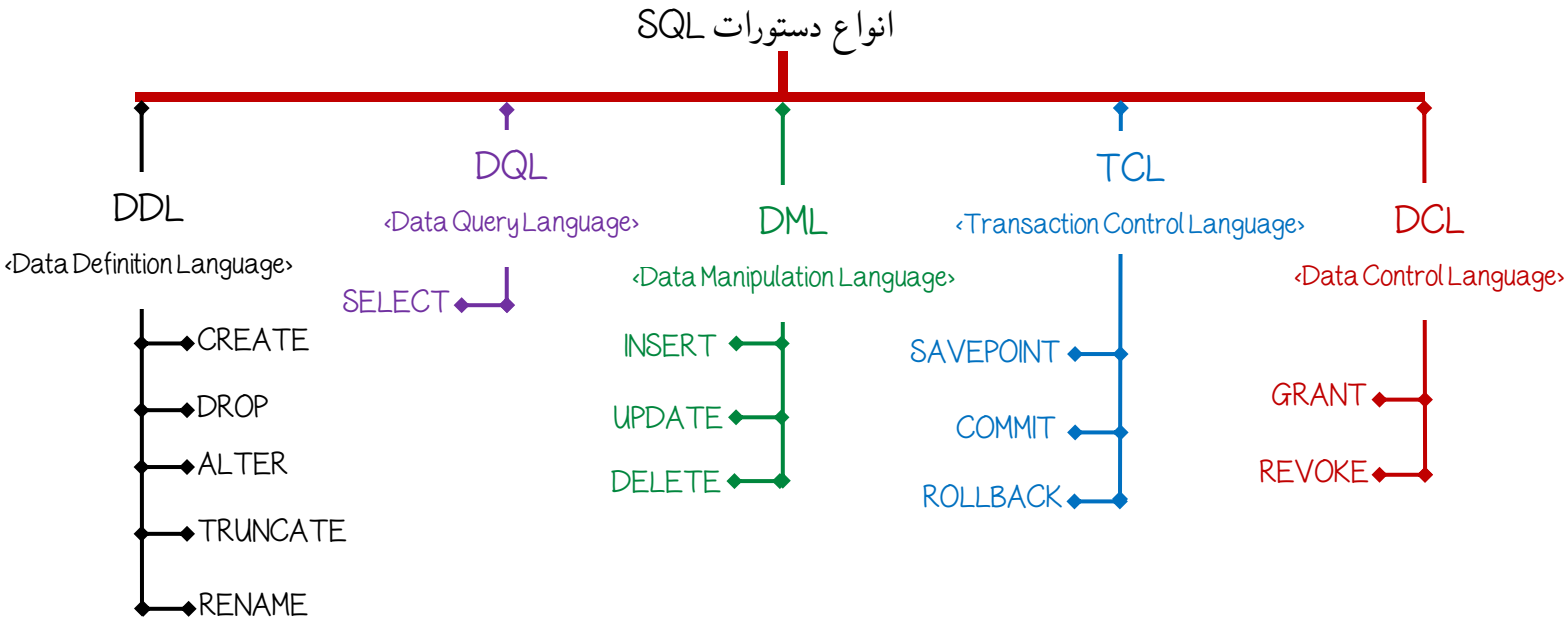
برای مقداردهی خودکار از توابع زیر استفاده می‌کنیم:

- CURRENT_DATE → تاریخ فعلی
- CURRENT_TIME → ساعت فعلی
- NOW() → تاریخ و ساعت فعلی
- CURRENT_TIMESTAMP → تاریخ و ساعت فعلی با منطقه زمانی

دسته‌بندی دستورات SQL

دستورات SQL مجموعه‌ای از فرامین هستند که برای تعامل با پایگاه‌های داده به کار می‌روند. این دستورات به چند دسته اصلی تقسیم می‌شوند:

- دستورات **DDL (Data Definition Language)** که برای **تعریف و مدیریت ساختار** پایگاه داده مانند CREATE، ALTER، TRUNCATE، RENAME و DROP استفاده می‌شوند.
- دستورات **DML (Data Manipulation Language)** که برای **مدیریت و دستکاری داده‌های ذخیره‌شده** در جداول مانند **INSERT**، **UPDATE** و **DELETE** به کار می‌روند.
- دستورات **DQL (Data Query Language)** که شامل فرمان **SELECT** برای **بازیابی اطلاعات** از پایگاه داده است.
- دستورات **DCL (Data Control Language)** که شامل **GRANT** و **REVOKE** برای **مدیریت سطح دسترسی کاربران** به داده‌ها هستند.
- و در نهایت دستورات **TCL (Transaction Control Language)** مانند **COMMIT**، **ROLLBACK** و **SAVEPOINT** که برای **کنترل تراکنش‌های** پایگاه داده استفاده می‌شوند.



دستورِ ایجاد پایگاه داده

برای ایجاد یک پایگاه داده، از دستور CREATE DATABASE استفاده می‌شود:

CREATE DATABASE نام پایگاه داده;

مثال: یک پایگاه داده برای دانشگاه و یک پایگاه داده برای داروخانه ایجاد کنید.

```
ایجاد پایگاه داده برای دانشگاه --

CREATE DATABASE universitydb;

ایجاد پایگاه داده برای داروخانه --

CREATE DATABASE pharmacydb;
```

نکته: در psql، نام پایگاه داده‌ها **به‌طور پیش‌فرض به حروف کوچک تبدیل می‌شوند**، مگر اینکه از علامت نقل قول دوگانه "" استفاده شود. این بدان معنی است که اگر نام پایگاه داده را با حروف بزرگ بنویسید، به صورت خودکار به حروف کوچک تبدیل می‌شود. برای مثال:

```
CREATE DATABASE UniversityDB;
```

این دستور یک پایگاه داده به نام universitydb ایجاد می‌کند. اگر می‌خواهید نام پایگاه داده با حروف بزرگ ذخیره شود، باید از علامت نقل قول دوگانه استفاده کنید:

```
CREATE DATABASE "UniversityDB";
```

نکته. استفاده از حروف کوچک برای نام پایگاه داده‌ها، جدول‌ها و ستون‌ها توصیه می‌شود.

دستورِ حذفِ پایگاه داده

برای حذف یک پایگاه داده، از دستور **DROP DATABASE** استفاده می‌شود:

```
DROP DATABASE [IF EXISTS] database_name;
```

- **IF EXISTS**: این گزینه اختیاری است و اگر پایگاه داده وجود نداشته باشد، به جای خطا، یک هشدار نمایش می‌دهد.
- **database_name**: نام پایگاه داده‌ای که می‌خواهید حذف کنید.

مثال:
فرض کنید پایگاه داده‌ای به نام my_database دارید که می‌خواهید آن را حذف کنید:

```
DROP DATABASE my_database;
```

اگر پایگاه داده وجود نداشته باشد و نمی‌خواهید خطا دریافت کنید، از گزینه IF EXISTS استفاده کنید:

```
DROP DATABASE IF EXISTS my_database;
```

- **قبل از حذف پایگاه داده، اطمینان حاصل کنید که هیچ اتصال فعالی به آن وجود ندارد. اگر اتصال فعال باشد، دستور اجرا نمی‌شود.**
- **از نسخه PostgreSQL 13 به بعد، می‌توانید از گزینه WITH (FORCE) برای بستن اتصالات فعال و حذف پایگاه داده استفاده کنید:**

```
DROP DATABASE my_database WITH (FORCE);
```

دستورِ ساختِ جدول

برای ساخت جدول در SQL از دستور CREATE TABLE استفاده می‌شود. این دستور به شما امکان می‌دهد جدولی با ستون‌ها و انواع داده‌های مشخص ایجاد کنید.

```
CREATE TABLE table_name (
    column_name data_type [constraints],
    column_name data_type [constraints],
    ...
);
```

- **table_name:** نام جدول.
- **column_name:** نام ستون.
- **data_type:** نوع داده‌ای که در ستون ذخیره می‌شود، مانند INTEGER, VARCHAR, TEXT, DATE و غیره.
- **constraints:** محدودیت‌هایی مانند NOT NULL, PRIMARY KEY, UNIQUE و غیره.

مثال ۱: ساخت جدول برای ذخیره داده‌های کتاب‌ها

```
CREATE TABLE books (
    id INT,
    title VARCHAR(200),
    author VARCHAR(100),
    publisher VARCHAR(100),
    publication_date DATE,
    price DECIMAL(10, 2)
);
```

مثال ۲: ساخت جدول برای ذخیره داده‌های کارکنان

```
CREATE TABLE employees (
    id INT,
    name VARCHAR(100),
    surname VARCHAR(100),
    age INT,
    department VARCHAR(50),
    hire_date DATE
);
```

مثال ۳: ساخت جدول برای ذخیره داده‌های مشتریان

```
CREATE TABLE customers (
    id INT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_number VARCHAR(20),
    address VARCHAR(200)
);
```

اگر می‌خواهید از خطا در صورت وجود داشتن جدول جلوگیری کنید، از گزینه IF NOT EXISTS استفاده کنید:

```
CREATE TABLE IF NOT EXISTS employees (
    id INT,
    name VARCHAR(100),
    surname VARCHAR(100),
    age INT,
    department VARCHAR(50),
    hire_date DATE
);
```

دستور حذف جدول

اگر نیاز به حذف جدول داشتید، از دستور زیر استفاده کنید:

DROP TABLE table_name;

table_name: نام جدولی که می‌خواهید حذف کنید.

✓ در SQL می‌توانید از گزینه **IF EXISTS** برای حذف جدول استفاده کنید تا اگر جدول وجود ندارد، خطایی رخ ندهد:

DROP TABLE IF EXISTS table_name;

درج داده در جدول

پس از ایجاد جدول، می توانید یک یا چند ردیف را در آن قرار دهید. دستور **INSERT INTO** در PostgreSQL برای افزودن رکوردهای جدید به یک جدول استفاده می‌شود. این دستور به دو شکل می‌تواند نوشته شود:

- **بدون مشخص کردن ستون‌ها:** در این حالت، باید مقادیر را به ترتیب ستون‌های جدول وارد کنید:

INSERT INTO table_name
VALUES (value1, value2, value3, ...);

- **با مشخص کردن ستون‌ها:** در این حالت، می‌توانید نام ستون‌ها را مشخص کنید و مقادیر را به ترتیب آن ستون‌ها وارد کنید:

INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

در این نحو:

- INSERT INTO کلمات کلیدی هستند که به PostgreSQL دستور می‌دهند تا یک ردیف جدید را در جدول وارد کند.
- table_name نام جدولی است که می‌خواهید داده اضافه کنید.

- پس از نام جدول، فهرستی از ستون‌های جدا شده با کاما را در داخل پرانتز () مشخص می‌کنید.
- VALUES یک کلمه کلیدی است که به PostgreSQL در مورد داده‌هایی که می‌خواهید به جدول اضافه کنید می‌گوید.
- بعد از کلمه کلیدی VALUES، فهرستی از مقادیر جدا شده با کاما را که مربوط به ستون‌های داخل مجموعه‌ای از پرانتز است، مشخص کنید.
- نقطه ویرگول (;) در انتهای عبارت اختیاری است. وقتی دو عبارت دارید و می‌خواهید آن‌ها را از هم جدا کنید می‌تواند مفید باشد.

مثال: افزودن داده به جدول کارکنان

```
INSERT INTO employees (id, name, surname, age, department, hire_date)
VALUES (1, 'Saman', 'Bahrami', 30, 'Sales', '2020-01-01');

INSERT INTO employees (id, name, surname, age, department, hire_date)
VALUES (2, 'Mehrshad', 'Karimi', 28, 'Marketing', '2021-06-01');
```

درج چند ردیف در جدول

PostgreSQL به شما امکان می‌دهد چندین ردیف را با استفاده از دستور زیر به‌طور هم‌زمان در یک جدول قرار دهید:

```
INSERT INTO table_name (column1, column2, ...)
VALUES
    (value11, value12, ...),
    (value21, value22, ...),
    (value31, value32, ...),
    ...;
```

در این نحو، به جای تعیین یک ردیف داده در عبارت VALUES، چندین ردیف داده را قرار می‌دهید.

مثال: افزودن داده به جدول کارکنان

```
INSERT INTO employees (id, name, surname, age, department, hire_date)
VALUES (1, 'Saman', 'Bahrami', 30, 'Sales', '2020-01-01'),
       (2, 'Mehrshad', 'Karimi', 28, 'Marketing', '2021-06-01');
```

عبارت (گزاره یا قید) RETURNING

در PostgreSQL دستور INSERT دارای یک عبارت RETURNING اختیاری است که اطلاعات سطر درج شده را برمی‌گرداند. اگر می‌خواهید کل ردیف درج شده را برگردانید، از علامت ستاره (*) بعد از کلمه کلیدی RETURNING استفاده کنید:

```
INSERT INTO table1(column1, column2, …)
VALUES (value1, value2, …)
RETURNING *;
```

اگر می خواهید برخی اطلاعات را در مورد ردیف درج شده برگردانید، می‌توانید یک یا چند ستون را بعد از عبارت RETURNING مشخص کنید. برای مثال، عبارت زیر شناسه سطر درج شده را برمی‌گرداند:

```
INSERT INTO employees (id, name, surname, age, department, hire_date)
VALUES (1, 'Saman', 'Bahrami', 30, 'Sales', '2020-01-01'),
       (2, 'Mehrshad', 'Karimi', 28, 'Marketing', '2021-06-01') RETURNING id;
```

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های آموزشی آمار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

شنبه، ۱۴۰۴/۰۱/۲۳

Saturday - 2025 12 April

In [ ]:
```
!pip install jupysql psycopg2-binary
```

In [1]:
```
%load_ext sql
```

Connecting to 'default'

In [2]:
```
%config SqlMagic.displaylimit = None
```

displaylimit: Value None will be treated as 0 (no limit)

اتصال به PostgreSQL با یکی از روش‌های زیر

```
%sql postgresql://your_username:your_password@localhost:5432/your_database
%sqlcmd connect
```

شایان ذکر است با اجرای دستور `%load_ext sql` به پایگاه داده پیش‌فرض متصل می‌شود و نیازی به اجرای دستورات بالا را ندارد.

کار با جدول و معرفی محدودیت‌ها (Constraints) در PostgreSQL

در PostgreSQL، محدودیت‌ها (Constraints) قوانینی هستند که برای کنترل داده‌های موجود در جداول اعمال می‌شوند. این محدودیت‌ها اطمینان حاصل می‌کنند که داده‌های وارد شده در یک ستون یا جدول، شرایط خاصی را برآورده کنند.

کلید اصلی

کلید اصلی یک ستون یا گروهی از ستون‌ها است که برای شناسایی منحصر به فرد یک ردیف در جدول استفاده می‌شود. یک جدول می‌تواند صفر یا یک کلید اصلی داشته باشد و نمی‌تواند بیش از یک کلید اصلی داشته باشد.

محدودیت کلید اصلی شامل محدودیت `NOT NULL` و محدودیت `UNIQUE` است.

نحو آن به صورت زیر است:

```
CREATE TABLE table_name (
  column_1 data_type PRIMARY KEY,
  column_2 data_type,
  …
);
```

در نحو بالا، شما **کلید اصلی** را به عنوان محدودیت ستون تعریف می‌کنید.

اگر کلید اصلی از بیش از یک ستون تشکیل شده باشد، باید آن را با استفاده از محدودیت جدول تعریف کنید:

```
CREATE TABLE table_name (
  column_1 data_type,
  column_2 data_type,
  column_3 data_type,
  …
  PRIMARY KEY(column_1, column2, ...)
);
```

مثال: ایجاد جدول با کلید اصلی که از یک ستون تشکیل شده است

عبارت زیر یک جدول با یک کلید اصلی ایجاد می‌کند که از یک ستون تشکیل شده است:

In [14]:
```
%%sql
DROP TABLE IF EXISTS orders;
```

```
CREATE TABLE
  orders (
    order_id SERIAL PRIMARY KEY,
    customer_id VARCHAR(255) NOT NULL,
    order_date DATE NOT NULL
  );
```

Running query in 'default'

Out[14]:

برای نمایش جدول‌های ساخته شده در پایگاه داده در محیط `JupySQL` از دستور زیر استفاده می‌شود:

In [15]:
```
%sqlcmd tables
```

Out[15]:

Name
students
users
customers
product_tags
warehouses
production_orders
brands
product_loans
profiles
profiles_test
student_nulls
items
staff
student
student_english
inventories
products
orders
orders_test
orders_test_2
customers_test

برای توضیح در مورد ساختار یک جدول ساخته شده در پایگاه داده در محیط `JupySQL` از دستور زیر استفاده می‌شود:

In [16]:
```
%sqlcmd columns -t orders
```

Out[16]:

name	type	nullable	default	autoincrement	comment
order_id	INTEGER	False	nextval('orders_order_id_seq'::regclass)	True	None
customer_id	VARCHAR(255)	False	None	False	None
order_date	DATE	False	None	False	None

ما ستون order_id را با نوع SERIAL تعریف می‌کنیم تا PostgreSQL یک عدد صحیح منحصر به فرد (1، 2، 3 و غیره) وقتی یک ردیف جدید را در جدول وارد می‌کنید، ایجاد کند، بدون اینکه مقدار ستون order_id را ارائه کنید. این تضمین می‌کند که مقدار order_id برای هر ردیف در جدول منحصر به فرد است.

In [17]:
```
%%sql
INSERT INTO
  orders (customer_id, order_date)
VALUES
  ('C001', '2022-01-01'),
  ('C002', '2022-01-05'),
  ('C003', '2022-01-10'),
  ('C001', '2022-01-15'),
  ('C004', '2022-01-20'),
  ('C005', '2022-01-25'),
```

```
  ('C002', '2022-02-01'),
  ('C006', '2022-02-05'),
  ('C007', '2022-02-10'),
  ('C003', '2022-02-15') RETURNING *;
```

Running query in 'default'

10 rows affected.

Out[17]:

order_id	customer_id	order_date
1	C001	2022-01-01
2	C002	2022-01-05
3	C003	2022-01-10
4	C001	2022-01-15
5	C004	2022-01-20
6	C005	2022-01-25
7	C002	2022-02-01
8	C006	2022-02-05
9	C007	2022-02-10
10	C003	2022-02-15

روش دیگر و مرسوم در `Postgres` برای افزاینده خودکار کلید اصلی

برای تعریف یک ستون خودافزاینده در PostgreSQL، از ویژگی GENERATED ALWAYS AS IDENTITY به صورت زیر استفاده می‌شود:

```
id INT GENERATED ALWAYS AS IDENTITY
```

برای تعریف یک ستون خودافزاینده به عنوان یک ستون کلید اصلی، محدودیت کلید اصلی اضافه می‌شود:

```
id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY
```

In [18]:
```
%%sql
DROP TABLE IF EXISTS brands;
CREATE TABLE
  brands (
    brand_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR(50)
  );
```

Running query in 'default'

Out[18]:

توجه داشته باشید که استفاده از SERIAL به دلیل برخی مشکلات کم‌تر توصیه می‌شود.

In [19]:
```
%%sql
INSERT INTO
  brands (name)
VALUES
  ('Apple') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[19]:

brand_id	name
1	Apple

In [20]:
```
%%sql
INSERT INTO
  brands (name)
VALUES
  ('Samsung') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[20]:

brand_id	name
2	Samsung

کلید اصلی ترکیبی

هنگامی که یک کلید اصلی از دو یا چند ستون تشکیل شده باشد، به آن کلید اصلی ترکیبی می‌گویند.

برای مثال، عبارت CREATE TABLE زیر یک جدول جدید product_tags ایجاد می‌کند که کلید اصلی آن شامل ستون‌های product_id و tag_id است:

In [21]:
```
%%sql
DROP TABLE IF EXISTS product_tags;
CREATE TABLE
  product_tags (
    product_id INT,
    tag_id INT,
    PRIMARY KEY (product_id, tag_id)
  );
```

Running query in 'default'

Out[21]:

افزودن کلید اصلی به جدول موجود

برای افزودن کلید اصلی به جدول موجود از نحو زیر استفاده می‌کنیم:

```
ALTER TABLE table_name
ADD PRIMARY KEY (column_1, column_2, ...);
```

اگر به صراحت نامی را برای محدودیت کلید اصلی مشخص نکنید، PostgreSQL یک نام پیش فرض را به محدودیت کلید اصلی اختصاص می‌دهد. به طور پیش فرض، PostgreSQL از فرمت table-name_pkey به عنوان نام پیش فرض برای محدودیت کلید اصلی استفاده می‌کند.

مثال: ابتدا جدولی به نام `products` بدون تعریف کلید اولیه ایجاد می‌کنیم.

In [22]:
```
%%sql
DROP TABLE products;
CREATE TABLE
  products (
    product_id INT,
    name VARCHAR,
    description TEXT,
    price DEC(10, 2)
  );
```

Running query in 'default'

Out[22]:

In [23]:
```
%%sql
ALTER TABLE products ADD PRIMARY KEY (product_id);
```

Running query in 'default'

Out[23]:

حذف کلید اصلی از جدول موجود

برای حذف یک کلید اصلی از جدول، از عبارت ALTER TABLE زیر استفاده می‌کنیم :

```
ALTER TABLE table_name
DROP CONSTRAINT primary_key_constraint;
```

مثال: کلید اصلی را از جدول `products` حذف کنید.

In [24]:
```
%%sql
ALTER TABLE products
DROP CONSTRAINT products_pkey;
```

Running query in 'default'

Out[24]:

محدودیت `NOT NULL`

NULL

NULL عدم وجود یک مقدار را نشان می‌دهد که نشان می دهد داده ناشناخته یا گم شده است. به عنوان مثال، اگر می‌خواهید انبار را ضبط کنید اما آدرس آن را در زمان ثبت نمی‌دانید، می توانید NULL را در ستون آدرس قرار دهید. بعداً می توانید NULL را در ستون آدرس با آدرس صحیح به‌روز کنید. توجه داشته باشید که NULL به معنای یک مقدار ناشناخته است. با رشته خالی ('') یا صفر (0) که مقادیر تعریف شده هستند متفاوت است.

وقتی ستونی را برای جدول تعریف می‌کنید، آن ستون NULL تعریف می‌شود. به این معنی است که می‌توانید NULL را در ستون وارد کنید.

In [26]:
```
%%sql
DROP TABLE IF EXISTS warehouses;
CREATE TABLE
  warehouses (
    warehouse_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR,
    address VARCHAR
  );
```

Running query in 'default'

Out[26]:

In [27]:
```
%sqlcmd columns -t warehouses
```

Out[27]:

name	type	nullable	default	autoincrement	comment	identity
warehouse_id	INTEGER	False	None	True	None	{'always': True, 'start': 1, 'increment': 1, 'minvalue': 1, 'maxvalue': 2147483647, 'cache': 1, 'cycle': False}
name	VARCHAR	True	None	False	None	
address	VARCHAR	True	None	False	None	

عبارت `INSERT` زیر یک ردیف جدید را در جدول `warehouses` با `NULL` در ستون `address` وارد می‌کند:

In [28]:
```
%%sql
INSERT INTO
  warehouses (name, address)
VALUES
  ('IranMall', NULL) RETURNING name,
  address;
```

Running query in 'default'

1 rows affected.

Out[28]:

name	address
IranMall	None

اگر ستونی را در عبارت `INSERT` حذف کنید، `PostgreSQL` از `NULL` به عنوان مقدار پیش فرض استفاده می‌کند. به عنوان مثال:

In [29]:
```
%%sql
INSERT INTO
  warehouses (name)
VALUES
  ('IranMall') RETURNING name,
  address;
```

Running query in 'default'

1 rows affected.

Out[29]:

name	address
IranMall	None

استفاده از محدودیت NOT NULL

برای اطمینان از اینکه یک ستون نمی‌تواند `NULL` داشته باشد، از محدودیت `NOT NULL` استفاده می‌شود. در زیر نحو برای تعریف یک محدودیت `NOT NULL` برای یک ستون آمده است:

```
column_name data_type NOT NULL
```

پس از تعریف یک محدودیت `NOT NULL`، ستون، `NULL` را نمی‌پذیرد. اگر سعی کنید این کار را انجام دهید، `PostgreSQL` یک خطا صادر می‌کند.

In [30]:
```
%%sql
DROP TABLE IF EXISTS warehouses;
```

Running query in 'default'

Out[30]:

In [31]:
```
%%sql
CREATE TABLE
  warehouses (
    warehouse_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR NOT NULL,
    address VARCHAR NOT NULL
  );
```

Running query in 'default'

Out[31]:

In [ ]:
```
%sqlcmd columns -t warehouses
```

In [32]:
```
%%sql
INSERT INTO
  warehouses (name, address)
VALUES
  ('DigiKala', NULL) RETURNING name,
  address;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.NotNullViolation) null value in column "address" of relation "warehouses" violates not-null cons
traint
DETAIL:  Failing row contains (1, DigiKala, null).

[SQL: INSERT INTO
  warehouses (name, address)
VALUES
  ('DigiKala', NULL) RETURNING name,
  address;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [33]:
```
%%sql
INSERT INTO
  warehouses (name, address)
VALUES
  ('DigiKala', 'Tehran') RETURNING name,
  address;
```

Running query in 'default'

1 rows affected.

Out[33]:

name	address
DigiKala	Tehran

افزودن محدودیت‌های NOT NULL به ستون‌های موجود

برای اضافه کردن محدودیت `NOT NULL` به ستونی از جدول موجود، از عبارت `ALTER TABLE` به صورت زیر استفاده می‌کنیم:

```
ALTER TABLE table_name
ALTER COLUMN column_name SET NOT NULL;
```

برای اضافه کردن چندین محدودیت NOT NULL به چندین ستون، از دستور زیر استفاده می‌کنیم:

```
ALTER TABLE table_name
ALTER COLUMN column_name_1 SET NOT NULL,
ALTER COLUMN column_name_2 SET NOT NULL,
...;
```

برای حذف کردن محدودیت `NOT NULL` از ستونی از جدول موجود، از عبارت `ALTER TABLE` به صورت زیر استفاده:

```
ALTER TABLE table_name
ALTER COLUMN column_name DROP NOT NULL;
```

مثال

In [35]:
```
%%sql
DROP TABLE IF EXISTS production_orders;
CREATE TABLE
  production_orders (
    id SERIAL PRIMARY KEY,
    description VARCHAR NOT NULL,
    material_id VARCHAR,
    qty NUMERIC,
    start_date DATE,
    finish_date DATE
  );
```

Running query in 'default'

Out[35]:

In [36]:
```
%sqlcmd columns -t production_orders
```

Out[36]:

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('production_orders_id_seq'::regclass)	True	None
description	VARCHAR	False	None	False	None
material_id	VARCHAR	True	None	False	None
qty	NUMERIC	True	None	False	None
start_date	DATE	True	None	False	None
finish_date	DATE	True	None	False	None

In [37]:
```
%%sql
ALTER TABLE production_orders
ALTER COLUMN qty
SET
  NOT NULL;
```

Running query in 'default'

Out[37]:

In [38]:
```
%sqlcmd columns -t production_orders
```

Out[38]:

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('production_orders_id_seq'::regclass)	True	None
description	VARCHAR	False	None	False	None
material_id	VARCHAR	True	None	False	None
qty	NUMERIC	False	None	False	None
start_date	DATE	True	None	False	None
finish_date	DATE	True	None	False	None

In [39]:
```
%%sql
ALTER TABLE production_orders
ALTER COLUMN qty
DROP NOT NULL;
```

Running query in 'default'

Out[39]:

In [40]:
```
%sqlcmd columns -t production_orders
```

Out[40]:

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('production_orders_id_seq'::regclass)	True	None
description	VARCHAR	False	None	False	None
material_id	VARCHAR	True	None	False	None
qty	NUMERIC	True	None	False	None
start_date	DATE	True	None	False	None
finish_date	DATE	True	None	False	None

محدودیت `UNIQUE`

گاهی اوقات، شما می‌خواهید اطمینان حاصل کنید که مقادیر ذخیره شده در یک ستون یا گروهی از ستون‌ها در کل جدول منحصر به فرد هستند، مانند آدرس‌های ایمیل یا نام‌های کاربری. PostgreSQL با محدودیت `UNIQUE` به شما کمک می‌کند تا یکتایی داده‌ها را به درستی حفظ کنید. هنگامی که یک محدودیت `UNIQUE` وجود دارد، هر بار که یک ردیف جدید وارد می‌شود، بررسی می‌کند که آیا مقدار قبلاً در جدول وجود دارد یا خیر. اگر مقدار قبلاً وجود داشته باشد، یک خطا صادر می‌کند.

در زیر نحو برای تعریف یک محدودیت UNIQUE آمده است:

```
CREATE TABLE table_name (
  column1 data_type CONSTRAINT constraint_name UNIQUE,
  ...
);
```

توجه داشته باشید که CONSTRAINT constraint_name اختیاری است. اگر آن را حذف کنید، PostgreSQL به طور خودکار یک نام محدودیت در قالب زیر ایجاد می‌کند:

```
{table_name}_{column_name}_key
```

این نحو یک محدودیت ستونی ایجاد می‌کند زیرا ما آن را به عنوان بخشی از تعریف ستون تعریف می‌کنیم.

شما می‌توانید یک محدودیت `UNIQUE` را به عنوان یک محدودیت جدول تعریف کنید:

```
CREATE TABLE table_name (
  column1 data_type,
  ...,
  CONSTRAINT constraint_name UNIQUE
);
```

در عمل، شما اغلب یک محدودیت جدول UNIQUE را زمانی تعریف می‌کنید که شامل دو یا چند ستون باشد:

```
CREATE TABLE table_name (
  column1 data_type,
  column2 data_type,
  CONSTRAINT constraint_name UNIQUE (column1, column2)
);
```

مثال

In [41]:
```
%%sql
DROP TABLE brands;
CREATE TABLE
  brands (
    brand_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR(255) NOT NULL UNIQUE
  );
```

Running query in 'default'

Out[41]:

In [42]:
```
%%sql
INSERT INTO
  brands (name)
VALUES
  ('Apple') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[42]:

brand_id	name
1	Apple

In [43]:
```
%%sql
INSERT INTO
  brands (name)
VALUES
  ('Apple') RETURNING *;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.UniqueViolation) duplicate key value violates unique constraint "brands_name_key"
DETAIL:  Key (name)=(Apple) already exists.

[SQL: INSERT INTO
  brands (name)
VALUES
  ('Apple') RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

مثال

In [44]:
```
%%sql
DROP TABLE IF EXISTS product_loans;
CREATE TABLE
  product_loans (
    loan_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    product_id INT NOT NULL,
    user_id INT NOT NULL,
    loan_date DATE NOT NULL,
    return_date DATE,
    UNIQUE (product_id, user_id)
  );
```

Running query in 'default'

Out[44]:

In [45]:
```
%%sql
INSERT INTO
  product_loans (product_id, user_id, loan_date)
VALUES
```

```
  (1, 1, '2024-11-23'),
  (1, 2, '2024-11-23') RETURNING *;
```

Running query in 'default'

2 rows affected.

Out[45]:

loan_id	product_id	user_id	loan_date	return_date
1	1	1	2024-11-23	None
2	1	2	2024-11-23	None

In [46]:
```
%%sql
INSERT INTO
  product_loans (product_id, user_id, loan_date)
VALUES
  (1, 1, '2024-11-24');
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.UniqueViolation) duplicate key value violates unique constraint "product_loans_product_id_user_i
d_key"
DETAIL:  Key (product_id, user_id)=(1, 1) already exists.

[SQL: INSERT INTO
  product_loans (product_id, user_id, loan_date)
VALUES
  (1, 1, '2024-11-24');]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

مدیریت NULL با محدودیت UNIQUE

در PostgreSQL یک NULL با NULL دیگر متفاوت است. بنابراین، می‌توانید چندین NULL را در یک ستون محدودیت UNIQUE وارد کنید. به عنوان مثال:

In [47]:
```
%%sql
DROP TABLE IF EXISTS profiles;
CREATE TABLE
  profiles (
    user_id INT PRIMARY KEY,
    first_name VARCHAR NOT NULL,
    last_name VARCHAR NOT NULL,
    phone VARCHAR UNIQUE
  );
```

Running query in 'default'

Out[47]:

In [48]:
```
%%sql
INSERT INTO
  profiles (user_id, first_name, last_name, phone)
VALUES
  (1, 'Milad', 'Vazan', NULL) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[48]:

user_id	first_name	last_name	phone
1	Milad	Vazan	None

In [49]:
```
%%sql
INSERT INTO
  profiles (user_id, first_name, last_name, phone)
VALUES
  (2, 'Shahrzad', 'Vazan', NULL) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[49]:

user_id	first_name	last_name	phone
2	Shahrzad	Vazan	None

از نسخه 15 به بعد، PostgreSQL گزینه‌ای به نام `NULLS NOT DISTINCT` معرفی کرده است که امکان تغییر این رفتار پیش‌فرض را فراهم می‌کند. با استفاده از این گزینه، مقادیر NULL به عنوان برابر تلقی می‌شوند و نمی‌توان بیش از یک مقدار NULL را وارد کرد:

In [50]:
```
%%sql
DROP TABLE IF EXISTS profiles_test;
CREATE TABLE
  profiles_test (
    user_id INT PRIMARY KEY,
```

```
    first_name VARCHAR NOT NULL,
    last_name VARCHAR NOT NULL,
    phone VARCHAR UNIQUE NULLS NOT DISTINCT
  );
```

Running query in 'default'

Out[50]:

In [51]:
```
%%sql
INSERT INTO
  profiles_test (user_id, first_name, last_name, phone)
VALUES
  (1, 'Milad', 'Vazan', NULL) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[51]:

user_id	first_name	last_name	phone
1	Milad	Vazan	None

In [ ]:
```
%%sql
INSERT INTO
  profiles_test (user_id, first_name, last_name, phone)
VALUES
  (2, 'Shahrzad', 'Vazan', NULL) RETURNING *;
```

افزودن محدودیت‌های UNIQUE به ستون‌های موجود

برای اضافه کردن محدودیت `UNIQUE` به ستونی از جدول موجود، از عبارت `ALTER TABLE` به صورت زیر استفاده می‌کنیم:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name UNIQUE (column_name);
```

اگر می‌خواهیم محدودیت منحصر به فرد را برای چند ستون ایجاد کنیم، می‌توانیم چند ستون را در داخل پرانتز مشخص کنیم:

```
ALTER TABLE table_name
ADD CONSTRAINT unique_name UNIQUE (column_name 1, column_name 2);
```

In [52]:
```
%%sql
DROP TABLE IF EXISTS users;
CREATE TABLE
  users (
    id SERIAL PRIMARY KEY,
    email VARCHAR,
    username VARCHAR
  );
```

Running query in 'default'

Out[52]:

In [53]:
```
%%sql
ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE (email);
```

Running query in 'default'

Out[53]:

In [54]:
```
%%sql
INSERT INTO
  users (email, username)
VALUES
  ('user1@example.com', 'user1') RETURNING *;

INSERT INTO
  users (email, username)
VALUES
  ('user2@example.com', 'user2') RETURNING *;
```

Running query in 'default'

1 rows affected.

1 rows affected.

Out[54]:

id	email	username
2	user2@example.com	user2

In [55]:
```
%%sql
INSERT INTO
  users (email, username)
VALUES
  ('user1@example.com', 'user1') RETURNING *;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.UniqueViolation) duplicate key value violates unique constraint "unique_email"
DETAIL:  Key (email)=(user1@example.com) already exists.

[SQL: INSERT INTO
  users (email, username)
VALUES
  ('user1@example.com', 'user1') RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [56]:
```
%%sql
ALTER TABLE users
DROP CONSTRAINT unique_email;
```

Running query in 'default'

Out[56]:

In [57]:
```
%%sql
INSERT INTO
  users (email, username)
VALUES
  ('user1@example.com', 'user1') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[57]:

id	email	username
4	user1@example.com	user1

In [59]:
```
%%sql
ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE (email);
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.UniqueViolation) could not create unique index "unique_email"
DETAIL:  Key (email)=(user1@example.com) is duplicated.

[SQL: ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE (email);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های آموزشی آمار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

دوشنبه، ۱۴۰۴/۰۱/۲۵

محدودیت DEFAULT

هنگام ایجاد یک جدول، می‌توانید یک مقدار پیش فرض برای یک ستون در جدول با استفاده از محدودیت DEFAULT تعریف کنید. نحو آن به صورت زیر است:

```
CREATE TABLE table_name(
    column1 type,
    column2 type DEFAULT default_value,
    column3 type,
    ...
);
```

اگر محدودیت DEFAULT را برای ستون مشخص نکنید، مقدار پیش فرض آن NULL است.

هنگام درج یک ردیف جدید در جدول، می‌توانید ستونی را که دارای مقدار پیش فرض است نادیده بگیرید. در این مورد، PostgreSQL از مقدار پیش فرض برای درج استفاده می‌کند:

```
INSERT INTO table_name(column1, colum3)
VALUES(value1, value2);
```

اگر ستونی را با یک محدودیت پیش‌فرض در عبارت INSERT مشخص کرده‌اید و می‌خواهید از مقدار پیش‌فرض برای درج استفاده کنید، می‌توانید از کلمه کلیدی DEFAULT به صورت زیر استفاده کنید:

```
INSERT INTO table_name(column1, column2, colum3)
VALUES(value1,DEFAULT,value2);
```

مثال

In [60]:
```
%%sql
DROP TABLE IF EXISTS items;
CREATE TABLE
  items (
    item_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    quantity INT NOT NULL,
    price DEC(11, 2) NOT NULL,
    tax DEC(11, 2) DEFAULT 0.05
  );
```

Running query in 'default'

Out[60]:

In [61]:
```
%%sql
INSERT INTO
  items (name, quantity, price)
VALUES
  ('iPhone 15 Pro', 1, 1299.99) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[61]:

item_id	name	quantity	price	tax
1	iPhone 15 Pro	1	1299.99	0.05

In [62]:
```
%%sql
INSERT INTO
  items (name, quantity, price, tax)
VALUES
  ('iPhone 16 Pro', 1, 1399.99, DEFAULT) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[62]:

item_id	name	quantity	price	tax
2	iPhone 16 Pro	1	1399.99	0.05

In [63]:
```
%%sql
INSERT INTO
  items (name, quantity, price, tax)
VALUES
  ('iPhone 17 Pro', 1, 1499.99, 0.08) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[63]:

item_id	name	quantity	price	tax
3	iPhone 17 Pro	1	1499.99	0.08

مثال: تنظیم مقادیر پیش فرض برای ستون‌های زمانی

In [65]:
```
%%sql
DROP TABLE IF EXISTS orders_test;
CREATE TABLE
  orders_test (
    order_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    customer VARCHAR NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  );
```

Running query in 'default'

Out[65]:

In [66]:
```
%%sql
INSERT INTO
  orders_test (customer)
VALUES
  ('Milad Vazan') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[66]:

order_id	customer	created_at
1	Milad Vazan	2025-04-22 19:07:22.725779

مثال

In [67]:
```
%%sql
DROP TABLE IF EXISTS orders_test_2;
CREATE TABLE
  orders_test_2 (
    order_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    customer VARCHAR NOT NULL DEFAULT 'Unknown Customer',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    order_total DECIMAL(10, 2) DEFAULT 0.00
  );
```

Running query in 'default'

Out[67]:

In [68]:
```
%%sql
INSERT INTO
  orders_test_2 (customer)
VALUES
  ('Saman') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[68]:

order_id	customer	created_at	order_total
1	Saman	2025-04-22 19:07:48.886891	0.00

In [69]:
```
%%sql
INSERT INTO
  orders_test_2 (customer, order_total)
VALUES
  ('Sara', 100.00) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[69]:

order_id	customer	created_at	order_total
2	Sara	2025-04-22 19:07:49.432809	100.00

In [70]:
```
%%sql
-- درج داده‌ها با استفاده از مقادیر پیش‌فرض برای همه ستون‌ها
INSERT INTO
  orders_test_2 DEFAULT
VALUES
  RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[70]:

order_id	customer	created_at	order_total
3	Unknown Customer	2025-04-22 19:07:50.199593	0.00

In [71]:
```
%%sql
-- درج داده‌ها با استفاده از مقادیر مشخص برای همه ستون‌ها
INSERT INTO
  orders_test_2 (customer, created_at, order_total)
VALUES
  ('Ali', '2023-01-01 12:00:00', 200.00) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[71]:

order_id	customer	created_at	order_total
4	Ali	2023-01-01 12:00:00	200.00

افزودن محدودیت‌های DEFAULT به ستون‌های موجود

برای اضافه کردن محدودیت `DEFAULT` به ستونی از جدول موجود، از عبارت `ALTER TABLE` به صورت زیر استفاده می‌کنیم:

```
ALTER TABLE table_name
ALTER COLUMN column
SET DEFAULT default_value;
```

برای حذف کردن محدودیت `DEFAULT` از ستونی از جدول موجود، از عبارت `ALTER TABLE` به صورت زیر استفاده می‌کنیم:

```
ALTER TABLE table_name
ALTER COLUMN column
DROP DEFAULT;
```

In [72]:
```
%%sql
DROP TABLE IF EXISTS customers_test;
CREATE TABLE customers_test (
    id INT PRIMARY KEY,
    name VARCHAR,
    country VARCHAR);
```

Running query in 'default'

Out[72]:

In [73]:
```
%sqlcmd columns -t customers_test
```

Out[73]:

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	None	False	None
name	VARCHAR	True	None	False	None
country	VARCHAR	True	None	False	None

In [74]:
```
%%sql
--  افزودن یک مقدار پیش‌فرض برای ستون country
ALTER TABLE customers_test
ALTER COLUMN country
SET DEFAULT 'IRAN';
```

Running query in 'default'

Out[74]:

In [75]:
```
%sqlcmd columns -t customers_test
```

Out[75]:

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	None	False	None
name	VARCHAR	True	None	False	None
country	VARCHAR	True	'IRAN'::character varying	False	None

In [76]:
```
%%sql
--  حذف  مقدار پیش‌فرض از ستون country
ALTER TABLE customers_test
ALTER COLUMN country
DROP DEFAULT;
```

Running query in 'default'

Out[76]:

In [77]:
```
%sqlcmd columns -t customers_test
```

Out[77]:

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	None	False	None
name	VARCHAR	True	None	False	None
country	VARCHAR	True	None	False	None

محدودیت `CHECK`

در PostgreSQL، یک محدودیت `CHECK` اطمینان حاصل می‌کند که مقادیر در یک ستون یا گروهی از ستون‌ها یک شرط خاص را برآورده کنند. یک محدودیت `CHECK` از یک عبارت بولین استفاده می‌کند تا مقادیر را ارزیابی کند و اطمینان حاصل کند که تنها داده‌های معتبر در جدول درج یا به روز می‌شوند.

عبارت بولی چیست؟ عبارات بولی ترکیبی از عملگرهای منطقی AND، OR، NOT و همچنین عملگرهای مقایسه‌ای مانند =، >= یا <> هستند. نتیجه یک عبارت بولی یک مقدار بولی TRUE یا FALSE است.

عملگرهای مقایسه‌ای

عملگر	توضیح	مثال
`=`	برابر	`age = 18`
`<>`	نابرابر	`age <> 18`
`>`	بزرگتر	`age > 18`
`<`	کوچکتر	`age < 18`
`=>`	بزرگتر یا مساوی	`age >= 18`
`=<`	کوچکتر یا مساوی	`age <= 18`

عملگرهای منطقی

عملگر	توضیح	مثال
AND	و	age > 18 AND country = USA
OR	یا	age > 18 OR country = Canada
NOT	نه	NOT (age > 18)

به طور معمول، هنگام ایجاد جدول با استفاده از دستور CREATE TABLE، یک محدودیت CHECK ایجاد می‌کنید:

```
CREATE TABLE table_name(
   column1 datatype,
   ...,
   CONSTRAINT constraint_name CHECK(boolean_expression)
);
```

اگر محدودیت CHECK فقط یک ستون را شامل می‌شود، می‌توانید آن را به عنوان یک محدودیت ستونی مانند زیر تعریف کنید:

```
CREATE TABLE table_name(
   column1 datatype,
   column1 datatype CHECK(boolean_expression),
   ...,
);
```

به طور پیش فرض، PostgreSQL با استفاده از فرمت زیر، نامی را به یک محدودیت CHECK اختصاص می‌دهد:

```
{table}_{column}_check
```

شایان ذکر است که می‌توانید چندین محدودیت را در یک ستون اعمال کنید. به عنوان مثال، می توانید از هر دو محدودیت NOT NULL و CHECK برای یک ستون استفاده کنید:

```
CREATE TABLE table_name (
    column1 data_type NOT NULL CHECK(boolean_expression),
    ...
);
```

مثال‌ها

فرض کنید می‌خواهید یک جدول به نام `users` ایجاد کنید که داده‌های مربوط به کاربران را ذخیره کند. این جدول باید دارای ویژگی‌های زیر باشد:

1. هر کاربر دارای یک شناسه (`id`) باشد که به صورت خودکار مقداردهی می‌شود و کلید اصلی جدول است.
2. سن (`age`) هر کاربر باید عددی صحیح و غیر تهی باشد.
3. سن هر کاربر باید **دقیقاً 18 سال** باشد.

با استفاده از دستورات SQL، کدی بنویسید که این جدول را ایجاد کند.

In [78]:
```
%%sql
DROP TABLE users;
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    age INTEGER NOT NULL CHECK (age = 18)
);
```

Running query in 'default'

Out[78]:

In [79]:
```
%%sql

```

```sql
-- بدون خطا
INSERT INTO users (age) VALUES (18) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[79]:

id	age
1	18

In [80]:
```sql
%%sql

-- با خطا
INSERT INTO users (age) VALUES (19);
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "users" violates check constraint "users_age_check"
DETAIL:  Failing row contains (2, 19).

[SQL: INSERT INTO users (age) VALUES (19);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

فرض کنید می‌خواهید یک جدول به نام `employees` ایجاد کنید که داده‌های مربوط به کارکنان را ذخیره کند. این جدول باید دارای ویژگی‌های زیر باشد:

1. هر کارمند دارای یک شناسه (`id`) باشد که به صورت خودکار مقداردهی می‌شود و کلید اصلی جدول است.
2. سن (`age`) هر کارمند باید عددی صحیح و غیر تهی باشد.
3. حقوق (`salary`) هر کارمند باید عددی اعشاری باشد که حداکثر شامل **8 رقم قبل از ممیز** و **2 رقم بعد از ممیز** است و مقدار آن نمی‌تواند تهی باشد.
4. یک شرط منطقی برای داده‌ها اعمال شود:
   - اگر سن کارمند بیشتر از 18 سال باشد، حقوق او باید بیشتر از صفر باشد.
   - اگر سن کارمند کمتر از 18 سال باشد، حقوق او باید برابر با صفر باشد.

با استفاده از دستورات SQL، کدی بنویسید که این جدول را ایجاد کند.

In [81]:
```sql
%%sql
DROP TABLE IF EXISTS employees;
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    age INTEGER NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    CHECK ((age > 18 AND salary > 0) OR (age < 18 AND salary = 0))
);
```

Running query in 'default'

Out[81]:

In [82]:
```sql
%%sql
-- بدون خطا
INSERT INTO employees (age, salary) VALUES (19, 1000) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[82]:

id	age	salary
1	19	1000.00

In [83]:
```sql
%%sql
-- بدون خطا
INSERT INTO employees (age, salary) VALUES (17, 0) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[83]:

id	age	salary
2	17	0.00

In [84]:
```sql
%%sql
-- با خطا
INSERT INTO employees (age, salary) VALUES (17, 1000) RETURNING *;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "employees" violates check constraint "employees_check"
DETAIL:  Failing row contains (3, 17, 1000.00).

[SQL: INSERT INTO employees (age, salary) VALUES (17, 1000) RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

فرض کنید می‌خواهید یک جدول به نام `products` در پایگاه داده خود ایجاد کنید که اطلاعات مربوط به محصولات را ذخیره کند. این جدول باید دارای ویژگی‌های زیر باشد:

1. هر محصول دارای یک شناسه (`product_id`) باشد که به صورت خودکار مقداردهی می‌شود و کلید اصلی جدول است.
2. قیمت (`price`) هر محصول باید عددی اعشاری باشد که شامل حداکثر 10 رقم (8 رقم قبل از ممیز و 2 رقم بعد از ممیز) بوده و مقدار آن باید **بیشتر از صفر** باشد.
3. نام محصول (`name`) باید یک رشته متنی باشد که حداکثر 255 کاراکتر را ذخیره کند و مقدار آن نمی‌تواند تهی باشد.

با استفاده از دستورات SQL، کدی بنویسید که این جدول را ایجاد کند.

In [85]:
```
%%sql
DROP TABLE IF EXISTS products;
CREATE TABLE
  products (
    product_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    price DECIMAL(10, 2) CHECK (price > 0),
    name VARCHAR(255) NOT NULL
  );
```

Running query in 'default'

Out[85]:

In [86]:
```
%%sql
-- داده معتبر
INSERT INTO
  products (price, name)
VALUES
  (100.00, 'Laptop') RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[86]:

product_id	price	name
1	100.00	Laptop

In [87]:
```
%%sql
-- داده نامعتبر (قیمت منفی)
INSERT INTO
  products (price, name)
VALUES
  (-50.00, 'Tablet');
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "products" violates check constraint "products_price_check"
DETAIL:  Failing row contains (2, -50.00, Tablet).

[SQL: INSERT INTO
  products (price, name)
VALUES
  (-50.00, 'Tablet');]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

فرض کنید می‌خواهید یک جدول به نام `students` در پایگاه داده خود ایجاد کنید که اطلاعات مربوط به دانش‌آموزان را ذخیره کند. این جدول باید دارای ویژگی‌های زیر باشد:

1. هر دانش‌آموز دارای یک شناسه (`student_id`) باشد که به صورت خودکار مقداردهی می‌شود و کلید اصلی جدول است.
2. سن (`age`) هر دانش‌آموز به صورت عدد صحیح ذخیره شود و مقدار آن می‌تواند تهی باشد.
3. معدل (`grade_point_average`) هر دانش‌آموز باید یک عدد اعشاری باشد که شامل حداکثر 3 رقم (1 رقم قبل از ممیز و 2 رقم بعد از ممیز) بوده و مقدار آن باید بین **0** تا **4** با

با استفاده از دستورات SQL، کدی بنویسید که این جدول را ایجاد کند. کند.

In [88]:
```
%%sql
DROP TABLE IF EXISTS students;
CREATE TABLE
  students (
    student_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    age INT,
    grade_point_average DECIMAL(3, 2) CHECK (
      grade_point_average >= 0
      AND grade_point_average <= 4
    )
  );
```

Running query in 'default'

Out[88]:

In [89]:
```
%%sql
-- داده معتبر
INSERT INTO
  students (age, grade_point_average)
```

```
VALUES
  (15, 3.50);
```

Running query in 'default'

1 rows affected.

Out[89]:

In [90]:
```
%%sql
-- داده نامعتبر (میانگین نمره منفی)
INSERT INTO
  students (age, grade_point_average)
VALUES
  (12, -0.50);

-- خطا
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "students" violates check constraint "students_grade_point_
average_check"
DETAIL:  Failing row contains (2, 12, -0.50).

[SQL: INSERT INTO
  students (age, grade_point_average)
VALUES
  (12, -0.50);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [91]:
```
%%sql
-- داده نامعتبر (میانگین نمره بیش از 4)
INSERT INTO
  students (age, grade_point_average)
VALUES
  (18, 4.50);

-- خطا
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "students" violates check constraint "students_grade_point_
average_check"
DETAIL:  Failing row contains (3, 18, 4.50).

[SQL: INSERT INTO
  students (age, grade_point_average)
VALUES
  (18, 4.50);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [92]:
```
%%sql
-- داده معتبر (میانگین نمره 4)
INSERT INTO
  students (age, grade_point_average)
VALUES
  (16, 4.00);
```

Running query in 'default'

1 rows affected.

Out[92]:

In [93]:
```
%%sql
DROP TABLE students;
```

Running query in 'default'

Out[93]:

فرض کنید می‌خواهید یک جدول به نام `students` ایجاد کنید که داده‌های مربوط به دانش‌آموزان را ذخیره کند. این جدول باید دارای ویژگی‌های زیر باشد:

1. هر دانش‌آموز دارای یک شناسه (`student_id`) باشد که به صورت خودکار مقداردهی می‌شود و کلید اصلی جدول است.
2. سن (`age`) هر دانش‌آموز باید یک عدد صحیح باشد و مقدار آن باید بین **10 تا 18 سال** باشد.
3. معدل (`grade_point_average`) هر دانش‌آموز باید یک عدد اعشاری باشد که شامل حداکثر 3 رقم (1 رقم قبل از ممیز و 2 رقم بعد از ممیز) بوده و مقدار آن باید بین **0 تا 4** باشد.
4. ترم (`semester`) هر دانش‌آموز باید یک عدد صحیح باشد و مقدار آن فقط می‌تواند **1 یا 2** باشد.

با استفاده از دستورات SQL، کدی بنویسید که این جدول را ایجاد کند.

در SQL، عملگر `IN` به شما امکان می‌دهد تا یک مقدار را با لیستی از مقادیر مقایسه کنید. اگر مقدار مورد نظر در لیست قرار داشته باشد، نتیجه مقایسه **درست** (`TRUE`) خواهد بود:

```
expression IN (value1, value2, ..., valueN)
```

این عملگر به شما می‌گوید که ایا expression یکی از مقادیر موجود در لیست است یا خیر.

In [94]:
```
%%sql
CREATE TABLE
  students (
    student_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    age INT CHECK (age >= 10 AND age <= 18),
    grade_point_average DECIMAL(3, 2) CHECK ( grade_point_average >= 0 AND grade_point_average <= 4),
    semester INT CHECK (semester IN (1, 2))
  );
```

Running query in 'default'

Out[94]:

معادل کد بالا

```
CREATE TABLE students (
  student_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  age INT,
  grade_point_average DECIMAL(3, 2),
  semester INT,
  CONSTRAINT valid_data CHECK (
    age >= 10
    AND age <= 18
    AND grade_point_average >= 0
    AND grade_point_average <= 4
    AND semester IN (1, 2)
  )
);
```

In [95]:
```
%%sql
-- داده معتبر
INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (15, 3.20, 1) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[95]:

student_id	age	grade_point_average	semester
1	15	3.20	1

In [96]:
```
%%sql
-- داده نامعتبر (سن کمتر از 10 سال)
INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (8, 3.20, 1) RETURNING *;

-- خطا
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "students" violates check constraint "students_age_check"
DETAIL:  Failing row contains (2, 8, 3.20, 1).


[SQL: INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (8, 3.20, 1) RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [97]:
```
%%sql
-- داده نامعتبر (میانگین نمره بیش از 4)
INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (15, 4.50, 1) RETURNING *;

-- خطا
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "students" violates check constraint "students_grade_point_
average_check"
DETAIL:  Failing row contains (3, 15, 4.50, 1).

[SQL: INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (15, 4.50, 1) RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [98]:
```sql
%%sql
-- داده نامعتبر (ترم 3)
INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (15, 3.20, 3) RETURNING *;

-- خطا
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "students" violates check constraint "students_semester_che
ck"
DETAIL:  Failing row contains (4, 15, 3.20, 3).

[SQL: INSERT INTO
  students (age, grade_point_average, semester)
VALUES
  (15, 3.20, 3) RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

فرض کنید می‌خواهید یک جدول به نام `students` ایجاد کنید که داده‌های مربوط به دانش‌آموزان را ذخیره کند. این جدول باید دارای ویژگی‌های زیر باشد:

1. هر دانش‌آموز دارای یک شناسه (`student_id`) باشد که به صورت خودکار مقداردهی می‌شود و کلید اصلی جدول است.
2. نام (`name`) هر دانش‌آموز باید یک رشته متنی باشد.
3. ایمیل (`email`) و تلفن (`phone`) هر دانش‌آموز به صورت رشته‌های متنی ذخیره شوند. **حداقل یکی از این دو باید پر شود.**

در SQL، عملگر `IS` به شما امکان می‌دهد تا یک مقدار را به عنوان **NULL** یا **NOT NULL** بررسی کنید. این عملگر به دو صورت استفاده می‌شود:

1. **IS NULL**: این عملگر بررسی می‌کند که آیا یک ستون یا عبارت **NULL** است یا خیر.
2. **IS NOT NULL**: این عملگر بررسی می‌کند که آیا یک ستون یا عبارت **غیر NULL** است یا خیر.

در SQL، نمی‌توانید از عملگرهای مقایسه‌ای مانند `=` یا `<>` برای مقایسه با **NULL** استفاده کنید. به جای آن، از `IS NULL` یا `IS NOT NULL` استفاده می‌کنید.

In [100…
```sql
%%sql
DROP TABLE IF EXISTS students;
CREATE TABLE students (
  student_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  name VARCHAR,
  email VARCHAR,
  phone VARCHAR,
  CONSTRAINT valid_contact CHECK (
    email IS NOT NULL OR phone IS NOT NULL
  )
);
```

Running query in 'default'

Out[100…

In [101…
```sql
%%sql
--  ایمیل وارد شده و تلفن خالی است
INSERT INTO students (name, email, phone)
VALUES ('Milad Vazan', 'm_vazan@sbu.ac.ir', NULL) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[101…

student_id	name	email	phone
1	Milad Vazan	m_vazan@sbu.ac.ir	None

In [102…
```sql
%%sql
--  تلفن وارد شده و ایمیل خالی است
INSERT INTO students (name, email, phone)
VALUES ('Milad Vazan', NULL, 0911495) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[102…

student_id	name	email	phone
2	Milad Vazan	None	911495

In [103…
```sql
%%sql
-- هر دو ایمیل و تلفن وارد شده‌اند
```

```
INSERT INTO students (name, email, phone)
VALUES ('Milad Vazan', 'm_vazan@sbu.ac.ir', 0937017) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[103…

student_id	name	email	phone
3	Milad Vazan	m_vazan@sbu.ac.ir	937017

In [104…

```
%%sql
-- هیچ‌کدام وارد نشده‌اند (این حالت خطا می‌دهد)
INSERT INTO students (name, email, phone)
VALUES ('Milad Vazan', NULL, NULL) RETURNING *;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "students" violates check constraint "valid_contact"
DETAIL:  Failing row contains (4, Milad Vazan, null, null).

[SQL: INSERT INTO students (name, email, phone)
VALUES ('Milad Vazan', NULL, NULL) RETURNING *;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

سوال ۱: استفاده از = و AND

سوال: یک جدول `users` با ستون‌های `age` , `id` , و `is_active` ایجاد کنید. محدودیت **CHECK** را طوری تنظیم کنید که فقط کاربران با سن **18** و `is_active = TRUE` بتوانند ثبت شوند. یک رکورد با سن **18** و `is_active = TRUE` و یک رکورد با سن **19** و `is_active = TRUE` را درج کنید.

In [ ]:

```
%%sql
DROP TABLE users;
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    age INTEGER NOT NULL,
    is_active BOOLEAN NOT NULL,
    CHECK (age = 18 AND is_active = TRUE)
);
```

In [ ]:

```
%%sql
-- بدون خطا
INSERT INTO users (age, is_active) VALUES (18, TRUE) RETURNING *;
```

In [ ]:

```
%%sql
-- با خطا
INSERT INTO users (age, is_active) VALUES (19, TRUE) RETURNING *;
```

سوال 2: استفاده از <> و AND

سوال: یک جدول `customers` با ستون‌های `country` , `id` , و `is_active` ایجاد کنید. محدودیت **CHECK** را طوری تنظیم کنید که فقط مشتریان با کشور غیر از **'USA'** و `is_active = TRUE` بتوانند ثبت شوند. یک رکورد با کشور **'Canada'** و `is_active = TRUE` را درج کنید و سپس یک رکورد با کشور **'USA'** و `is_active = TRUE` را درج کنید.

In [ ]:

```
%%sql
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    country VARCHAR(50) NOT NULL,
    is_active BOOLEAN NOT NULL,
    CHECK (country <> 'USA' AND is_active = TRUE)
);
```

In [ ]:

```
%%sql
-- بدون خطا
INSERT INTO customers (country, is_active) VALUES ('Canada', TRUE) RETURNING *;
```

In [ ]:

```
%%sql
-- با خطا
INSERT INTO customers (country, is_active) VALUES ('USA', TRUE);
```

سوال 3: استفاده از <= , AND , و NOT

سوال: یک جدول `employees` با ستون‌های `salary` , `age` , `id` , و `is_manager` ایجاد کنید. محدودیت **CHECK** را طوری تنظیم کنید که کارمندان با سن **30** یا بیشتر و حقوق **50000** یا بیشتر و `is_manager = TRUE` یا کارمندان با سن کمتر از **30** و حقوق کمتر از **50000** و `is_manager = FALSE` بتوانند ثبت شوند. همچنین اطمینان حاصل کنید که کارمندان با سن **30** یا بیشتر و حقوق کمتر از **50000** و `is_manager = TRUE` هرگز ثبت نشوند. یک رکورد با سن **30**, حقوق **50000**, و `is_manager = TRUE` را درج کنید و سپس یک رکورد با سن **30**, حقوق **49999**, و `is_manager = TRUE` را درج کنید.

In [105…

```
%%sql
DROP TABLE employees;
```

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    age INTEGER NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    is_manager BOOLEAN NOT NULL,
    CHECK (((age >= 30 AND salary >= 50000 AND is_manager = TRUE) OR
             (age < 30 AND salary < 50000 AND is_manager = FALSE)) AND
             NOT (age >= 30 AND salary < 50000 AND is_manager = TRUE))
);
```

Running query in 'default'

Out[105…

In [106…

```
%%sql
-- بدون خطا
INSERT INTO employees (age, salary, is_manager) VALUES (30, 50000, TRUE) RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[106…

id	age	salary	is_manager
1	30	50000.00	True

In [107…

```
%%sql
-- با خطا
INSERT INTO employees (age, salary, is_manager) VALUES (30, 49999, TRUE);
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.CheckViolation) new row for relation "employees" violates check constraint "employees_check"
DETAIL:  Failing row contains (2, 30, 49999.00, t).

[SQL: INSERT INTO employees (age, salary, is_manager) VALUES (30, 49999, TRUE);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های آموزشی آمار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

شنبه ـ ۳۰ فروردین ۱۴۰۴

Saturday - 2025 19 April

معرفی دستور ALTER TABLE در PostgreSQL

برای تغییر ساختار یک جدول موجود، از دستور `ALTER TABLE` در PostgreSQL استفاده می‌شود.

ساختار اساسی دستور `ALTER TABLE` به شرح زیر است:

```
ALTER TABLE table_name action;
```

PostgreSQL به شما امکان انجام بسیاری از اقدامات را می‌دهد:

- **اضافه کردن یک ستون**
- **حذف یک ستوون**
- **تغییر نام یک ستون**
- **تغییر نام یک جدول**

برای اضافه کردن یک ستون جدید به جدول، از دستور `ALTER TABLE ADD COLUMN` استفاده می‌شود:

```
ALTER TABLE table_name
ADD COLUMN column_name datatype column_constraint;
```

برای حذف یک ستون از جدول، از دستور `ALTER TABLE DROP COLUMN` استفاده کنید:

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

برای تغییر نام یک ستون، از دستور زیر استفاده کنید:

```
ALTER TABLE table_name
RENAME COLUMN column_name
```

```
TO new_column_name;
```

برای تغییر نام یک جدول، از دستور زیر استفاده می‌شود:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

مثال

In [108…
```
%%sql
DROP TABLE IF EXISTS employees;
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR,
    email VARCHAR
);
```

Running query in 'default'

Out[108…

In [109…
```
%sqlcmd columns -t employees
```

Out[109…

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('employees_id_seq1'::regclass)	True	None
name	VARCHAR	True	None	False	None
email	VARCHAR	True	None	False	None

اضافه کردن یک ستون جدید

In [110…
```
%%sql
ALTER TABLE employees
ADD COLUMN department VARCHAR;
```

Running query in 'default'

Out[110…

In [111…
```
%sqlcmd columns -t employees
```

Out[111…

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('employees_id_seq1'::regclass)	True	None
name	VARCHAR	True	None	False	None
email	VARCHAR	True	None	False	None
department	VARCHAR	True	None	False	None

حذف یک ستون

In [112…
```
%%sql
ALTER TABLE employees
DROP COLUMN email;
```

Running query in 'default'

Out[112…

In [113…
```
%sqlcmd columns -t employees
```

Out[113…

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('employees_id_seq1'::regclass)	True	None
name	VARCHAR	True	None	False	None
department	VARCHAR	True	None	False	None

تغییر نام یک ستون

In [114…
```
%%sql
ALTER TABLE employees
RENAME COLUMN name
TO full_name;
```

Running query in 'default'

Out[114...

In [115...
```
%sqlcmd columns -t employees
```

Out[115...

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('employees_id_seq1'::regclass)	True	None
full_name	VARCHAR	True	None	False	None
department	VARCHAR	True	None	False	None

تغییر نام جدول

In [116...
```
%%sql
ALTER TABLE employees
RENAME TO staff;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.DuplicateTable) relation "staff" already exists

[SQL: ALTER TABLE employees
RENAME TO staff;]
(Background on this error at: https://sqlalche.me/e/20/f405)
```

In [119...
```
%sqlcmd columns -t staff
```

Out[119...

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('employees_id_seq'::regclass)	True	None
full_name	VARCHAR	True	None	False	None
department	VARCHAR	True	None	False	None

پرسمان داده‌ها با دستور `SELECT`

برای بازیابی داده‌ها از یک جدول، از عبارت SELECT استفاده می‌شود.

نحو پایه دستور SELECT به صورت زیر است:

```
SELECT column1, column2, ...
FROM table_name;
```

در این نحو:

1. **لیست ستون‌ها**: پس از کلمه کلیدی `SELECT` ، یک یا چند ستون از جدول را برای بازیابی داده مشخص کنید. برای جدا کردن چندین ستون، از کاما استفاده کنید.
2. **نام جدول**: پس از کلمه کلیدی `FROM` ، نام جدول را مشخص کنید.

در بخش `SELECT` می‌توانید داده را از یک ستون بازیابی کنید:

```
SELECT column1
FROM table_name;
```

یا داده را از چندین ستون بازیابی کنید:

```
SELECT column1, column2
FROM table_name;
```

از `SELECT *` برای بازیابی داده از همه ستون‌ها استفاده کنید:

```
SELECT *
FROM table_name;
```

In [47]:
```
%%sql
DROP TABLE IF EXISTS student;
CREATE TABLE student (
    first_name TEXT,
    last_name TEXT,
    age INTEGER,
    grade TEXT,
    major TEXT,
```

```
    gpa REAL,
    student_id INTEGER
);


INSERT INTO student (first_name, last_name, age, grade, major, gpa, student_id)
VALUES
('علی', 'رضایی', 20, 'کارشناسی', 'مهندسی', 16.5, 1001),
('سارا', 'احمدی', 22, 'کارشناسی ارشد', 'پزشکی', 18.2, 1002),
('مهدی', 'حسینی', 19, 'کارشناسی', 'حقوق', 14.8, 1003),
('ندا', 'کریمی', 21, 'کارشناسی ارشد', 'بازرگانی', 17.5, 1004),
('امیر', 'شیرازی', 23, 'دکتری', 'فیزیک', 19.5, 1005),
('لیلا', 'جعفری', 20, 'کارشناسی', 'هنر', 15.9, 1006),
('حسین', 'غفاری', 24, 'دکتری', 'شیمی', 18.8, 1007),
('مریم', 'ابراهیمی', 22, 'کارشناسی ارشد', 'زیست‌شناسی', 18.5, 1008),
('رضا', 'محمدی', 19, 'کارشناسی', 'علوم کامپیوتر', 15.2, 1009),
('فاطمه', 'رمضانی', 21, 'کارشناسی ارشد', 'ریاضی', 17.8, 1010),
('سینا', 'شاهبازی', 20, 'کارشناسی', 'تاریخ', 16.1, 1011),
('زهرا', 'خلیلی', 22, 'کارشناسی ارشد', 'ادبیات', 18.2, 1012),
('پویا', 'فرهادی', 19, 'کارشناسی', 'اقتصاد', 14.5, 1013),
('رویا', 'عزیزی', 21, 'کارشناسی ارشد', 'روان‌شناسی', 17.9, 1014),
('کیان', 'بهرامی', 23, 'دکتری', 'اخترشناسی', 19.8, 1015),
('شیرین', 'نصیری', 20, 'کارشناسی', 'جامعه‌شناسی', 15.6, 1016),
('بابک', 'شکوهی', 24, 'دکتری', 'زمین‌شناسی', 18.9, 1017),
('افسانه', 'رحیمی', 22, 'کارشناسی ارشد', 'فلسفه', 18.6, 1018),
('نوید', 'خسروی', 19, 'کارشناسی', 'علوم سیاسی', 15.3, 1019),
('هما', 'صادقی', 21, 'کارشناسی ارشد', 'انسان‌شناسی', 17.7, 1020),
('میلاد', 'وزان', 32, 'دکتری', 'علوم کامپیوتر', 18.96, 1021);
```

Running query in 'default'

21 rows affected.

Out[47]:

In [121…

```
%%sql
select * from student;
```

Running query in 'default'

21 rows affected.

Out[121…

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
حسین	غفاری	24	دکتری	شیمی	18.8	1007
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021

In [122…

```
%%sql
DROP TABLE IF EXISTS student_english;
CREATE TABLE student_english (
    first_name TEXT,
    last_name TEXT,
    age INTEGER,
    grade TEXT,
    major TEXT,
```

```
    gpa REAL,
    student_id INTEGER
);


INSERT INTO student_english (first_name, last_name, age, grade, major, gpa, student_id)
VALUES
('Ali', 'Rezaei', 20, 'BSc', 'Engineering', 16.5, 1001),
('Sara', 'Ahmadi', 22, 'MSc', 'Medicine', 18.2, 1002),
('Mehdi', 'Hosseini', 19, 'BSc', 'Law', 14.8, 1003),
('Neda', 'Karimi', 21, 'MSc', 'Business', 17.5, 1004),
('Amir', 'Shirazi', 23, 'PhD', 'Physics', 19.5, 1005),
('Leila', 'Jafari', 20, 'BSc', 'Arts', 15.9, 1006),
('Hossein', 'Ghaffari', 24, 'PhD', 'Chemistry', 18.8, 1007),
('Maryam', 'Ebrahimi', 22, 'MSc', 'Biology', 18.5, 1008),
('Reza', 'Mohammadi', 19, 'BSc', 'Computer Science', 15.2, 1009),
('Fatemeh', 'Ramezani', 21, 'MSc', 'Mathematics', 17.8, 1010),
('Sina', 'Shahbazi', 20, 'BSc', 'History', 16.1, 1011),
('Zahra', 'Khalili', 22, 'MSc', 'Literature', 18.2, 1012),
('Pouya', 'Farhadi', 19, 'BSc', 'Economics', 14.5, 1013),
('Roya', 'Azizi', 21, 'MSc', 'Psychology', 17.9, 1014),
('Kian', 'Bahrami', 23, 'PhD', 'Astronomy', 19.8, 1015),
('Shirin', 'Nassiri', 20, 'BSc', 'Sociology', 15.6, 1016),
('Babak', 'Shokouhi', 24, 'PhD', 'Geology', 18.9, 1017),
('Afsaneh', 'Rahimi', 22, 'MSc', 'Philosophy', 18.6, 1018),
('Navid', 'Khosravi', 19, 'BSc', 'Political Science', 15.3, 1019),
('Homa', 'Sadeghi', 21, 'MSc', 'Anthropology', 17.7, 1020),
('Milad', 'Vazan', 32, 'PhD', 'Computer Science', 18.96, 1021);
```

Running query in 'default'

21 rows affected.

Out[122…

In [123…

```
%%sql
select * from student_english;
```

Running query in 'default'

21 rows affected.

Out[123…

first_name	last_name	age	grade	major	gpa	student_id
Ali	Rezaei	20	BSc	Engineering	16.5	1001
Sara	Ahmadi	22	MSc	Medicine	18.2	1002
Mehdi	Hosseini	19	BSc	Law	14.8	1003
Neda	Karimi	21	MSc	Business	17.5	1004
Amir	Shirazi	23	PhD	Physics	19.5	1005
Leila	Jafari	20	BSc	Arts	15.9	1006
Hossein	Ghaffari	24	PhD	Chemistry	18.8	1007
Maryam	Ebrahimi	22	MSc	Biology	18.5	1008
Reza	Mohammadi	19	BSc	Computer Science	15.2	1009
Fatemeh	Ramezani	21	MSc	Mathematics	17.8	1010
Sina	Shahbazi	20	BSc	History	16.1	1011
Zahra	Khalili	22	MSc	Literature	18.2	1012
Pouya	Farhadi	19	BSc	Economics	14.5	1013
Roya	Azizi	21	MSc	Psychology	17.9	1014
Kian	Bahrami	23	PhD	Astronomy	19.8	1015
Shirin	Nassiri	20	BSc	Sociology	15.6	1016
Babak	Shokouhi	24	PhD	Geology	18.9	1017
Afsaneh	Rahimi	22	MSc	Philosophy	18.6	1018
Navid	Khosravi	19	BSc	Political Science	15.3	1019
Homa	Sadeghi	21	MSc	Anthropology	17.7	1020
Milad	Vazan	32	PhD	Computer Science	18.96	1021

In [124…

```
%%sql
select last_name, major, gpa from student;
```

Running query in 'default'

21 rows affected.

Out[124...

last_name	major	gpa
رضایی	مهندسی	16.5
احمدی	پزشکی	18.2
حسینی	حقوق	14.8
کریمی	بازرگانی	17.5
شیرازی	فیزیک	19.5
جعفری	هنر	15.9
غفاری	شیمی	18.8
ابراهیمی	زیست‌شناسی	18.5
محمدی	علوم کامپیوتر	15.2
رمضانی	ریاضی	17.8
شاهبازی	تاریخ	16.1
خلیلی	ادبیات	18.2
فرهادی	اقتصاد	14.5
عزیزی	روان‌شناسی	17.9
بهرامی	اخترشناسی	19.8
نصیری	جامعه‌شناسی	15.6
شکوهی	زمین‌شناسی	18.9
رحیمی	فلسفه	18.6
خسروی	علوم سیاسی	15.3
صادقی	انسان‌شناسی	17.7
وزان	علوم کامپیوتر	18.96

فیلتر کردن ردیف‌های غیرجالب

اغلب ما فقط به زیر مجموعه‌ای از داده‌های موجود علاقه‌مندیم. یعنی، حتی اگر ممکن است داده‌هایی درباره افراد یا چیزهای زیادی داشته باشیم، اغلب فقط می‌خواهیم داده‌هایی را که در مورد افراد یا چیزهای بسیار خاص داریم، ببینیم. اینجاست که شرط `WHERE` به کار می‌آید. این به ما امکان می‌دهد تا مشخص کنیم که به کدام ردیف‌های خاص از جدول خود علاقه‌مندیم. نحو آن در زیرآمده است:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

زمانی که یک عبارت (statement) در PostgreSQL اجرا می‌شود، ترتیب ارزیابی بخش‌های مختلف آن به صورت زیر است:

1. ابتدا بخش FROM اجرا می‌شود تا هر ردیف (row) از جدول مشخص شده در FROM بازیابی شود.
2. سپس شرط موجود در بخش WHERE ارزیابی می‌شود و تنها ردیف‌هایی که شرط در آنها درست (true) باشد، وارد مجموعه نتایج می‌شوند.
3. در نهایت، ستون‌های مشخص شده در بخش SELECT از ردیف‌های انتخاب شده استخراج می‌شوند.

به عبارت دیگر، PostgreSQL ابتدا داده‌ها را از جدول می‌گیرد، سپس آنها را بر اساس شرط فیلتر می‌کند و در پایان ستون‌های مورد نظر را نمایش می‌دهد.

سوال؟

دانشجویانی که معدل بالاتر از ۱۸ دارند را پیدا کن.

In [125...

```
%%sql
SELECT * FROM student
WHERE gpa > 18;
```

Running query in 'default'

9 rows affected.

Out[125...

first_name	last_name	age	grade	major	gpa	student_id
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
حسین	غفاری	24	دکتری	شیمی	18.8	1007
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021

سوال؟

دانشجویانی که در مقطع کارشناسی تحصیل می‌کنند و سنشان کمتر از ۲۱ سال است را نمایش بده.

In [126...

```
%%sql
SELECT * FROM student
WHERE grade = 'کارشناسی' AND age < 21;
```

Running query in 'default'

8 rows affected.

Out[126...

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019

سوال؟

دانشجویانی که رشته آن‌ها `علوم کامپیوتر` یا `فیزیک` است را نمایش بده.

In [127...

```
%%sql
SELECT * FROM student
WHERE major = 'علوم کامپیوتر' OR major = 'فیزیک';
```

Running query in 'default'

3 rows affected.

Out[127...

first_name	last_name	age	grade	major	gpa	student_id
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021

سوال؟

دانشجویانی که در مقطع دکتری هستند و معدلشان کمتر از ۱۹ است را نمایش بده.

In [128...

```
%%sql
SELECT * FROM student
WHERE grade = 'دکتری' AND gpa < 19;
```

Running query in 'default'

3 rows affected.

Out[128…

first_name	last_name	age	grade	major	gpa	student_id
حسین	غفاری	24	دکتری	شیمی	18.8	1007
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021

سوال؟

دانشجویانی که در یکی از رشته‌های زیر تحصیل می‌کنند را نمایش بده: `Medicine` , `Law` , `Business`

In [129…

```sql
%%sql
SELECT * FROM student_english
WHERE major IN ('Medicine', 'Law', 'Business');
```

Running query in 'default'

3 rows affected.

Out[129…

first_name	last_name	age	grade	major	gpa	student_id
Sara	Ahmadi	22	MSc	Medicine	18.2	1002
Mehdi	Hosseini	19	BSc	Law	14.8	1003
Neda	Karimi	21	MSc	Business	17.5	1004

سوال؟

دانشجویانی که در یکی از مقاطع `MSc` یا `PhD` هستند را نمایش بده.

In [130…

```sql
%%sql
SELECT * FROM student_english
WHERE grade IN ('MSc', 'PhD');
```

Running query in 'default'

13 rows affected.

Out[130…

first_name	last_name	age	grade	major	gpa	student_id
Sara	Ahmadi	22	MSc	Medicine	18.2	1002
Neda	Karimi	21	MSc	Business	17.5	1004
Amir	Shirazi	23	PhD	Physics	19.5	1005
Hossein	Ghaffari	24	PhD	Chemistry	18.8	1007
Maryam	Ebrahimi	22	MSc	Biology	18.5	1008
Fatemeh	Ramezani	21	MSc	Mathematics	17.8	1010
Zahra	Khalili	22	MSc	Literature	18.2	1012
Roya	Azizi	21	MSc	Psychology	17.9	1014
Kian	Bahrami	23	PhD	Astronomy	19.8	1015
Babak	Shokouhi	24	PhD	Geology	18.9	1017
Afsaneh	Rahimi	22	MSc	Philosophy	18.6	1018
Homa	Sadeghi	21	MSc	Anthropology	17.7	1020
Milad	Vazan	32	PhD	Computer Science	18.96	1021

سوال؟

دانشجویانی که نام کوچکشان یکی از موارد زیر است را نمایش بده: `Ali` , `Leila` , `Roya`

In [131…

```sql
%%sql
SELECT * FROM student_english
WHERE first_name IN ('Ali', 'Leila', 'Roya');
```

Running query in 'default'

3 rows affected.

Out[131…

first_name	last_name	age	grade	major	gpa	student_id
Ali	Rezaei	20	BSc	Engineering	16.5	1001
Leila	Jafari	20	BSc	Arts	15.9	1006
Roya	Azizi	21	MSc	Psychology	17.9	1014

سوال؟

دانشجویانی که در رشته‌هایی غیر از `Medicine` ، `Engineering` ، و `Law` تحصیل می‌کنند.

In [132…

```sql
%%sql
SELECT * FROM student_english
WHERE major NOT IN ('Engineering', 'Medicine', 'Law');
```

Running query in 'default'

18 rows affected.

Out[132…

first_name	last_name	age	grade	major	gpa	student_id
Neda	Karimi	21	MSc	Business	17.5	1004
Amir	Shirazi	23	PhD	Physics	19.5	1005
Leila	Jafari	20	BSc	Arts	15.9	1006
Hossein	Ghaffari	24	PhD	Chemistry	18.8	1007
Maryam	Ebrahimi	22	MSc	Biology	18.5	1008
Reza	Mohammadi	19	BSc	Computer Science	15.2	1009
Fatemeh	Ramezani	21	MSc	Mathematics	17.8	1010
Sina	Shahbazi	20	BSc	History	16.1	1011
Zahra	Khalili	22	MSc	Literature	18.2	1012
Pouya	Farhadi	19	BSc	Economics	14.5	1013
Roya	Azizi	21	MSc	Psychology	17.9	1014
Kian	Bahrami	23	PhD	Astronomy	19.8	1015
Shirin	Nassiri	20	BSc	Sociology	15.6	1016
Babak	Shokouhi	24	PhD	Geology	18.9	1017
Afsaneh	Rahimi	22	MSc	Philosophy	18.6	1018
Navid	Khosravi	19	BSc	Political Science	15.3	1019
Homa	Sadeghi	21	MSc	Anthropology	17.7	1020
Milad	Vazan	32	PhD	Computer Science	18.96	1021

سوال؟

دانشجویانی که در مقطع `BSc` هستند ولی رشته‌شون در بین `Arts` ، `Computer Science` ، و `History` نیست.

In [133…

```sql
%%sql
SELECT * FROM student_english
WHERE grade = 'BSc' AND major NOT IN ('Computer Science', 'Arts', 'History');
```

Running query in 'default'

5 rows affected.

Out[133…

first_name	last_name	age	grade	major	gpa	student_id
Ali	Rezaei	20	BSc	Engineering	16.5	1001
Mehdi	Hosseini	19	BSc	Law	14.8	1003
Pouya	Farhadi	19	BSc	Economics	14.5	1013
Shirin	Nassiri	20	BSc	Sociology	15.6	1016
Navid	Khosravi	19	BSc	Political Science	15.3	1019

سوال؟

دانشجویانی که سن آن‌ها غیر از ۲۰، ۲۱ و ۲۲ سال است.

In [134…
```sql
%%sql
SELECT * FROM student_english
WHERE age NOT IN (20, 21, 22);
```

Running query in 'default'

9 rows affected.

Out[134…

first_name	last_name	age	grade	major	gpa	student_id
Mehdi	Hosseini	19	BSc	Law	14.8	1003
Amir	Shirazi	23	PhD	Physics	19.5	1005
Hossein	Ghaffari	24	PhD	Chemistry	18.8	1007
Reza	Mohammadi	19	BSc	Computer Science	15.2	1009
Pouya	Farhadi	19	BSc	Economics	14.5	1013
Kian	Bahrami	23	PhD	Astronomy	19.8	1015
Babak	Shokouhi	24	PhD	Geology	18.9	1017
Navid	Khosravi	19	BSc	Political Science	15.3	1019
Milad	Vazan	32	PhD	Computer Science	18.96	1021

سوال؟

دانشجویانی که در مقاطع بالا (یعنی `PhD` یا `MSc`) هستند، اما رشته‌شون در بین `Astronomy` , `Philosophy` , یا `Mathematics` نیست.

In [135…
```sql
%%sql
SELECT * FROM student_english
WHERE grade IN ('MSc', 'PhD') AND major NOT IN ('Philosophy', 'Astronomy', 'Mathematics');
```

Running query in 'default'

10 rows affected.

Out[135…

first_name	last_name	age	grade	major	gpa	student_id
Sara	Ahmadi	22	MSc	Medicine	18.2	1002
Neda	Karimi	21	MSc	Business	17.5	1004
Amir	Shirazi	23	PhD	Physics	19.5	1005
Hossein	Ghaffari	24	PhD	Chemistry	18.8	1007
Maryam	Ebrahimi	22	MSc	Biology	18.5	1008
Zahra	Khalili	22	MSc	Literature	18.2	1012
Roya	Azizi	21	MSc	Psychology	17.9	1014
Babak	Shokouhi	24	PhD	Geology	18.9	1017
Homa	Sadeghi	21	MSc	Anthropology	17.7	1020
Milad	Vazan	32	PhD	Computer Science	18.96	1021

سوال؟

دانشجویانی که در رشته‌هایی غیر از `Sociology` , `Economics` , و `Political Science` هستند و معدل بالای ۱۵ دارند.

In [136…
```sql
%%sql
SELECT * FROM student_english
WHERE major NOT IN ('Economics', 'Sociology', 'Political Science') AND gpa > 15;
```

Running query in 'default'

17 rows affected.

Out[136...

first_name	last_name	age	grade	major	gpa	student_id
Ali	Rezaei	20	BSc	Engineering	16.5	1001
Sara	Ahmadi	22	MSc	Medicine	18.2	1002
Neda	Karimi	21	MSc	Business	17.5	1004
Amir	Shirazi	23	PhD	Physics	19.5	1005
Leila	Jafari	20	BSc	Arts	15.9	1006
Hossein	Ghaffari	24	PhD	Chemistry	18.8	1007
Maryam	Ebrahimi	22	MSc	Biology	18.5	1008
Reza	Mohammadi	19	BSc	Computer Science	15.2	1009
Fatemeh	Ramezani	21	MSc	Mathematics	17.8	1010
Sina	Shahbazi	20	BSc	History	16.1	1011
Zahra	Khalili	22	MSc	Literature	18.2	1012
Roya	Azizi	21	MSc	Psychology	17.9	1014
Kian	Bahrami	23	PhD	Astronomy	19.8	1015
Babak	Shokouhi	24	PhD	Geology	18.9	1017
Afsaneh	Rahimi	22	MSc	Philosophy	18.6	1018
Homa	Sadeghi	21	MSc	Anthropology	17.7	1020
Milad	Vazan	32	PhD	Computer Science	18.96	1021

نام مستعار ستون

نام مستعار ستونی به شما امکان می‌دهد تا به یک ستون در لیست انتخاب دستور SELECT یک نام موقت اختصاص دهید. ستون مستعار به طور موقت در طول اجرای پرس و جو وجود دارد. شکل زیر نحو استفاده از نام مستعار ستونی را نشان می‌دهد:

```
SELECT column_name AS alias_name
FROM table_name;
```

کلمه کلیدی AS اختیاری است بنابراین می‌توانید آن را به صورت زیر حذف کنید:

```
SELECT column_name alias_name
FROM table_name;
```

اگر نام مستعار ستون حاوی یک یا چند فاصله باشد، باید اطراف آن را با نقل قول‌های دوتایی قرار دهید:

```
column_name AS "column alias"
```

In [137...

```
%%sql
SELECT first_name AS "First Name", last_name AS "Last Name", gpa AS "Grade Point Average"
FROM student_english;
```

Running query in 'default'

21 rows affected.

Out[137...

First Name	Last Name	Grade Point Average
Ali	Rezaei	16.5
Sara	Ahmadi	18.2
Mehdi	Hosseini	14.8
Neda	Karimi	17.5
Amir	Shirazi	19.5
Leila	Jafari	15.9
Hossein	Ghaffari	18.8
Maryam	Ebrahimi	18.5
Reza	Mohammadi	15.2
Fatemeh	Ramezani	17.8
Sina	Shahbazi	16.1
Zahra	Khalili	18.2
Pouya	Farhadi	14.5
Roya	Azizi	17.9
Kian	Bahrami	19.8
Shirin	Nassiri	15.6
Babak	Shokouhi	18.9
Afsaneh	Rahimi	18.6
Navid	Khosravi	15.3
Homa	Sadeghi	17.7
Milad	Vazan	18.96

In [138...

```sql
%%sql
--  معادل کد بالا بدون استفاده از AS
SELECT first_name "First Name", last_name "Last Name", gpa "Grade Point Average"
FROM student_english;
```

Running query in 'default'

21 rows affected.

Out[138...

First Name	Last Name	Grade Point Average
Ali	Rezaei	16.5
Sara	Ahmadi	18.2
Mehdi	Hosseini	14.8
Neda	Karimi	17.5
Amir	Shirazi	19.5
Leila	Jafari	15.9
Hossein	Ghaffari	18.8
Maryam	Ebrahimi	18.5
Reza	Mohammadi	15.2
Fatemeh	Ramezani	17.8
Sina	Shahbazi	16.1
Zahra	Khalili	18.2
Pouya	Farhadi	14.5
Roya	Azizi	17.9
Kian	Bahrami	19.8
Shirin	Nassiri	15.6
Babak	Shokouhi	18.9
Afsaneh	Rahimi	18.6
Navid	Khosravi	15.3
Homa	Sadeghi	17.7
Milad	Vazan	18.96

In [139…
```sql
%%sql
-- خطا دارد به دلیل اینکه نام ستون حاوی فضای خالی است ولی نام در داخل "" قرار نگرفته است
SELECT first_name First Name, last_name Last Name, gpa Grade Point Average
FROM student_english;
```

Running query in 'default'

```
RuntimeError: If using snippets, you may pass the --with argument explicitly.
For more details please refer: https://jupysql.ploomber.io/en/latest/compose.html#with-argument


Original error message from DB driver:
(psycopg2.errors.SyntaxError) syntax error at or near "Name"
LINE 1: SELECT first_name First Name, last_name Last Name, gpa Grade...
                                ^

[SQL: SELECT first_name First Name, last_name Last Name, gpa Grade Point Average
FROM student_english;]
(Background on this error at: https://sqlalche.me/e/20/f405)
```

استفاده از دستور SELECT برای تبدیل داده‌ها

برای استفاده از دستور SELECT در PostgreSQL جهت بازیابی و تبدیل داده‌ها، می‌توانید علاوه بر استخراج اطلاعات از جدول، عملیات محاسباتی یا تبدیل داده‌ها را نیز در همان دستور انجام دهید.

In [140…
```sql
%%sql
DROP TABLE IF EXISTS inventories;
CREATE TABLE inventories (
  name VARCHAR(255),
  brand VARCHAR(50),
  quantity INT,
  price DECIMAL(19, 2)
);


INSERT INTO inventories (name, brand, quantity, price)
VALUES ('iPhone 14 Pro', 'Apple', 10, 999.99),
       ('Galaxy S23 Ultra', 'Samsung', 15, 1199.99),
       ('Pixel 7 Pro', 'Google', 8, 899.99),
       ('Xperia 1 IV', 'Sony', 7, 1299.99) RETURNING *;
```

Running query in 'default'

4 rows affected.

Out[140…

name	brand	quantity	price
iPhone 14 Pro	Apple	10	999.99
Galaxy S23 Ultra	Samsung	15	1199.99
Pixel 7 Pro	Google	8	899.99
Xperia 1 IV	Sony	7	1299.99

به عنوان مثال، اگر بخواهید مقدار موجودی کالاها را با ضرب تعداد در قیمت هر محصول در جدول inventories محاسبه کنید، می‌توانید از دستور زیر استفاده کنید:

In [141…
```sql
%%sql
SELECT
  name,
  quantity * price
FROM
  inventories;
```

Running query in 'default'

4 rows affected.

Out[141…

name	?column?
iPhone 14 Pro	9999.90
Galaxy S23 Ultra	17999.85
Pixel 7 Pro	7199.92
Xperia 1 IV	9099.93

در این دستور، ما داده‌ها را از ستون name جدول inventories بازیابی می‌کنیم. علاوه بر این، داده‌های ستون‌های quantity و price را نیز گرفته و همزمان آنها را در هم ضرب می‌کنیم.

علاوه بر عملگر ضرب (*)، می‌توانید از سایر عملگرهای ریاضی مانند جمع (+)، تفریق (-) و تقسیم (/) نیز استفاده کنید.

توجه داشته باشید که خروجی ستون محاسبه‌شده (مثل quantity * price) به صورت ?column? نمایش داده می‌شود. این نام یک نام موقت است که PostgreSQL به آن اختصاص می‌دهد و معمولاً نام معناداری ندارد.

برای اختصاص یک نام ستون معنی دار به یک ستون محاسبه شده، می‌توانید از نام مستعار ستون استفاده کنید.

In [142…
```sql
%%sql
SELECT
  name,
  quantity * price AS amount
FROM
  inventories;
```

Running query in 'default'

4 rows affected.

Out[142…

name	amount
iPhone 14 Pro	9999.90
Galaxy S23 Ultra	17999.85
Pixel 7 Pro	7199.92
Xperia 1 IV	9099.93

عبارت WHERE و نام مستعار ستون

از آنجایی که PostgreSQL عبارت WHERE را قبل از عبارت SELECT ارزیابی می‌کند، نام مستعار ستون در زمان ارزیابی عبارت WHERE در دسترس نیست.

دستور زیر سعی می‌کند از ستون مقدار مستعار در WHERE استفاده کند و منجر به خطا می‌شود:

In [143…
```sql
%%sql
SELECT
  name,
  quantity * price AS amount
FROM
  inventories
WHERE
  amount > 10000;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.UndefinedColumn) column "amount" does not exist
LINE 7:   amount > 10000;
          ^

[SQL: SELECT
  name,
  quantity * price AS amount
FROM
  inventories
WHERE
  amount > 10000;]
(Background on this error at: https://sqlalche.me/e/20/f405)
```

خروجی نشان می‌دهد که ستون مقدار وجود ندارد. برای رفع این خطا، باید از عبارت WHERE به صورت زیر استفاده کنید:

In [144…
```sql
%%sql
SELECT
  name,
  quantity * price AS amount
FROM
  inventories
WHERE
  quantity * price > 10000;
```

Running query in 'default'

1 rows affected.

Out[144…

name	amount
Galaxy S23 Ultra	17999.85

استفاده از دستور PostgreSQL SELECT بدون بند FROM برای انجام محاسبات یا بازیابی مقادیر مستقیم

می توانید از دستور SELECT برای محاسبه مجموع اعداد بدون ارجاع به جدول استفاده کنید:

In [145…
```sql
%%sql
SELECT 10 + 15 AS result;
```

Running query in 'default'

1 rows affected.

Out[145…

result
25

In [146…
```sql
%%sql
SELECT 10 + 5 AS addition,
       10 - 5 AS subtraction,
       10 * 5 AS multiplication,
       10 / 5 AS division;
```

Running query in 'default'

1 rows affected.

Out[146…

addition	subtraction	multiplication	division
15	5	50	2

از تابع NOW() برای دریافت تاریخ و زمان فعلی استفاده کنید:

In [147…
```sql
%%sql
SELECT NOW()
```

Running query in 'default'

1 rows affected.

Out[147…

now
2025-04-22 19:14:26.514096+03:30

اطلاعات مربوط به پایگاه داده فعلی را دریافت کنید:

In [148…
```sql
%%sql
SELECT current_database();
```

Running query in 'default'

1 rows affected.

Out[148…

current_database
jupysql_sbu

In [149…
```sql
%%sql
SELECT current_user;
```

Running query in 'default'

1 rows affected.

Out[149…

current_user
postgres

عبارت ORDER BY

وقتی داده‌ها را از یک جدول پرس‌وجو می‌کنید (کوئری می‌زنید)، دستور SELECT ردیف‌ها را به ترتیب نامشخصی برمی‌گرداند. برای مرتب‌سازی ردیف‌های مجموعه نتایج، از عبارت `ORDER BY` در عبارت SELECT استفاده می‌کنید.

نحو ORDER BY به صورت زیر است:

```sql
SELECT
  column1,
  column2,
  ...
FROM
  table_name
ORDER BY
  sort_expression1 [ASC | DESC],
  sort_expression2 [ASC | DESC],
  ...;
```

از `ASC` برای مرتب‌سازی ردیف‌ها به ترتیب صعودی و `DESC` برای مرتب‌سازی ردیف‌ها به ترتیب نزولی استفاده کنید. گزینه `ASC` یا `DESC` اختیاری است. اگر به صراحت آن را مشخص نکرده باشید، عبارت `ORDER BY` به‌طور پیش فرض از `ASC` استفاده می‌کند.

در PostgreSQL، ترتیب ارزیابی بخش‌های مختلف یک دستور SELECT به شکل زیر است:

- FROM
- SELECT
- ORDER BY

این ترتیب به این معناست که ابتدا داده‌ها از جدول (یا جداول) انتخاب می‌شوند (FROM)، سپس ستون‌ها و عبارات مورد نظر در `SELECT` محاسبه می‌شوند، و در نهایت نتیجه بر اساس دستور `ORDER BY` مرتب می‌شود. از آنجا که بخش `SELECT` قبل از `ORDER BY` ارزیابی می‌شود، شما می‌توانید از نام مستعار که در SELECT تعریف کرده‌اید، در `ORDER BY` استفاده کنید.

سوال؟

لیست تمام دانشجویان را به ترتیب صعودی معدل (gpa) نمایش بده.

In [150…
```sql
%%sql
SELECT * FROM student
ORDER BY gpa ASC;
```

Running query in 'default'

21 rows affected.

Out[150…

first_name	last_name	age	grade	major	gpa	student_id
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
حسین	غفاری	24	دکتری	شیمی	18.8	1007
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015

سوال؟

دانشجویان را بر اساس مقطع تحصیلی (grade) و سپس بر اساس سن (age) به صورت نزولی مرتب کن.

In [151…
```sql
%%sql
SELECT * FROM student
ORDER BY grade DESC, age DESC;
```

Running query in 'default'

21 rows affected.

Out[151...

first_name	last_name	age	grade	major	gpa	student_id
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
حسین	غفاری	24	دکتری	شیمی	18.8	1007
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015

سوال؟

تمام دانشجویان را بر اساس نام خانوادگی (last_name) به ترتیب الفبایی نمایش بده.

In [152...

```sql
%%sql
SELECT * FROM student
ORDER BY last_name ASC;
```

Running query in 'default'

21 rows affected.

Out[152...

first_name	last_name	age	grade	major	gpa	student_id
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
حسین	غفاری	24	دکتری	شیمی	18.8	1007
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021

سوال؟

دانشجویان مقطع دکتری را نمایش بده و آن‌ها را بر اساس معدل از بیشترین به کمترین مرتب کن.

In [153...

```
%%sql
SELECT * FROM student
WHERE grade = 'دکتری'
ORDER BY gpa DESC;
```

Running query in 'default'

5 rows affected.

Out[153...

first_name	last_name	age	grade	major	gpa	student_id
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
حسین	غفاری	24	دکتری	شیمی	18.8	1007

In [48]:

```
%%sql
INSERT INTO student (first_name, last_name, age, grade, major, gpa, student_id)
VALUES
('1022 ,16.2 ,'ریاضی' ,'کارشناسی' ,20 ,'کاظمی' ,'نیما'),
('1023 ,18.1 ,'ریاضی' ,'کارشناسی ارشد' ,22 ,'کریمی' ,'مریم'),
('1024 ,18.7 ,'ریاضی' ,'دکتری' ,23 ,'رستمی' ,'یاسر'),
('1025 ,17.4 ,'ریاضی' ,'کارشناسی' ,21 ,'سجادی' ,'الهام'),
('1026 ,15.8 ,'ریاضی' ,'کارشناسی' ,19 ,'بابایی' ,'سینا'),
('1027 ,19.1 ,'ریاضی' ,'دکتری' ,24 ,'دهقان' ,'حامد'),
('1028 ,16.9 ,'ریاضی' ,'کارشناسی' ,20 ,'حبیبی' ,'نسترن'),
('1029 ,18.4 ,'ریاضی' ,'کارشناسی ارشد' ,22 ,'نوری' ,'فرزاد'),
('1030 ,17.6 ,'ریاضی' ,'کارشناسی ارشد' ,21 ,'قنبری' ,'آرزو'),
('1031 ,16.5 ,'ریاضی' ,'کارشناسی' ,20 ,'موسوی' ,'مجتبی'),
('1032 ,17.8 ,'آمار' ,'کارشناسی ارشد' ,21 ,'مرادی' ,'زهرا'),
('1033 ,18.0 ,'آمار' ,'کارشناسی ارشد' ,22 ,'توحیدی' ,'احمد'),
('1034 ,16.4 ,'آمار' ,'کارشناسی' ,20 ,'نعمتی' ,'فاطمه'),
('1035 ,18.9 ,'آمار' ,'دکتری' ,23 ,'پورحسین' ,'رضا'),
('1036 ,15.7 ,'آمار' ,'کارشناسی' ,19 ,'مهدوی' ,'ترانه'),
('1037 ,19.3 ,'آمار' ,'دکتری' ,24 ,'معصومی' ,'حسین'),
('1038 ,16.8 ,'آمار' ,'کارشناسی' ,21 ,'عبدی' ,'نازنین'),
('1039 ,17.5 ,'آمار' ,'کارشناسی ارشد' ,22 ,'کاشانی' ,'جواد'),
```

```
('1040' ,16.0 ,'آمار' ,'کارشناسی' ,20 ,'سهرابی' ,'پریسا'),
('1041' ,18.6 ,'آمار' ,'دکتری' ,23 ,'سعیدی' ,'مهدی') RETURNING *;
```

Running query in 'default'

20 rows affected.

Out[48]:

first_name	last_name	age	grade	major	gpa	student_id
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
مریم	کریمی	22	کارشناسی ارشد	ریاضی	18.1	1023
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
الهام	سجادی	21	کارشناسی	ریاضی	17.4	1025
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
حامد	دهقان	24	دکتری	ریاضی	19.1	1027
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
فرزاد	نوری	22	کارشناسی ارشد	ریاضی	18.4	1029
آرزو	قنبری	21	کارشناسی ارشد	ریاضی	17.6	1030
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
زهرا	مرادی	21	کارشناسی ارشد	آمار	17.8	1032
احمد	توحیدی	22	کارشناسی ارشد	آمار	18.0	1033
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
رضا	پورحسین	23	دکتری	آمار	18.9	1035
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
حسین	معصومی	24	دکتری	آمار	19.3	1037
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
جواد	کاشانی	22	کارشناسی ارشد	آمار	17.5	1039
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040
مهدی	سعیدی	23	دکتری	آمار	18.6	1041

سوال؟

لیست دانشجویانی که در رشته‌ی 'ریاضی' یا 'آمار' هستند را بر اساس سن به صورت افزایشی و سپس بر اساس معدل به صورت کاهشی مرتب کن.

In [155…

```
%%sql
-- فقط در داخل هر گروه سنی مساوی، معدل به صورت نزولی مرتب می‌شه
SELECT * FROM student
WHERE major IN ('آمار' ,'ریاضی')
ORDER BY age ASC, gpa DESC;
```

Running query in 'default'

21 rows affected.

Out[155...

first_name	last_name	age	grade	major	gpa	student_id
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
زهرا	مرادی	21	کارشناسی ارشد	آمار	17.8	1032
آرزو	قنبری	21	کارشناسی ارشد	ریاضی	17.6	1030
الهام	سجادی	21	کارشناسی	ریاضی	17.4	1025
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
فرزاد	نوری	22	کارشناسی ارشد	ریاضی	18.4	1029
مریم	کریمی	22	کارشناسی ارشد	ریاضی	18.1	1023
احمد	توحیدی	22	کارشناسی ارشد	آمار	18.0	1033
جواد	کاشانی	22	کارشناسی ارشد	آمار	17.5	1039
رضا	پورحسین	23	دکتری	آمار	18.9	1035
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
مهدی	سعیدی	23	دکتری	آمار	18.6	1041
حسین	معصومی	24	دکتری	آمار	19.3	1037
حامد	دهقان	24	دکتری	ریاضی	19.1	1027

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های آموزشی آمار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

دوشنبه ـ ۱ اردیبهشت ۱۴۰۴

Monday - 2025 21 April

PostgreSQL ORDER BY: برخورد با NULL ها در مرتب‌سازی

در PostgreSQL، مقدار NULL به معنی داده ناشناخته یا مفقود است. از آنجا که NULL ناشناخته است، نمی‌توان آن را با سایر مقادیر مقایسه کرد. با این حال، PostgreSQL برای انجام مرتب‌سازی باید بداند کدام مقادیر قبل یا بعد از سایر مقادیر قرار می‌گیرند. در مورد NULL، PostgreSQL دو گزینه در بخش ORDER BY ارائه می‌دهد:

```
ORDER BY sort_expression NULLS FIRST
ORDER BY sort_expression NULLS LAST
```

در این دستور:

- `NULLS FIRST` باعث می‌شود مقادیر NULL قبل از سایر مقادیر غیر NULL قرار بگیرند.
- `NULLS LAST` باعث می‌شود مقادیر NULL بعد از سایر مقادیر غیر NULL قرار بگیرند.

همچنین توجه داشته باشید که بین عبارت مرتب‌سازی و NULLS FIRST یا NULLS LAST می‌توانید از گزینه‌های ASC یا DESC استفاده کنید.

In [158...

```
%%sql
DROP TABLE IF EXISTS student_nulls;
CREATE TABLE student_nulls (
    first_name TEXT,
    last_name TEXT,
    age INTEGER,
```

```
    grade TEXT,
    major TEXT,
    gpa REAL,
    student_id INTEGER
);

INSERT INTO student_nulls (first_name, last_name, age, grade, major, gpa, student_id)
VALUES
('2001 ,16.5 ,'مهندسی' ,'کارشناسی' ,20 ,'رضایی' ,'علی'),
('2002 ,18.2 ,'پزشکی' ,'کارشناسی ارشد' ,22 ,'احمدی' ,'سارا'),
('حقوق' ,'کارشناسی' ,19 ,'حسینی' ,'مهدی', NULL, 2003),
('2004 ,17.5 ,'بازرگانی' ,'کارشناسی ارشد' ,21 ,'کریمی' ,'ندا'),
('2005 ,19.5 ,'فیزیک' ,'دکتری' ,23 ,'شیرازی' ,'امیر'),
('هنر' ,'کارشناسی' ,20 ,'جعفری' ,'لیلا', NULL, 2006),
('2007 ,18.8 ,'شیمی' ,'دکتری' ,24 ,'غفاری' ,'حسین'),
('2008 ,18.5 ,'زیست‌شناسی' ,'کارشناسی ارشد' ,22 ,'ابراهیمی' ,'مریم'),
('2009 ,15.2 ,'علوم کامپیوتر' ,'کارشناسی' ,19 ,'محمدی' ,'رضا'),
('ریاضی' ,'کارشناسی ارشد' ,21 ,'رمضانی' ,'فاطمه', NULL, 2010),
('2011 ,16.1 ,'ریاضی' ,'کارشناسی' ,20 ,'شاهبازی' ,'سینا'),
('2012 ,14.5 ,'آمار' ,'کارشناسی' ,19 ,'فرهادی' ,'پویا'),
('آمار' ,'کارشناسی ارشد' ,22 ,'خلیلی' ,'زهرا', NULL, 2013),
('2014 ,15.3 ,'علوم سیاسی' ,'کارشناسی' ,19 ,'خسروی' ,'نوید'),
('2015 ,18.96 ,'علوم کامپیوتر' ,'دکتری' ,32 ,'وزان' ,'میلاد') RETURNING *;
```

Running query in 'default'

15 rows affected.

Out[158…

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	2001
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	2002
مهدی	حسینی	19	کارشناسی	حقوق	None	2003
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	2004
امیر	شیرازی	23	دکتری	فیزیک	19.5	2005
لیلا	جعفری	20	کارشناسی	هنر	None	2006
حسین	غفاری	24	دکتری	شیمی	18.8	2007
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	2008
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	2009
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	None	2010
سینا	شاهبازی	20	کارشناسی	ریاضی	16.1	2011
پویا	فرهادی	19	کارشناسی	آمار	14.5	2012
زهرا	خلیلی	22	کارشناسی ارشد	آمار	None	2013
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	2014
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	2015

سوال؟

لیست دانشجویان را بر اساس معدل به صورت نزولی نمایش بده، طوری که دانشجویانی که معدل ندارند (NULL) در انتهای لیست قرار بگیرند.

In [159…

```
%%sql
SELECT * FROM student_nulls
ORDER BY gpa DESC NULLS LAST;
```

Running query in 'default'

15 rows affected.

Out[159...

first_name	last_name	age	grade	major	gpa	student_id
امیر	شیرازی	23	دکتری	فیزیک	19.5	2005
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	2015
حسین	غفاری	24	دکتری	شیمی	18.8	2007
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	2008
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	2002
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	2004
علی	رضایی	20	کارشناسی	مهندسی	16.5	2001
سینا	شاهبازی	20	کارشناسی	ریاضی	16.1	2011
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	2014
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	2009
پویا	فرهادی	19	کارشناسی	آمار	14.5	2012
لیلا	جعفری	20	کارشناسی	هنر	None	2006
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	None	2010
زهرا	خلیلی	22	کارشناسی ارشد	آمار	None	2013
مهدی	حسینی	19	کارشناسی	حقوق	None	2003

سوال؟

لیست دانشجویان را بر اساس معدل به صورت صعودی مرتب کن، با این تفاوت که دانشجویانی که معدل ندارند، در ابتدای لیست قرار بگیرند.

In [160...

```
%%sql
SELECT * FROM student_nulls
ORDER BY gpa ASC NULLS FIRST;
```

Running query in 'default'

15 rows affected.

Out[160...

first_name	last_name	age	grade	major	gpa	student_id
لیلا	جعفری	20	کارشناسی	هنر	None	2006
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	None	2010
مهدی	حسینی	19	کارشناسی	حقوق	None	2003
زهرا	خلیلی	22	کارشناسی ارشد	آمار	None	2013
پویا	فرهادی	19	کارشناسی	آمار	14.5	2012
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	2009
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	2014
سینا	شاهبازی	20	کارشناسی	ریاضی	16.1	2011
علی	رضایی	20	کارشناسی	مهندسی	16.5	2001
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	2004
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	2002
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	2008
حسین	غفاری	24	دکتری	شیمی	18.8	2007
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	2015
امیر	شیرازی	23	دکتری	فیزیک	19.5	2005

سوال؟

لیست دانشجویان رشته‌ی "ریاضی" را بر اساس معدل به صورت نزولی مرتب کن، و اگر معدلی وجود نداشت، آن‌ها را در ابتدا نمایش بده.

In [161...

```
%%sql
SELECT * FROM student_nulls
WHERE major = 'ریاضی'
ORDER BY gpa DESC NULLS FIRST;
```

Running query in 'default'

2 rows affected.

Out[161…

first_name	last_name	age	grade	major	gpa	student_id
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	None	2010
سینا	شاهبازی	20	کارشناسی	ریاضی	16.1	2011

سوال؟

لیست دانشجویان مقطع کارشناسی را بر اساس معدل به صورت صعودی نمایش بده، طوری که ابتدا کسانی که معدل ندارند بیایند، و بعد بقیه.

In [162…

```
%%sql
SELECT * FROM student_nulls
WHERE grade = 'کارشناسی'
ORDER BY gpa ASC NULLS FIRST;
```

Running query in 'default'

7 rows affected.

Out[162…

first_name	last_name	age	grade	major	gpa	student_id
مهدی	حسینی	19	کارشناسی	حقوق	None	2003
لیلا	جعفری	20	کارشناسی	هنر	None	2006
پویا	فرهادی	19	کارشناسی	آمار	14.5	2012
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	2009
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	2014
سینا	شاهبازی	20	کارشناسی	ریاضی	16.1	2011
علی	رضایی	20	کارشناسی	مهندسی	16.5	2001

عبارت LIMIT

LIMIT یک عبارت اختیاری از عبارت SELECT است که تعداد سطرهای برگردانده شده توسط پرسمان را محدود می‌کند. در زیر نحو اصلی عبارت LIMIT آمده است:

```
SELECT
  select_list
FROM
  table_name
ORDER BY
  sort_expression
LIMIT
  row_count;
```

بسیار مهم است که همیشه باید از دستور LIMIT با عبارت ORDER BY برای به دست آوردن **مجموعه نتایج مورد انتظار** استفاده کنید. اگر از دستور LIMIT با عبارت ORDER BY استفاده نکنید، دستور SELECT ردیف‌ها را به ترتیب نامشخصی برمی‌گرداند، و دستور LIMIT ردیف‌های بالایی را از ردیف‌های نامرتب انتخاب می‌کند، که احتمالاً منجر به نتایج غیرمنتظره می‌شود.

سوال؟

لیست ۵ دانشجوی اول جدول را نمایش بده.

In [163…

```
%%sql
SELECT * FROM student
LIMIT 5;
```

Running query in 'default'

5 rows affected.

Out[163…

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005

سوال؟

لیست ۳ دانشجویی که بالاترین معدل را دارند نمایش بده.

In [164…
```
%%sql
SELECT * FROM student
ORDER BY gpa DESC
LIMIT 3;
```

Running query in 'default'

3 rows affected.

Out[164…

first_name	last_name	age	grade	major	gpa	student_id
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
حسین	معصومی	24	دکتری	آمار	19.3	1037

سوال؟

۵ دانشجویی که کمترین سن را دارند نمایش بده.

In [165…
```
%%sql
SELECT * FROM student
ORDER BY age ASC
LIMIT 5;
```

Running query in 'default'

5 rows affected.

Out[165…

first_name	last_name	age	grade	major	gpa	student_id
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026

سوال؟

۳ دانشجوی اولی که در رشته «ریاضی» هستند و بالاترین معدل را دارند، نمایش بده.

In [166…
```
%%sql
SELECT * FROM student
WHERE major = 'ریاضی'
ORDER BY gpa DESC
LIMIT 3;
```

Running query in 'default'

3 rows affected.

Out[166…

first_name	last_name	age	grade	major	gpa	student_id
حامد	دهقان	24	دکتری	ریاضی	19.1	1027
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
فرزاد	نوری	22	کارشناسی ارشد	ریاضی	18.4	1029

اگر می‌خواهید قبل از بازگرداندن ردیف‌های row_count از تعدادی ردیف بگذرید، می‌توانید از عبارت OFFSET که بعد از عبارت LIMIT قرار گرفته است استفاده کنید:

```
SELECT
  column1,
  column2
FROM
  table_name
ORDER BY
  sort_expression
LIMIT
  row_count
OFFSET
  skip_count;
```

سوال؟

۵ دانشجویی که بعد از ۵ دانشجویی که بالاترین معدل را دارند قرار دارند، نمایش بده.

یعنی دانشجویانی با رتبه‌های ۶ تا ۱۰ از نظر معدل.

In [167…
```
%%sql
SELECT * FROM student
ORDER BY gpa DESC
LIMIT 5
OFFSET 5;
```

Running query in 'default'

5 rows affected.

Out[167…

first_name	last_name	age	grade	major	gpa	student_id
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
رضا	پورحسین	23	دکتری	آمار	18.9	1035
حسین	غفاری	24	دکتری	شیمی	18.8	1007
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018

سوال؟

۳ دانشجویی که در رشته «آمار» هستند و از رده سوم به بعد قرار دارند (بر اساس معدل نزولی)، نمایش بده.

یعنی نفرات ۳، ۴ و ۵ رشته آمار از نظر معدل.

In [168…
```
%%sql
SELECT * FROM student
WHERE major = 'آمار'
ORDER BY gpa DESC
LIMIT 3
OFFSET 2;
```

Running query in 'default'

3 rows affected.

Out[168…

first_name	last_name	age	grade	major	gpa	student_id
مهدی	سعیدی	23	دکتری	آمار	18.6	1041
احمد	توحیدی	22	کارشناسی ارشد	آمار	18.0	1033
زهرا	مرادی	21	کارشناسی ارشد	آمار	17.8	1032

سوال؟

۵ دانشجوی بعد از اولین دانشجوی رشته‌ی «آمار» که معدل آن‌ها کمتر از ۱۸ است را نمایش بده.

In [169…
```
%%sql
SELECT * FROM student
WHERE major = 'آمار' AND gpa < 18
ORDER BY gpa DESC
LIMIT 5
OFFSET 1;
```

Running query in 'default'

5 rows affected.

Out[169…

first_name	last_name	age	grade	major	gpa	student_id
جواد	کاشانی	22	کارشناسی ارشد	آمار	17.5	1039
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036

سوال؟

از بین دانشجویان «دکتری» که معدل‌شان کمتر از ۱۹ است، ۲ دانشجو را پس از نفر اول نمایش بده.

In [170…

```
%%sql
SELECT * FROM student
WHERE grade = 'دکتری' AND gpa < 19
ORDER BY gpa DESC
LIMIT 2
OFFSET 1;
```

Running query in 'default'

2 rows affected.

Out[170…

first_name	last_name	age	grade	major	gpa	student_id
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
رضا	پورحسین	23	دکتری	آمار	18.9	1035

عملگر BETWEEN

عملگر BETWEEN به شما اجازه می‌دهد بررسی کنید که آیا یک مقدار در بازه‌ای از مقادیر قرار دارد یا خیر.

ساختار پایه‌ای عملگر BETWEEN به صورت زیر است:

```
value BETWEEN low AND high;
```

اگر مقدار مورد نظر بزرگ‌تر یا مساوی مقدار پایین (low) و کوچکتر یا مساوی مقدار بالا (high) باشد، عملگر BETWEEN مقدار true برمی‌گرداند؛ در غیر این صورت، مقدار false برمی‌گرداند.

می‌توانید عملگر BETWEEN را با استفاده از عملگرهای بزرگ‌تر یا مساوی ، کوچکتر یا مساوی و عملگر منطقی AND بازنویسی کنید:

```
value >= low AND value <= high
```

اگر بخواهید بررسی کنید که مقدار خارج از بازه مشخصی است، می‌توانید از عملگر NOT BETWEEN به صورت زیر استفاده کنید:

```
value NOT BETWEEN low AND high
```

عبارت زیر معادل عبارت استفاده شده با عملگر NOT BETWEEN است:

```
value < low OR value > high
```

سوال؟

انتخاب دانشجویانی که سن آن‌ها بین ۲۰ تا ۲۲ سال است:

In [171…

```
%%sql
SELECT *
FROM student
WHERE age BETWEEN 20 AND 22;
```

Running query in 'default'

25 rows affected.

Out[171...

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
مریم	کریمی	22	کارشناسی ارشد	ریاضی	18.1	1023
الهام	سجادی	21	کارشناسی	ریاضی	17.4	1025
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
فرزاد	نوری	22	کارشناسی ارشد	ریاضی	18.4	1029
آرزو	قنبری	21	کارشناسی ارشد	ریاضی	17.6	1030
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
زهرا	مرادی	21	کارشناسی ارشد	آمار	17.8	1032
احمد	توحیدی	22	کارشناسی ارشد	آمار	18.0	1033
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
جواد	کاشانی	22	کارشناسی ارشد	آمار	17.5	1039
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040

سوال؟

لیست دانشجویان کارشناسی که معدلشان بین 15 تا 17 است:

In [172...

```sql
%%sql
SELECT *
FROM student
WHERE grade = 'کارشناسی' AND gpa BETWEEN 15 AND 17
order by gpa desc;
```

```
Running query in 'default'
14 rows affected.
```

Out[172...

first_name	last_name	age	grade	major	gpa	student_id
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009

سوال؟

نمایش دانشجویانی که سن آن‌ها بین 19 تا 23 و معدلشان بین 18 تا 20 است:

In [173...

```
%%sql
SELECT *
FROM student
WHERE age BETWEEN 19 AND 23 AND gpa BETWEEN 18 AND 20;
```

Running query in 'default'

12 rows affected.

Out[173...

first_name	last_name	age	grade	major	gpa	student_id
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
مریم	کریمی	22	کارشناسی ارشد	ریاضی	18.1	1023
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
فرزاد	نوری	22	کارشناسی ارشد	ریاضی	18.4	1029
احمد	توحیدی	22	کارشناسی ارشد	آمار	18.0	1033
رضا	پورحسین	23	دکتری	آمار	18.9	1035
مهدی	سعیدی	23	دکتری	آمار	18.6	1041

سوال؟

نمایش دانشجویانی که سن آن‌ها خارج از بازه ۲۰ تا ۲۲ سال است:

In [174...

```
%%sql
SELECT *
FROM student
WHERE age NOT BETWEEN 20 AND 22;
```

Running query in 'default'

16 rows affected.

Out[174...

first_name	last_name	age	grade	major	gpa	student_id
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
حسین	غفاری	24	دکتری	شیمی	18.8	1007
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
حامد	دهقان	24	دکتری	ریاضی	19.1	1027
رضا	پورحسین	23	دکتری	آمار	18.9	1035
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
حسین	معصومی	24	دکتری	آمار	19.3	1037
مهدی	سعیدی	23	دکتری	آمار	18.6	1041

سوال؟

نمایش دانشجویانی که یا سنشان خارج از بازه 19 تا 23 است یا معدلشان خارج از بازه 18 تا 20:

In [175...

```
%%sql
SELECT *
FROM student
WHERE age NOT BETWEEN 19 AND 23 OR gpa NOT BETWEEN 18 AND 20;
```

Running query in 'default'

29 rows affected.

Out[175...

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
حسین	غفاری	24	دکتری	شیمی	18.8	1007
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
الهام	سجادی	21	کارشناسی	ریاضی	17.4	1025
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
حامد	دهقان	24	دکتری	ریاضی	19.1	1027
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
آرزو	قنبری	21	کارشناسی ارشد	ریاضی	17.6	1030
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
زهرا	مرادی	21	کارشناسی ارشد	آمار	17.8	1032
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
حسین	معصومی	24	دکتری	آمار	19.3	1037
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
جواد	کاشانی	22	کارشناسی ارشد	آمار	17.5	1039
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040

In [176...

```
%%sql
SELECT *
FROM student
order by grade desc;
```

```
Running query in 'default'
41 rows affected.
```

Out[176...

first_name	last_name	age	grade	major	gpa	student_id
زهرا	مرادی	21	کارشناسی ارشد	آمار	17.8	1032
احمد	توحیدی	22	کارشناسی ارشد	آمار	18.0	1033
آرزو	قنبری	21	کارشناسی ارشد	ریاضی	17.6	1030
افسانه	رحیمی	22	کارشناسی ارشد	فلسفه	18.6	1018
مریم	ابراهیمی	22	کارشناسی ارشد	زیست‌شناسی	18.5	1008
هما	صادقی	21	کارشناسی ارشد	انسان‌شناسی	17.7	1020
فرزاد	نوری	22	کارشناسی ارشد	ریاضی	18.4	1029
ندا	کریمی	21	کارشناسی ارشد	بازرگانی	17.5	1004
مریم	کریمی	22	کارشناسی ارشد	ریاضی	18.1	1023
فاطمه	رمضانی	21	کارشناسی ارشد	ریاضی	17.8	1010
جواد	کاشانی	22	کارشناسی ارشد	آمار	17.5	1039
زهرا	خلیلی	22	کارشناسی ارشد	ادبیات	18.2	1012
سارا	احمدی	22	کارشناسی ارشد	پزشکی	18.2	1002
رویا	عزیزی	21	کارشناسی ارشد	روان‌شناسی	17.9	1014
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
الهام	سجادی	21	کارشناسی	ریاضی	17.4	1025
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040
کیان	بهرامی	23	دکتری	اخترشناسی	19.8	1015
حسین	غفاری	24	دکتری	شیمی	18.8	1007
یاسر	رستمی	23	دکتری	ریاضی	18.7	1024
مهدی	سعیدی	23	دکتری	آمار	18.6	1041
رضا	پورحسین	23	دکتری	آمار	18.9	1035
حامد	دهقان	24	دکتری	ریاضی	19.1	1027
امیر	شیرازی	23	دکتری	فیزیک	19.5	1005
میلاد	وزان	32	دکتری	علوم کامپیوتر	18.96	1021
بابک	شکوهی	24	دکتری	زمین‌شناسی	18.9	1017
حسین	معصومی	24	دکتری	آمار	19.3	1037

توابع تجمعی (Aggregate functions)

یک تابع تجمعی گروهی از مقادیر را دریافت می‌کند، محاسبه‌ای انجام می‌دهد و یک مقدار واحد بازمی‌گرداند.

رایج‌ترین توابع تجمعی:

- **تابع MIN**: کمترین مقدار در یک مجموعه را بازمی‌گرداند.
- **تابع MAX**: بیشترین مقدار در یک مجموعه را بازمی‌گرداند.

- **تابع SUM**: مجموع مقادیر یک مجموعه را بازمی‌گرداند.
- **تابع AVG**: میانگین مقادیر یک مجموعه را بازمی‌گرداند.
- **تابع COUNT**: تعداد آیتم‌های موجود در یک مجموعه را بازمی‌گرداند.

MIN

پیدا کردن کمترین معدل بین همه دانشجوها:

In [177…
```sql
%%sql
SELECT MIN(gpa) "min stu" FROM student;
```

Running query in 'default'
1 rows affected.

Out[177…

min stu
14.5

MAX

بیشترین سن در جدول دانشجویان:

In [178…
```sql
%%sql
SELECT MAX(age) AS max_age FROM student;
```

Running query in 'default'
1 rows affected.

Out[178…

max_age
32

SUM

مجموع معدل تمام دانشجویان مقطع دکتری:

In [179…
```sql
%%sql
SELECT SUM(gpa) AS total_gpa FROM student;
```

Running query in 'default'
1 rows affected.

Out[179…

total_gpa
713.95996

SUM

مجموع معدل تمام دانشجویان مقطع دکتری:

In [180…
```sql
%%sql
SELECT SUM(gpa) AS total_gpa_phd
FROM student
WHERE grade = 'دکتری';
```

Running query in 'default'
1 rows affected.

Out[180…

total_gpa_phd
190.56001

AVG

معدل کل دانشجوها

In [181…
```sql
%%sql
SELECT AVG(gpa) AS avg_gpa FROM student;
```

Running query in 'default'
1 rows affected.

Out[181…

avg_gpa
17.413658490995083

COUNT

تعداد کل دانشجوها

In [182…

```
%%sql
SELECT COUNT(*) AS total_students FROM student;
```

Running query in 'default'

1 rows affected.

Out[182…

total_students
41

عبارت `GROUP BY`

دستور `GROUP BY` در SQL برای گروه‌بندی ردیف‌های بازگردانده‌شده توسط دستور `SELECT` بر اساس مقدار یک یا چند ستون استفاده می‌شود. این دستور معمولاً همراه با توابع تجمعی مانند `MIN`، `AVG`، `COUNT`، `SUM` و `MAX` به کار می‌رود تا محاسباتی مانند جمع، تعداد، میانگین و غیره را روی هر گروه انجام دهد.

کاربردهای اصلی GROUP BY

- **تجمیع داده‌ها:** محاسبه آماری مانند جمع کل فروش، تعداد سفارش‌ها، میانگین امتیازها و غیره برای هر گروه.
- **گروه‌بندی داده‌های مشابه:** دسته‌بندی ردیف‌ها بر اساس یک یا چند ستون—به عنوان مثال، گروه‌بندی فروش‌ها بر اساس دسته‌بندی محصول یا کارمندان بر اساس دپارتمان.
- **خلاصه‌سازی داده‌های بزرگ:** کاهش تعداد ردیف‌ها با خلاصه‌سازی اطلاعات در گروه‌ها.

نحو پایه GROUP BY

```
SELECT
   column_1,
   column_2,
   ...,
   aggregate_function(column_3)
FROM
   table_name
GROUP BY
   column_1,
   column_2,
   ...;
```

ترتیب قرارگیری GROUP BY در کوئری

دستور `GROUP BY` همیشه بعد از `FROM` و `WHERE` و قبل از `HAVING` و `ORDER BY` قرار می‌گیرد:

```
SELECT
    column1, aggregate_function(column2)
FROM
    table_name
WHERE ...
GROUP BY
    column1
HAVING ...
ORDER BY
    column1;
```

⚠ نکته مهم:

هر ستونی که توی SELECT باشه و داخل تابع تجمیعی نباشه، حتماً باید توی GROUP BY بیاد. وگرنه PostgreSQL خطا می‌ده.

سوال؟

برای هر رشته‌ی تحصیلی (major)، معدل (gpa) دانشجوها رو محاسبه کن، و نتیجه رو بر اساس معدل از بیشترین به کمترین مرتب کن.

In [183…
```sql
%%sql
SELECT major, AVG(gpa) AS avg_gpa
FROM student
GROUP BY major
ORDER BY avg_gpa DESC;
```

Running query in 'default'

21 rows affected.

Out[183…

major	avg_gpa
اخترشناسی	19.799999237060547
فیزیک	19.5
زمین‌شناسی	18.899999618530273
شیمی	18.799999237060547
فلسفه	18.600000381469727
زیست‌شناسی	18.5
پزشکی	18.200000762939453
ادبیات	18.200000762939453
روان‌شناسی	17.899999618530273
انسان‌شناسی	17.700000762939453
ریاضی	17.500000086697664
بازرگانی	17.5
آمار	17.499999713897704
علوم کامپیوتر	17.079999446868896
مهندسی	16.5
تاریخ	16.100000381469727
هنر	15.899999618530273
جامعه‌شناسی	15.600000381469727
علوم سیاسی	15.300000190734863
حقوق	14.800000190734863
اقتصاد	14.5

چطوری کار میکنه؟

1. `GROUP BY major` : هر رشته تحصیلی مثل علوم کامپیوتر، پزشکی، فلسفه و... یه گروه می‌شه.
2. `AVG(gpa)` : معدل‌ها رو توی هر گروه میانگین می‌گیره.
3. `ORDER BY avg_gpa DESC` : خروجی رو طوری مرتب می‌کنه که رشته‌هایی با معدل بالاتر اول باشن.

سوال؟

برای هر مقطع تحصیلی (grade)، کمترین سن (age) بین دانشجوهاش رو نشان بده.

In [184…
```sql
%%sql
SELECT grade, MIN(age) AS min_age
FROM student
GROUP BY grade;
```

Running query in 'default'

3 rows affected.

Out[184…

grade	min_age
کارشناسی	19
دکتری	23
کارشناسی ارشد	21

چطوری کار میکنه؟

1. `GROUP BY grade` : تمام دانشجوهایی که مقطع تحصیلی یکسان دارن (مثل کارشناسی، کارشناسی ارشد، دکتری) رو توی یک گروه می‌ذاره.
2. `MIN(age)` : از هر گروه، کمترین مقدار سن رو پیدا می‌کنه.

سوال؟

برای هر مقطع تحصیلی (grade)، تعداد دانشجوها رو بشمار.

In [185…
```sql
%%sql
SELECT grade, COUNT(*)
FROM student
GROUP BY grade;
```

Running query in 'default'

3 rows affected.

Out[185…

grade	count
کارشناسی	17
دکتری	10
کارشناسی ارشد	14

چطوری کار میکنه؟

- `GROUP BY grade` : دانشجوها رو بر اساس مقطع تحصیلی دسته‌بندی می‌کنه (مثلاً کارشناسی، ارشد، دکتری).
- `COUNT(*)` : تعداد ردیف‌ها (دانشجوها) در هر گروه رو می‌شماره.

سوال؟

بیشترین جمعیت دانشجو مربوط به کدوم مقطعه؟

In [186…
```sql
%%sql
SELECT grade, COUNT(*) AS student_count
FROM student
GROUP BY grade
ORDER BY student_count DESC
LIMIT 1;
```

Running query in 'default'

1 rows affected.

Out[186…

grade	student_count
کارشناسی	17

چطوری کار میکنه؟

- اول **تعداد دانشجوها برای هر مقطع** رو می‌شماره.
- بعد با `ORDER BY student_count DESC` ، نتایج رو از **بیشترین به کمترین** مرتب می‌کنه.
- در نهایت با `LIMIT 1` فقط **پرجمعیت‌ترین مقطع** رو برمی‌گردونه.

سوال؟

معدل در هر مقطع تحصیلی

In [187…
```sql
%%sql
SELECT grade, AVG(gpa) AS avg_gpa
FROM student
GROUP BY grade
ORDER BY avg_gpa DESC;
```

Running query in 'default'

3 rows affected.

Out[187…

grade	avg_gpa
دکتری	19.055999755859375
کارشناسی ارشد	17.9857143674578
کارشناسی	15.976470554576201

سوال؟

پیدا کردن تعداد دانشجوها در هر ترکیب مقطع تحصیلی (grade) و رشته (major)، و مرتب‌سازی بر اساس تعداد دانشجو در هر ترکیب، از بیشترین به کمترین.

مثلا ببینی چند نفر در مقطع "کارشناسی" و رشته "آمار" هستن.

می‌خوای بدونی کدوم رشته در کدوم مقطع پرطرفدارتره یا بخوای آمار دقیق از توزیع دانشجوها داشته باشی

In [188…

```sql
%%sql
SELECT grade, major, COUNT(*) AS student_count
FROM student
GROUP BY grade, major
ORDER BY student_count DESC;
```

Running query in 'default'

26 rows affected.

Out[188…

grade	major	student_count
کارشناسی	ریاضی	5
کارشناسی ارشد	ریاضی	4
کارشناسی	آمار	4
کارشناسی ارشد	آمار	3
دکتری	آمار	3
دکتری	ریاضی	2
کارشناسی ارشد	پزشکی	1
کارشناسی ارشد	فلسفه	1
کارشناسی ارشد	بازرگانی	1
دکتری	زمین‌شناسی	1
دکتری	علوم کامپیوتر	1
کارشناسی	تاریخ	1
کارشناسی	علوم سیاسی	1
کارشناسی	هنر	1
کارشناسی	جامعه‌شناسی	1
کارشناسی	مهندسی	1
کارشناسی ارشد	ادبیات	1
دکتری	شیمی	1
کارشناسی	اقتصاد	1
کارشناسی	علوم کامپیوتر	1
کارشناسی	حقوق	1
دکتری	اخترشناسی	1
کارشناسی ارشد	انسان‌شناسی	1
کارشناسی ارشد	زیست‌شناسی	1
کارشناسی ارشد	روان‌شناسی	1
دکتری	فیزیک	1

In [189…

```sql
%%sql
DROP TABLE IF EXISTS inventories;
CREATE TABLE inventories (
    id INTEGER,
    product_name TEXT,
    brand TEXT,
    warehouse TEXT,
    quantity INTEGER
);

INSERT INTO inventories (id, product_name, brand, warehouse, quantity)
VALUES
(1, 'iPhone 16', 'Apple', 'Tehran', 50),
(2, 'iPhone 16', 'Apple', 'Isfahan', 30),
(3, 'iPhone 16', 'Apple', 'Mashhad', 25),
(4, 'iPhone 16', 'Apple', 'Tabriz', 35),
(5, 'iPhone 16 Pro', 'Apple', 'Tehran', 40),
(6, 'iPhone 16 Pro', 'Apple', 'Isfahan', 20),
(7, 'iPhone 16 Pro', 'Apple', 'Mashhad', 30),
(8, 'iPhone 16 Pro Max', 'Apple', 'Tabriz', 22),
(9, 'Galaxy S22', 'Samsung', 'Tehran', 50),
(10, 'Galaxy S22', 'Samsung', 'Isfahan', 30),
```

```
(11, 'Galaxy S22', 'Samsung', 'Mashhad', 25),
(12, 'Galaxy S22 Ultra', 'Samsung', 'Tabriz', 20),
(13, 'Galaxy Z Fold 4', 'Samsung', 'Tehran', 55),
(14, 'Galaxy Z Fold 4', 'Samsung', 'Isfahan', 35),
(15, 'Galaxy Z Fold 4', 'Samsung', 'Mashhad', 28),
(16, 'Pixel 8', 'Google', 'Tehran', 40),
(17, 'Pixel 8', 'Google', 'Tabriz', 18),
(18, 'Pixel 8 Pro', 'Google', 'Mashhad', 22),
(19, 'Pixel 8 Pro', 'Google', 'Isfahan', 26),
(20, 'Pixel Fold', 'Google', 'Tehran', 15) RETURNING *;
```

Running query in 'default'

20 rows affected.

Out[189…

id	product_name	brand	warehouse	quantity
1	iPhone 16	Apple	Tehran	50
2	iPhone 16	Apple	Isfahan	30
3	iPhone 16	Apple	Mashhad	25
4	iPhone 16	Apple	Tabriz	35
5	iPhone 16 Pro	Apple	Tehran	40
6	iPhone 16 Pro	Apple	Isfahan	20
7	iPhone 16 Pro	Apple	Mashhad	30
8	iPhone 16 Pro Max	Apple	Tabriz	22
9	Galaxy S22	Samsung	Tehran	50
10	Galaxy S22	Samsung	Isfahan	30
11	Galaxy S22	Samsung	Mashhad	25
12	Galaxy S22 Ultra	Samsung	Tabriz	20
13	Galaxy Z Fold 4	Samsung	Tehran	55
14	Galaxy Z Fold 4	Samsung	Isfahan	35
15	Galaxy Z Fold 4	Samsung	Mashhad	28
16	Pixel 8	Google	Tehran	40
17	Pixel 8	Google	Tabriz	18
18	Pixel 8 Pro	Google	Mashhad	22
19	Pixel 8 Pro	Google	Isfahan	26
20	Pixel Fold	Google	Tehran	15

سوال؟

مجموع موجودی (quantity) برای هر برند:

In [190…

```
%%sql
SELECT brand, SUM(quantity) AS total_quantity
FROM inventories
GROUP BY brand;
```

Running query in 'default'

3 rows affected.

Out[190…

brand	total_quantity
Google	121
Samsung	243
Apple	252

سوال؟

برای هر محصول (product_name)، کمترین مقدار موجودی (quantity) بین همه‌ی انبارها رو حساب کن:

In [191…

```
%%sql
SELECT product_name, MIN(quantity) AS min_quantity
FROM inventories
GROUP BY product_name;
```

Running query in 'default'

9 rows affected.

Out[191...

product_name	min_quantity
Pixel Fold	15
Galaxy S22	25
Galaxy Z Fold 4	28
Pixel 8 Pro	22
iPhone 16 Pro Max	22
Pixel 8	18
iPhone 16 Pro	20
Galaxy S22 Ultra	20
iPhone 16	25

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های اموزشی امار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

شنبه ـ ۶ اردیبهشت ۱۴۰۴

Saturday - 2025 26 April

دستور `HAVING`

عبارت HAVING یک ابزار قدرتمند در SQL است که به شما اجازه می‌دهد شرطی را برای فیلتر کردن گروه‌هایی که توسط دستور GROUP BY ایجاد شده‌اند، مشخص کنید. ساختار پایه استفاده از HAVING در زیر نشان داده شده است:

```
SELECT
  column1,
  aggregate_function (column2)
FROM
  table_name
GROUP BY
  column1
HAVING
  condition;
```

در این ساختار:

- ابتدا، دستور GROUP BY ردیف‌ها را بر اساس مقادیر ستون column1 به گروه‌هایی تقسیم می‌کند.
- سپس، عبارت HAVING گروه‌ها را بر اساس شرط تعیین شده فیلتر می‌کند.
- اگر یک گروه شرط مشخص‌شده را داشته باشد، در نتایج نهایی نمایش داده می‌شود.

ترتیب ارزیابی دستورات در PostgreSQL به این صورت است:

- FROM
- WHERE
- GROUP BY
- HAVING
- DISTINCT
- SELECT
- ORDER BY
- LIMIT

به همین دلیل، چون HAVING قبل از SELECT ارزیابی می‌شود، نمی‌توانید از نام مستعار ستون (alias) که در SELECT تعریف کرده‌اید، در HAVING استفاده کنید.

برخلاف WHERE که ردیف‌ها را به صورت تکی و قبل از گروه‌بندی فیلتر می‌کند، HAVING گروه‌ها را پس از گروه‌بندی و بر اساس نتایج توابع تجمعی فیلتر می‌کند. این یعنی ابتدا ردیف‌ها گروه‌بندی می‌شوند و سپس شرط HAVING روی گروه‌ها اعمال می‌شود تا فقط گروه‌هایی که شرط را دارند، نمایش داده شوند.

به عبارت دیگر، شرط موجود در عبارت WHERE را روی ردیف‌ها اعمال می‌کنید در حالی که شرط موجود در عبارت HAVING را برای گروه‌های ردیف اعمال می‌کنید.

مثال!

In [4]:
```
%%sql
CREATE TABLE students_new (
    student_id SERIAL PRIMARY KEY,
    first_name VARCHAR,
    last_name VARCHAR,
    age INT,
    department VARCHAR,
    gpa REAL
);
```

Running query in 'default'

Out[4]:

In [5]:
```
%%sql
INSERT INTO students_new (first_name, last_name, age, department, gpa)
VALUES
('علی', 'محمدی', 20, 'مهندسی کامپیوتر', 18.5),
('سارا', 'احمدی', 21, 'مدیریت', 17.2),
('مرتضی', 'عباس‌پور', 22, 'ریاضیات', 19.3),
('مینا', 'حسینی', 20, 'فیزیک', 16.8),
('رضا', 'رحمانی', 23, 'اقتصاد', 14.5),
('نیما', 'موسوی', 21, 'مهندسی', 18.0),
('آرش', 'بیگی', 22, 'مهندسی کامپیوتر', 19.0),
('نیلوفر', 'سلیمی', 20, 'حقوق', 17.0),
('شهرام', 'نیکوکار', 24, 'فلسفه', 13.5),
('پریسا', 'صادقی', 21, 'مدیریت', 16.2),
('امید', 'امیری', 20, 'مهندسی کامپیوتر', 18.9),
('سیما', 'طاهری', 23, 'مهندسی', 17.8),
('امیر', 'کریمی', 22, 'حقوق', 15.0),
('یاسمین', 'مقدم', 21, 'ریاضیات', 18.2),
('وحید', 'مظفری', 20, 'فیزیک', 14.0),
('آوا', 'شریفی', 22, 'اقتصاد', 19.5),
('فرهاد', 'زارعی', 21, 'مهندسی', 17.5),
('سادف', 'محمدی', 20, 'مهندسی کامپیوتر', 19.0),
('کیان', 'خراسانی', 23, 'فلسفه', 15.5),
('آرش', 'نجفی', 21, 'مدیریت', 18.7) RETURNING *;
```

Running query in 'default'

20 rows affected.

Out[5]:

student_id	first_name	last_name	age	department	gpa
1	علی	محمدی	20	مهندسی کامپیوتر	18.5
2	سارا	احمدی	21	مدیریت	17.2
3	مرتضی	عباس‌پور	22	ریاضیات	19.3
4	مینا	حسینی	20	فیزیک	16.8
5	رضا	رحمانی	23	اقتصاد	14.5
6	نیما	موسوی	21	مهندسی	18.0
7	آرش	بیگی	22	مهندسی کامپیوتر	19.0
8	نیلوفر	سلیمی	20	حقوق	17.0
9	شهرام	نیکوکار	24	فلسفه	13.5
10	پریسا	صادقی	21	مدیریت	16.2
11	امید	امیری	20	مهندسی کامپیوتر	18.9
12	سیما	طاهری	23	مهندسی	17.8
13	امیر	کریمی	22	حقوق	15.0
14	یاسمین	مقدم	21	ریاضیات	18.2
15	وحید	مظفری	20	فیزیک	14.0
16	آوا	شریفی	22	اقتصاد	19.5
17	فرهاد	زارعی	21	مهندسی	17.5
18	سادف	محمدی	20	مهندسی کامپیوتر	19.0
19	کیان	خراسانی	23	فلسفه	15.5
20	آرش	نجفی	21	مدیریت	18.7

فقط رشته‌هایی را نمایش دهید که تعداد دانشجویان آن بیشتر از 2 نفر است.

In [15]:
```
%%sql
SELECT department, COUNT(student_id) AS student_count
FROM students_new
GROUP BY department
HAVING COUNT(student_id) > 2;
```

Running query in 'default'

3 rows affected.

Out[15]:

department	student_count
مهندسی	3
مدیریت	3
مهندسی کامپیوتر	4

توضیح: این کوئری داده‌ها را بر اساس department گروه‌بندی کرده و تعداد دانشجویان هر رشته را محاسبه می‌کند. سپس با استفاده از HAVING COUNT(student_id) > 2 فقط رشته‌هایی که تعداد دانشجویان آن‌ها بیشتر از 2 نفر است نمایش داده می‌شود.

رشته‌هایی که کمترین معدل دانشجویان آن‌ها کمتر از ۱۵ است را پیدا کنید.

(به دنبال یافتن رشته‌های تحصیلی هستیم که حداقل یک دانشجو با معدل زیر ۱۵ در آنها وجود داشته باشد.)

In [27]:
```
%%sql
SELECT department, MIN(gpa) AS min_gpa
FROM students_new
GROUP BY department
HAVING MIN(gpa) < 15;  -- فقط رشته‌هایی که کمترین معدل زیر ۱۵ دارند
```

Running query in 'default'

3 rows affected.

Out[27]:

department	min_gpa
اقتصاد	14.5
فلسفه	13.5
فیزیک	14.0

در اینجا، داده‌ها بر اساس رشته تحصیلی گروه‌بندی می‌شوند و سپس کمترین معدل برای هر رشته محاسبه می‌شود. بعد از آن، تنها رشته‌هایی که کمترین معدل آن‌ها کمتر از ۱۵ است نمایش داده می‌شود.

نمایش رشته‌های تحصیلی که معدل دانشجویان آن‌ها بیشتر از ۱۸ است.

In [16]:
```sql
%%sql
SELECT department, AVG(gpa) AS avg_gpa
FROM students_new
GROUP BY department
HAVING AVG(gpa) > 18;
```

Running query in 'default'

2 rows affected.

Out[16]:

department	avg_gpa
ریاضیات	18.75
مهندسی کامپیوتر	18.84999990463257

این کوئری ابتدا داده‌ها را بر اساس department گروه‌بندی می‌کند، سپس معدل‌ها را محاسبه می‌کند. تنها رشته‌هایی که معدل بیشتر از ۱۸ دارند در نتیجه نمایش داده می‌شوند.

فرض کنید شما جدولی به نام students دارید و می‌خواهید تعداد دانشجویانی که معدلشان بالای ۱۵ است را در هر رشته محاسبه کنید، و سپس فقط رشته‌هایی که تعداد دانشجویان بیشتر از 2 نفر دارند را نمایش دهید.

In [26]:
```sql
%%sql
SELECT department, COUNT(student_id) AS student_count
FROM students_new
WHERE gpa > 15  -- فیلتر کردن داده‌های فردی قبل از گروه‌بندی
GROUP BY department
HAVING COUNT(student_id) > 2;  -- فیلتر کردن گروه‌ها بعد از گروه‌بندی
```

Running query in 'default'

3 rows affected.

Out[26]:

department	student_count
مهندسی	3
مدیریت	3
مهندسی کامپیوتر	4

تعداد دانشجویان در هر رشته تحصیلی که معدل آن‌ها بالای ۱۷ است را محاسبه کنید.

In [18]:
```sql
%%sql
SELECT department, COUNT(student_id) AS student_count
FROM students_new
WHERE gpa > 17
GROUP BY department;
```

Running query in 'default'

5 rows affected.

Out[18]:

department	student_count
اقتصاد	1
ریاضیات	2
مهندسی	3
مدیریت	2
مهندسی کامپیوتر	4

این کوئری ابتدا فیلتر می‌کند که فقط دانشجویانی که معدل بالای ۱۷ دارند در نظر گرفته شوند، سپس تعداد دانشجویان هر رشته تحصیلی را که شرایط مورد نظر را دارند محاسبه می‌کند.

نمایش تعداد دانشجویان در هر گروه سنی که معدل آن‌ها در بازه ۱۵ تا ۲۰ قرار دارد.

In [19]:
```sql
%%sql
SELECT age, COUNT(student_id) AS student_count
FROM students_new
WHERE gpa BETWEEN 15 AND 20
GROUP BY age;
```

Running query in 'default'

4 rows affected.

Out[19]:

age	student_count
20	5
21	6
22	4
23	2

این کوئری داده‌ها را بر اساس age گروه‌بندی می‌کند، اما تنها دانشجویانی که معدل آن‌ها بین ۱۵ و ۲۰ است در نظر گرفته می‌شوند.
سپس تعداد دانشجویان در هر گروه سنی محاسبه می‌شود.

سوال؟

فقط رشته‌هایی رو نشون بده که معدل هیچ‌کدوم از دانشجوهاش زیر 15 نبوده!

In [8]:
```sql
%%sql
SELECT department
FROM students_new
GROUP BY department
HAVING MIN(gpa) >= 15;
```

Running query in 'default'
5 rows affected.

Out[8]:

department
ریاضیات
حقوق
مهندسی
مدیریت
مهندسی کامپیوتر

دستور `GROUPING SETS`

در PostgreSQL، دستور GROUPING SETS یک ویژگی پیشرفته از عبارت GROUP BY است که به شما امکان می‌دهد چندین گروه را در یک پرس و جو ایجاد کنید.

در حالت معمول، با GROUP BY فقط یک مدل گروه‌بندی انجام می‌دیم. ولی با GROUPING SETS می‌تونیم چند مدل مختلف گروه‌بندی رو در یک کوئری بنویسیم.

نحو کلی `GROUPING SETS` به‌صورت زیر است:

```sql
SELECT
  column1,
  column2,
  aggregate_function (column3)
FROM
  table_name
GROUP BY GROUPING SETS (
      (column1, column2),
      (column1),
      (column2),
      ()
   );
```

مثال:

در جدول `inventory_reports` داده‌های مربوط به موجودی کالاهای انبارهای یک شرکت ذخیره شده است. این جدول شامل ستون‌های زیر است:

- `warehouse` : نام شهر محل انبار (تهران، مشهد، اصفهان)
- `brand` : برند کالا (Samsung, Apple, Sony)
- `quantity` : تعداد موجودی از آن برند در آن انبار

با استفاده از دستور `GROUPING SETS` ، کوئری‌ای بنویسید که مجموع موجودی کالاها را به ۴ روش مختلف محاسبه و نمایش دهد:

1. به تفکیک **انبار** و **برند**

2. به تفکیک فقط **انبار** (صرف‌نظر از برند)
3. به تفکیک فقط **برند** (صرف‌نظر از انبار)
4. و در نهایت، **مجموع کلی موجودی تمام کالاها**

نتایج باید به ترتیب بر اساس نام انبار و سپس برند (با `NULLS LAST`) مرتب شده باشند.

In [41]:
```
%%sql
CREATE TABLE inventory_reports (
  warehouse TEXT,
  brand TEXT,
  quantity INT
);
INSERT INTO inventory_reports (warehouse, brand, quantity) VALUES
('Tehran', 'Samsung', 120),
('Tehran', 'Apple', 90),
('Tehran', 'Sony', 60),
('Mashhad', 'Samsung', 80),
('Mashhad', 'Apple', 100),
('Mashhad', 'Sony', 50),
('Isfahan', 'Samsung', 110),
('Isfahan', 'Apple', 70),
('Isfahan', 'Sony', 95) RETURNING *;
```

Running query in 'default'

9 rows affected.

Out[41]:

warehouse	brand	quantity
Tehran	Samsung	120
Tehran	Apple	90
Tehran	Sony	60
Mashhad	Samsung	80
Mashhad	Apple	100
Mashhad	Sony	50
Isfahan	Samsung	110
Isfahan	Apple	70
Isfahan	Sony	95

In [42]:
```
%%sql
SELECT
  warehouse,
  brand,
  SUM(quantity) AS total
FROM
  inventory_reports
GROUP BY
  GROUPING SETS (
    (warehouse, brand),  -- هر انبار و برند
    (warehouse),         -- فقط انبار
    (brand),             -- فقط برند
    ()                   -- مجموع کلی
  )
ORDER BY
  warehouse NULLS LAST,
  brand NULLS LAST;
```

Running query in 'default'

16 rows affected.

Out[42]:

warehouse	brand	total
Isfahan	Apple	70
Isfahan	Samsung	110
Isfahan	Sony	95
Isfahan	None	275
Mashhad	Apple	100
Mashhad	Samsung	80
Mashhad	Sony	50
Mashhad	None	230
Tehran	Apple	90
Tehran	Samsung	120
Tehran	Sony	60
Tehran	None	270
None	Apple	260
None	Samsung	310
None	Sony	205
None	None	775

با استفاده از دستور `GROUPING SETS` کوئری‌ای بنویسید که معدل دانشجویان را به چهار صورت زیر محاسبه و نمایش دهد:

1. به تفکیک **مقطع و رشته**
2. به تفکیک فقط **مقطع تحصیلی**
3. به تفکیک فقط **رشته تحصیلی**
4. معدل **کل دانشجویان**

نتایج را بر اساس مقطع (`grade`) و سپس رشته (`major`) مرتب کنید، به‌گونه‌ای که مقدارهای `NULL` در انتها قرار بگیرند.

In [30]:
```sql
%%sql
SELECT
  grade,
  major,
  AVG(gpa) AS avg_gpa
FROM student
GROUP BY GROUPING SETS (
  (grade, major), -- معدل هر رشته در هر مقطع
  (grade),        -- معدل کل هر مقطع
  (major),        -- معدل کل هر رشته
  ()              -- معدل کل دانشجوها
)
ORDER BY grade NULLS LAST, major NULLS LAST;
```

```
Running query in 'default'
51 rows affected.
```

Out[30]:

grade	major	avg_gpa
دکتری	اخترشناسی	19.799999237060547
دکتری	آمار	18.933333079020183
دکتری	ریاضی	18.90000057220459
دکتری	زمین‌شناسی	18.899999618530273
دکتری	شیمی	18.799999237060547
دکتری	علوم کامپیوتر	18.959999084472656
دکتری	فیزیک	19.5
دکتری	None	19.055999755859375
کارشناسی	اقتصاد	14.5
کارشناسی	آمار	16.22499966621399
کارشناسی	تاریخ	16.100000381469727
کارشناسی	جامعه‌شناسی	15.600000381469727
کارشناسی	حقوق	14.800000190734863
کارشناسی	ریاضی	16.560000038146974
کارشناسی	علوم سیاسی	15.300000190734863
کارشناسی	علوم کامپیوتر	15.199999809265137
کارشناسی	مهندسی	16.5
کارشناسی	هنر	15.899999618530273
کارشناسی	None	15.976470554576201
کارشناسی ارشد	ادبیات	18.200000762939453
کارشناسی ارشد	آمار	17.766666412353516
کارشناسی ارشد	انسان‌شناسی	17.700000762939453
کارشناسی ارشد	بازرگانی	17.5
کارشناسی ارشد	پزشکی	18.200000762939453
کارشناسی ارشد	روان‌شناسی	17.899999618530273
کارشناسی ارشد	ریاضی	17.97499990463257
کارشناسی ارشد	زیست‌شناسی	18.5
کارشناسی ارشد	فلسفه	18.600000381469727
کارشناسی ارشد	None	17.9857143674578
None	اخترشناسی	19.799999237060547
None	ادبیات	18.200000762939453
None	اقتصاد	14.5
None	آمار	17.499999713897704
None	انسان‌شناسی	17.700000762939453
None	بازرگانی	17.5
None	پزشکی	18.200000762939453
None	تاریخ	16.100000381469727
None	جامعه‌شناسی	15.600000381469727
None	حقوق	14.800000190734863
None	روان‌شناسی	17.899999618530273
None	ریاضی	17.500000086697664
None	زمین‌شناسی	18.899999618530273
None	زیست‌شناسی	18.5
None	شیمی	18.799999237060547
None	علوم سیاسی	15.300000190734863
None	علوم کامپیوتر	17.079999446868896
None	فلسفه	18.600000381469727
None	فیزیک	19.5

grade	major	avg_gpa
None	مهندسی	16.5
None	هنر	15.899999618530273
None	None	17.413658490995083

با استفاده از دستور `GROUPING SETS` کوئری‌ای بنویسید که تعداد دانشجویان را به چهار شکل زیر محاسبه و نمایش دهد:

1. به تفکیک **مقطع تحصیلی** و **رشته تحصیلی**
2. به تفکیک فقط **مقطع تحصیلی** (بدون توجه به رشته)
3. به تفکیک فقط **رشته تحصیلی** (بدون توجه به مقطع)
4. **تعداد کل دانشجویان** بدون هیچ گروه‌بندی

نتایج باید بر اساس ستون‌های `grade` و `major` مرتب شوند، به‌گونه‌ای که مقدارهای `NULL` در انتهای جدول نمایش داده شوند.

In [31]:
```sql
%%sql
SELECT
  grade,
  major,
  COUNT(*) AS student_count
FROM student
GROUP BY GROUPING SETS (
  (grade, major),
  (grade),
  (major),
  ()
)
ORDER BY grade NULLS LAST, major NULLS LAST;
```

Running query in 'default'

51 rows affected.

Out[31]:

grade	major	student_count
دکتری	اخترشناسی	1
دکتری	آمار	3
دکتری	ریاضی	2
دکتری	زمین‌شناسی	1
دکتری	شیمی	1
دکتری	علوم کامپیوتر	1
دکتری	فیزیک	1
دکتری	None	10
کارشناسی	اقتصاد	1
کارشناسی	آمار	4
کارشناسی	تاریخ	1
کارشناسی	جامعه‌شناسی	1
کارشناسی	حقوق	1
کارشناسی	ریاضی	5
کارشناسی	علوم سیاسی	1
کارشناسی	علوم کامپیوتر	1
کارشناسی	مهندسی	1
کارشناسی	هنر	1
کارشناسی	None	17
کارشناسی ارشد	ادبیات	1
کارشناسی ارشد	آمار	3
کارشناسی ارشد	انسان‌شناسی	1
کارشناسی ارشد	بازرگانی	1
کارشناسی ارشد	پزشکی	1
کارشناسی ارشد	روان‌شناسی	1
کارشناسی ارشد	ریاضی	4
کارشناسی ارشد	زیست‌شناسی	1
کارشناسی ارشد	فلسفه	1
کارشناسی ارشد	None	14
None	اخترشناسی	1
None	ادبیات	1
None	اقتصاد	1
None	آمار	10
None	انسان‌شناسی	1
None	بازرگانی	1
None	پزشکی	1
None	تاریخ	1
None	جامعه‌شناسی	1
None	حقوق	1
None	روان‌شناسی	1
None	ریاضی	11
None	زمین‌شناسی	1
None	زیست‌شناسی	1
None	شیمی	1
None	علوم سیاسی	1
None	علوم کامپیوتر	2
None	فلسفه	1
None	فیزیک	1

grade	major	student_count
None	مهندسی	1
None	هنر	1
None	None	41

دستور CUBE

دستور CUBE یکی از امکانات قدرتمند برای تحلیل داده‌ها در SQL هست. وقتی چند ستون رو در GROUP BY CUBE(col1, col2, ...) قرار می‌دهیم، این دستور تمام ترکیب‌های ممکن از گروه‌بندی بر اساس اون ستون‌ها رو تولید می‌کند.

نحو کلی CUBE به صورت زیر است:

```
SELECT
  column1,
  column2,
  aggregate_function (column3)
FROM
  table_name
GROUP BY
  CUBE (column1, column2);
```

دستور CUBE روشی کوتاه و مختصر برای تعریف چندین مجموعه گروه‌بندی است، به طوری که عبارات زیر معادل یکدیگرند:

```
CUBE(c1,c2,c3)
=
GROUPING SETS (
    (c1,c2,c3),
    (c1,c2),
    (c1,c3),
    (c2,c3),
    (c1),
    (c2),
    (c3),
    ()
 )
```

مثال

```
CUBE(grade, major) = GROUPING SETS ((grade, major), (grade), (major), ())
```

In [44]:
```
%%sql
SELECT
  grade,
  major,
  COUNT(*) AS student_count
FROM
  student
GROUP BY
  CUBE(grade, major)
ORDER BY
  grade NULLS LAST,
  major NULLS LAST;
```

Running query in 'default'

51 rows affected.

Out[44]:

grade	major	student_count
دکتری	اخترشناسی	1
دکتری	آمار	3
دکتری	ریاضی	2
دکتری	زمین‌شناسی	1
دکتری	شیمی	1
دکتری	علوم کامپیوتر	1
دکتری	فیزیک	1
دکتری	None	10
کارشناسی	اقتصاد	1
کارشناسی	آمار	4
کارشناسی	تاریخ	1
کارشناسی	جامعه‌شناسی	1
کارشناسی	حقوق	1
کارشناسی	ریاضی	5
کارشناسی	علوم سیاسی	1
کارشناسی	علوم کامپیوتر	1
کارشناسی	مهندسی	1
کارشناسی	هنر	1
کارشناسی	None	17
کارشناسی ارشد	ادبیات	1
کارشناسی ارشد	آمار	3
کارشناسی ارشد	انسان‌شناسی	1
کارشناسی ارشد	بازرگانی	1
کارشناسی ارشد	پزشکی	1
کارشناسی ارشد	روان‌شناسی	1
کارشناسی ارشد	ریاضی	4
کارشناسی ارشد	زیست‌شناسی	1
کارشناسی ارشد	فلسفه	1
کارشناسی ارشد	None	14
None	اخترشناسی	1
None	ادبیات	1
None	اقتصاد	1
None	آمار	10
None	انسان‌شناسی	1
None	بازرگانی	1
None	پزشکی	1
None	تاریخ	1
None	جامعه‌شناسی	1
None	حقوق	1
None	روان‌شناسی	1
None	ریاضی	11
None	زمین‌شناسی	1
None	زیست‌شناسی	1
None	شیمی	1
None	علوم سیاسی	1
None	علوم کامپیوتر	2
None	فلسفه	1
None	فیزیک	1

grade	major	student_count
None	مهندسی	1
None	هنر	1
None	None	41

In [45]:
```
%%sql
SELECT
  grade,
  major,
  AVG(gpa) AS avg_gpa
FROM
  student
GROUP BY
  CUBE(grade, major)
ORDER BY
  grade NULLS LAST,
  major NULLS LAST;
```

Running query in 'default'

51 rows affected.

Out[45]:

grade	major	avg_gpa
دکتری	اخترشناسی	19.799999237060547
دکتری	آمار	18.933333079020183
دکتری	ریاضی	18.90000057220459
دکتری	زمین‌شناسی	18.899999618530273
دکتری	شیمی	18.799999237060547
دکتری	علوم کامپیوتر	18.959999084472656
دکتری	فیزیک	19.5
دکتری	None	19.055999755859375
کارشناسی	اقتصاد	14.5
کارشناسی	آمار	16.22499966621399
کارشناسی	تاریخ	16.100000381469727
کارشناسی	جامعه‌شناسی	15.600000381469727
کارشناسی	حقوق	14.800000190734863
کارشناسی	ریاضی	16.560000038146974
کارشناسی	علوم سیاسی	15.300000190734863
کارشناسی	علوم کامپیوتر	15.199999809265137
کارشناسی	مهندسی	16.5
کارشناسی	هنر	15.899999618530273
کارشناسی	None	15.976470554576201
کارشناسی ارشد	ادبیات	18.200000762939453
کارشناسی ارشد	آمار	17.766666412353516
کارشناسی ارشد	انسان‌شناسی	17.700000762939453
کارشناسی ارشد	بازرگانی	17.5
کارشناسی ارشد	پزشکی	18.200000762939453
کارشناسی ارشد	روان‌شناسی	17.899999618530273
کارشناسی ارشد	ریاضی	17.97499990463257
کارشناسی ارشد	زیست‌شناسی	18.5
کارشناسی ارشد	فلسفه	18.600000381469727
کارشناسی ارشد	None	17.9857143674578
None	اخترشناسی	19.799999237060547
None	ادبیات	18.200000762939453
None	اقتصاد	14.5
None	آمار	17.499999713897704
None	انسان‌شناسی	17.700000762939453
None	بازرگانی	17.5
None	پزشکی	18.200000762939453
None	تاریخ	16.100000381469727
None	جامعه‌شناسی	15.600000381469727
None	حقوق	14.800000190734863
None	روان‌شناسی	17.899999618530273
None	ریاضی	17.500000086697664
None	زمین‌شناسی	18.899999618530273
None	زیست‌شناسی	18.5
None	شیمی	18.799999237060547
None	علوم سیاسی	15.300000190734863
None	علوم کامپیوتر	17.079999446868896
None	فلسفه	18.600000381469727
None	فیزیک	19.5

grade	major	avg_gpa
None	مهندسی	16.5
None	هنر	15.899999618530273
None	None	17.413658490995083

با استفاده از دستور CUBE, کوئری‌ای بنویسید که مجموع موجودی کالاها را به تفکیک برند محاسبه کند، همچنین مجموع موجودی کالاها برای تمام برندها را نیز به صورت کلی محاسبه کرده و نمایش دهد.

In [48]:
```sql
%%sql
SELECT
  brand,
  SUM(quantity) total_quantity
FROM
  inventory_reports
GROUP BY
  CUBE (brand)
ORDER BY
  brand NULLS LAST;
```

Running query in 'default'

4 rows affected.

Out[48]:

brand	total_quantity
Apple	260
Samsung	310
Sony	205
None	775

دستور `SELECT DISTINCT`

عبارت SELECT DISTINCT مقادیر منحصر به فرد را از یک جدول بازیابی می‌کند (ردیف‌های تکراری را از مجموعه نتایج حذف می‌کند). در زیر نحو عبارت SELECT DISTINCT آمده است:

```sql
SELECT DISTINCT column1
FROM table_name;
```

مثال

لیست انبارهایی (warehouse) که داریم، بدون تکرار

In [4]:
```sql
%%sql
SELECT DISTINCT warehouse
FROM inventory_reports;
```

Running query in 'default'

3 rows affected.

Out[4]:

warehouse
Mashhad
Tehran
Isfahan

برندهایی که در انبارها هستن، بدون تکرار

In [6]:
```sql
%%sql
SELECT DISTINCT brand
FROM inventory_reports;
```

Running query in 'default'

3 rows affected.

Out[6]:

brand
Sony
Samsung
Apple

In [11]:
```sql
%%sql
DROP TABLE IF EXISTS inventory_reports;
CREATE TABLE inventory_reports (
  warehouse TEXT,
  brand TEXT,
  product TEXT,
  quantity INT
);

INSERT INTO inventory_reports (warehouse, brand, product, quantity) VALUES
-- برندهای در چند انبار
('Tehran',   'Samsung', 'TV', 100),
('Mashhad',  'Samsung', 'TV', 120),
('Isfahan',  'Samsung', 'TV', 90),

('Tehran',   'Sony', 'Camera', 80),
('Isfahan',  'Sony', 'Camera', 95),

-- برندهای فقط در یک انبار
('Tehran',   'LG', 'Monitor', 60),
('Mashhad',  'Huawei', 'Phone', 70),
('Isfahan',  'Xiaomi', 'Tablet', 75),
('Isfahan',  'Nokia', 'Headset', 40) RETURNING *;
```

Running query in 'default'

9 rows affected.

Out[11]:

warehouse	brand	product	quantity
Tehran	Samsung	TV	100
Mashhad	Samsung	TV	120
Isfahan	Samsung	TV	90
Tehran	Sony	Camera	80
Isfahan	Sony	Camera	95
Tehran	LG	Monitor	60
Mashhad	Huawei	Phone	70
Isfahan	Xiaomi	Tablet	75
Isfahan	Nokia	Headset	40

می‌خوایم برندهایی رو پیدا کنیم که در بیش از یک انبار مختلف محصول دارن.

- یعنی مثلاً اگر برند "Samsung" توی Tehran و Mashhad و Isfahan محصول داره → حسابش کن.
- ولی اگه برند "LG" فقط توی Tehran باشه → حسابش نکن.

In [16]:
```sql
%%sql
SELECT brand
FROM inventory_reports
GROUP BY brand
HAVING COUNT(DISTINCT warehouse) > 1;
```

Running query in 'default'

2 rows affected.

Out[16]:

brand
Samsung
Sony

محصولاتی که در بیش از یک انبار موجود هستن

In [25]:
```sql
%%sql
SELECT product
FROM inventory_reports
GROUP BY product
HAVING COUNT(DISTINCT warehouse) > 1;
```

Running query in 'default'

2 rows affected.

Out[25]:

product
Camera
TV

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های آموزشی آمار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

دوشنبه ـ ۸ اردیبهشت ۱۴۰۴

Monday - 27 April 2025

دستور UPDATE

دستور UPDATE به شما امکان می‌دهد داده‌ها را در یک یا چند ستون از یک یا چند ردیف در جدول به‌روز کنید. در زیر نحو اصلی دستور UPDATE آمده است:

```
UPDATE table_name
SET column1 = value1,
    column2 = value2,
    ...
WHERE condition;
```

مثال

افزایش معدل یک دانشجو مشخص:

In [31]:
```
%%sql
SELECT * FROM student
WHERE student_id = 1003;
```

Running query in 'default'

1 rows affected.

Out[31]:

first_name	last_name	age	grade	major	gpa	student_id
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003

In [34]:
```
%%sql
UPDATE student
SET gpa = 17.8
WHERE student_id = 1003 RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[34]:

first_name	last_name	age	grade	major	gpa	student_id
مهدی	حسینی	19	کارشناسی	حقوق	17.8	1003

تغییر رشته برای یک دانشجو:

In [35]:
```
%%sql
SELECT * FROM student
WHERE first_name = 'رضا' AND last_name = 'محمدی';
```

Running query in 'default'

1 rows affected.

Out[35]:

first_name	last_name	age	grade	major	gpa	student_id
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009

In [36]:
```
%%sql
UPDATE student
SET major = 'بیم‌سنجی'
WHERE first_name = 'رضا' AND last_name = 'محمدی' RETURNING *;
```

Running query in 'default'

1 rows affected.

Out[36]:

first_name	last_name	age	grade	major	gpa	student_id
رضا	محمدی	19	کارشناسی	بیم‌سنجی	15.2	1009

تغییر گروه دانشجویان با معدل پایین:

In [44]:
```
%%sql
UPDATE student
SET grade = 'اخراج'
WHERE gpa < 16 RETURNING *;
```

Running query in 'default'

7 rows affected.

Out[44]:

first_name	last_name	age	grade	major	gpa	student_id
لیلا	جعفری	20	اخراج	هنر	15.9	1006
شیرین	نصیری	20	اخراج	جامعه‌شناسی	15.6	1016
نوید	خسروی	19	اخراج	علوم سیاسی	15.3	1019
سینا	بابایی	19	اخراج	ریاضی	15.8	1026
ترانه	مهدوی	19	اخراج	آمار	15.7	1036
رضا	محمدی	19	اخراج	بیم‌سنجی	15.2	1009
پویا	فرهادی	19	اخراج	اقتصاد	14.5	1013

مثال

In [77]:
```
%%sql
DROP TABLE IF EXISTS book_sales;

CREATE TABLE book_sales (
    book_id SERIAL PRIMARY KEY,
    title TEXT,
    author TEXT,
    price REAL,
    quantity_sold INT
);

INSERT INTO book_sales (title, author, price, quantity_sold)
VALUES
('چنین گفت زرتشت', 'فردریش نیچه', 180000, 120),
('غروب بت‌ها', 'فردریش نیچه', 190000, 95),
('بیگانه', 'آلبر کامو', 200000, 140),
('افسانه سیزیف', 'آلبر کامو', 170000, 100),
('جنایت و مکافات', 'فیودور داستایوفسکی', 250000, 110),
('برادران کارامازوف', 'فیودور داستایوفسکی', 280000, 90) RETURNING *;
```

Running query in 'default'

6 rows affected.

Out[77]:

book_id	title	author	price	quantity_sold
1	چنین گفت زرتشت	فردریش نیچه	180000.0	120
2	غروب بت‌ها	فردریش نیچه	190000.0	95
3	بیگانه	آلبر کامو	200000.0	140
4	افسانه سیزیف	آلبر کامو	170000.0	100
5	جنایت و مکافات	فیودور داستایوفسکی	250000.0	110
6	برادران کارامازوف	فیودور داستایوفسکی	280000.0	90

افزایش ۵٪ قیمت همه کتاب‌ها:

In [57]:
```
%%sql
UPDATE book_sales
SET price = price * 1.05
RETURNING *;
```

Running query in 'default'
6 rows affected.

Out[57]:

book_id	title	author	price	quantity_sold
1	چنین گفت زرتشت	فردریش نیچه	189000.0	120
2	غروب بت‌ها	فردریش نیچه	199500.0	95
3	بیگانه	آلبر کامو	210000.0	140
4	افسانه سیزیف	آلبر کامو	178500.0	100
5	جنایت و مکافات	فیودور داستایوفسکی	262500.0	110
6	برادران کارامازوف	فیودور داستایوفسکی	294000.0	90

دستور DELETE

دستور DELETE به شما این امکان را می‌دهد که یک یا چند ردیف از یک جدول را حذف کنید. شکل زیر نحو اصلی عبارت DELETE را نشان می‌دهد:

```
DELETE FROM table_name
WHERE condition;
```

عبارت WHERE اختیاری است. اگر عبارت WHERE را حذف کنید، دستور DELETE تمام ردیف‌های جدول را حذف می‌کند.

مثال

حذف کتابی خاص با عنوان مشخص:

In [58]:
```
%%sql
DELETE FROM book_sales
WHERE title = 'بیگانه'
RETURNING *;
```

Running query in 'default'
1 rows affected.

Out[58]:

book_id	title	author	price	quantity_sold
3	بیگانه	آلبر کامو	210000.0	140

In [59]:
```
%%sql
SELECT * FROM book_sales
```

Running query in 'default'
5 rows affected.

Out[59]:

book_id	title	author	price	quantity_sold
1	چنین گفت زرتشت	فردریش نیچه	189000.0	120
2	غروب بت‌ها	فردریش نیچه	199500.0	95
4	افسانه سیزیف	آلبر کامو	178500.0	100
5	جنایت و مکافات	فیودور داستایوفسکی	262500.0	110
6	برادران کارامازوف	فیودور داستایوفسکی	294000.0	90

حذف تمام کتاب‌هایی که کمتر از 100 نسخه فروختن:

In [61]:
```
%%sql
DELETE FROM book_sales
WHERE quantity_sold < 100
RETURNING *;
```

Running query in 'default'
2 rows affected.

Out[61]:

book_id	title	author	price	quantity_sold
2	غروب بت‌ها	فردریش نیچه	199500.0	95
6	برادران کارامازوف	فیودور داستایوفسکی	294000.0	90

حذف تمام دانشجوهای مقطع کارشناسی:

In [62]:
```
%%sql
DELETE FROM student
WHERE grade = 'کارشناسی'
RETURNING *;
```

Running query in 'default'

17 rows affected.

Out[62]:

first_name	last_name	age	grade	major	gpa	student_id
علی	رضایی	20	کارشناسی	مهندسی	16.5	1001
مهدی	حسینی	19	کارشناسی	حقوق	14.8	1003
لیلا	جعفری	20	کارشناسی	هنر	15.9	1006
رضا	محمدی	19	کارشناسی	علوم کامپیوتر	15.2	1009
سینا	شاهبازی	20	کارشناسی	تاریخ	16.1	1011
پویا	فرهادی	19	کارشناسی	اقتصاد	14.5	1013
شیرین	نصیری	20	کارشناسی	جامعه‌شناسی	15.6	1016
نوید	خسروی	19	کارشناسی	علوم سیاسی	15.3	1019
نیما	کاظمی	20	کارشناسی	ریاضی	16.2	1022
الهام	سجادی	21	کارشناسی	ریاضی	17.4	1025
سینا	بابایی	19	کارشناسی	ریاضی	15.8	1026
نسترن	حبیبی	20	کارشناسی	ریاضی	16.9	1028
مجتبی	موسوی	20	کارشناسی	ریاضی	16.5	1031
فاطمه	نعمتی	20	کارشناسی	آمار	16.4	1034
ترانه	مهدوی	19	کارشناسی	آمار	15.7	1036
نازنین	عبدی	21	کارشناسی	آمار	16.8	1038
پریسا	سهرابی	20	کارشناسی	آمار	16.0	1040

حذف تمام داده‌های جدول با دستور DELETE

تمام داده‌های جدول پاک می‌شن، ولی ساختار جدول (ستون‌ها، تنظیمات) دست‌نخورده می‌مونه.

In [76]:
```
%%sql
DELETE FROM book_sales
RETURNING *;
```

Running query in 'default'

6 rows affected.

Out[76]:

book_id	title	author	price	quantity_sold
1	چنین گفت زرتشت	فردریش نیچه	180000.0	120
2	غروب بت‌ها	فردریش نیچه	190000.0	95
3	بیگانه	آلبر کامو	200000.0	140
4	افسانه سیزیف	آلبر کامو	170000.0	100
5	جنایت و مکافات	فیودور داستایوفسکی	250000.0	110
6	برادران کارامازوف	فیودور داستایوفسکی	280000.0	90

In [70]:
```
%%sql
SELECT * FROM book_sales
```

Running query in 'default'

Out[70]:

book_id	title	author	price	quantity_sold

حذف تمام داده‌های جدول با دستور TRUNCATE

دستور TRUNCATE TABLE همه داده‌ها را از یک جدول خیلی سریع حذف می‌کند. در زیر نحو اصلی عبارت TRUNCATE TABLE آمده است:

```
TRUNCATE TABLE table_name;
```

In [74]:
```
%%sql
TRUNCATE TABLE book_sales;
```

Running query in 'default'

Out[74]:

دستور `LIKE`

عملگر LIKE در PostgreSQL به شما اجازه می‌دهد تا در یک ستون به دنبال الگوی مشخصی بگردید. می‌توانید از عملگر LIKE در بخش WHERE برای فیلتر کردن ردیف‌ها بر اساس الگو استفاده کنید. نحو عملگر LIKE به صورت زیر است:

```
SELECT
  column1, column2
FROM
  table_name
WHERE
  column1 LIKE pattern;
```

معمولاً برای ساختن یک الگو از کاراکترهای جایگزین زیر استفاده می‌شود:

- `%` نمایانگر صفر یا بیشتر از هر کاراکتری است.
- `_` نمایانگر دقیقاً یک کاراکتر است.

In [5]:
```
%%sql

-- جدول داده‌های فارسی
CREATE TABLE students_fa (
    student_id SERIAL PRIMARY KEY,
    first_name VARCHAR,
    last_name VARCHAR,
    grade_level VARCHAR,
    major VARCHAR,
    gpa REAL,
    age INT
);

-- جدول داده‌های انگلیسی
CREATE TABLE students_en (
    student_id SERIAL PRIMARY KEY,
    first_name VARCHAR,
    last_name VARCHAR,
    grade_level VARCHAR,
    major VARCHAR,
    gpa REAL,
    age INT
);


INSERT INTO students_fa (first_name, last_name, grade_level, major, gpa, age) VALUES
('امیر علی', 'کریمی', 'کارشناسی', 'مهندسی کامپیوتر', 16.75, 22),
('مریم', 'حسینی', 'کارشناسی ارشد', 'علوم داده', 17.2, 25),
('سارا', 'میرزایی', 'کارشناسی', 'مهندسی برق', 15.4, 23),
('محمد رضا', 'قاسمی', 'کارشناسی', 'فیزیک', 14.8, 24),
('نگار', 'نیکو منش', 'کارشناسی ارشد', 'مهندسی صنایع', 18.1, 26),
('امیر حسین', 'فرهمند', 'دکتری', 'زیست شناسی', 17.95, 29),
('شایان', 'جمشیدی', 'کارشناسی', 'ریاضی کاربردی', 15.75, 22),
('الناز', 'بهرامی', 'کارشناسی ارشد', 'معماری', 18.5, 27),
('رضا', 'جهان آرای', 'کارشناسی', 'شیمی', 16.0, 23),
('امیر حسن', 'صالح نژاد', 'کارشناسی', 'مهندسی عمران', 14.6, 24);


INSERT INTO students_en (first_name, last_name, grade_level, major, gpa, age) VALUES
('Amir Ali', 'Karimi', 'Bachelor', 'Computer Engineering', 16.75, 22),
('Maryam', 'Hosseini', 'Master', 'Data Science', 17.2, 25),
('Sara', 'Mirzaei', 'Bachelor', 'Electrical Engineering', 15.4, 23),
('Mohammad Reza', 'Ghasemi', 'Bachelor', 'Physics', 14.8, 24),
('Negar', 'Nikou Manesh', 'Master', 'Industrial Engineering', 18.1, 26),
('Amir Hossein', 'Farahmand', 'PhD', 'Biology', 17.95, 29),
('Shayan', 'Jamshidi', 'Bachelor', 'Applied Mathematics', 15.75, 22),
('Elnaz', 'Bahrami', 'Master', 'Architecture', 18.5, 27),
('Reza', 'Jahan Ara', 'Bachelor', 'Chemistry', 16.0, 23),
('Amir Hassan', 'Saleh Nejad', 'Bachelor', 'Civil Engineering', 14.6, 24);
```

Running query in 'default'

10 rows affected.

10 rows affected.

Out[5]:

پیدا کردن همه کسانی که اسمشون با 'امیر' شروع میشه

In [9]:
```
%%sql
SELECT * FROM students_fa
```

```
WHERE first_name LIKE 'امیر%';
```

Running query in 'default'

3 rows affected.

Out[9]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	امیر علی	کریمی	کارشناسی	مهندسی کامپیوتر	16.75	22
6	امیر حسین	فرهمند	دکتری	زیست شناسی	17.95	29
10	امیر حسن	صالح نژاد	کارشناسی	مهندسی عمران	14.6	24

In [17]:

```
%%sql
SELECT * FROM students_en
WHERE first_name LIKE 'Amir%';
```

Running query in 'default'

3 rows affected.

Out[17]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	Amir Ali	Karimi	Bachelor	Computer Engineering	16.75	22
6	Amir Hossein	Farahmand	PhD	Biology	17.95	29
10	Amir Hassan	Saleh Nejad	Bachelor	Civil Engineering	14.6	24

پیدا کردن کسانی که فامیلیشون شامل 'منش' است

In [11]:

```
%%sql
SELECT * FROM students_fa
WHERE last_name LIKE '%منش%';
```

Running query in 'default'

1 rows affected.

Out[11]:

student_id	first_name	last_name	grade_level	major	gpa	age
5	نگار	نیکو منش	کارشناسی ارشد	مهندسی صنایع	18.1	26

In [18]:

```
%%sql
SELECT * FROM students_en
WHERE last_name LIKE '%Manesh%';
```

Running query in 'default'

1 rows affected.

Out[18]:

student_id	first_name	last_name	grade_level	major	gpa	age
5	Negar	Nikou Manesh	Master	Industrial Engineering	18.1	26

پیدا کردن کسانی که اسمشون شامل 'محمد' است

In [12]:

```
%%sql
SELECT * FROM students_fa
WHERE first_name LIKE '%محمد%';
```

Running query in 'default'

1 rows affected.

Out[12]:

student_id	first_name	last_name	grade_level	major	gpa	age
4	محمد رضا	قاسمی	کارشناسی	فیزیک	14.8	24

In [19]:

```
%%sql
SELECT * FROM students_en
WHERE first_name LIKE '%Mohammad%';
```

Running query in 'default'

1 rows affected.

Out[19]:

student_id	first_name	last_name	grade_level	major	gpa	age
4	Mohammad Reza	Ghasemi	Bachelor	Physics	14.8	24

پیدا کردن کسانی که فامیلیشون با 'ج' شروع میشه

In [13]:

```
%%sql
SELECT * FROM students_fa
WHERE last_name LIKE 'ج%';
```

Running query in 'default'

2 rows affected.

Out[13]:

student_id	first_name	last_name	grade_level	major	gpa	age
7	شایان	جمشیدی	کارشناسی	ریاضی کاربردی	15.75	22
9	رضا	جهان آرای	کارشناسی	شیمی	16.0	23

In [20]:
```
%%sql
SELECT * FROM students_en
WHERE last_name LIKE 'J%';
```

Running query in 'default'

2 rows affected.

Out[20]:

student_id	first_name	last_name	grade_level	major	gpa	age
7	Shayan	Jamshidi	Bachelor	Applied Mathematics	15.75	22
9	Reza	Jahan Ara	Bachelor	Chemistry	16.0	23

پیدا کردن کسانی که فامیلیشون با 'ی' تمام میشه

In [14]:
```
%%sql
SELECT * FROM students_fa
WHERE last_name LIKE '%ی';
```

Running query in 'default'

7 rows affected.

Out[14]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	امیر علی	کریمی	کارشناسی	مهندسی کامپیوتر	16.75	22
2	مریم	حسینی	کارشناسی ارشد	علوم داده	17.2	25
3	سارا	میرزایی	کارشناسی	مهندسی برق	15.4	23
4	محمد رضا	قاسمی	کارشناسی	فیزیک	14.8	24
7	شایان	جمشیدی	کارشناسی	ریاضی کاربردی	15.75	22
8	الناز	بهرامی	کارشناسی ارشد	معماری	18.5	27
9	رضا	جهان آرای	کارشناسی	شیمی	16.0	23

اسم دقیقا ۵ کاراکتر

In [22]:
```
%%sql
SELECT * FROM students_fa
WHERE first_name LIKE '_____';
```

Running query in 'default'

2 rows affected.

Out[22]:

student_id	first_name	last_name	grade_level	major	gpa	age
7	شایان	جمشیدی	کارشناسی	ریاضی کاربردی	15.75	22
8	الناز	بهرامی	کارشناسی ارشد	معماری	18.5	27

In [23]:
```
%%sql
SELECT * FROM students_en
WHERE first_name LIKE '_____';
```

Running query in 'default'

2 rows affected.

Out[23]:

student_id	first_name	last_name	grade_level	major	gpa	age
5	Negar	Nikou Manesh	Master	Industrial Engineering	18.1	26
8	Elnaz	Bahrami	Master	Architecture	18.5	27

فامیل دقیقا ۸ کاراکتر

In [24]:
```
%%sql
SELECT * FROM students_fa
WHERE last_name LIKE '________';
```

Running query in 'default'

1 rows affected.

Out[24]:

student_id	first_name	last_name	grade_level	major	gpa	age
5	نگار	نیکو منش	کارشناسی ارشد	مهندسی صنایع	18.1	26

In [49]:
```
%%sql
SELECT * FROM students_en
WHERE last_name LIKE '________';
```

Running query in 'default'

2 rows affected.

Out[49]:

student_id	first_name	last_name	grade_level	major	gpa	age
2	Maryam	Hosseini	Master	Data Science	17.2	25
7	Shayan	Jamshidi	Bachelor	Applied Mathematics	15.75	22

دستور `ILIKE`

PostgreSQL عملگر ILIKE را ارائه می‌دهد که مشابه LIKE است اما تطبیق الگو را به صورت بدون حساسیت به حروف بزرگ و کوچک انجام می‌دهد

جستجوی کسانی که اسمشون شامل "am" باشه (به‌طور غیر حساس به حروف بزرگ و کوچک):

In [8]:
```
%%sql
SELECT * FROM students_en
WHERE first_name ILIKE '%am%';
```

Running query in 'default'

5 rows affected.

Out[8]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	Amir Ali	Karimi	Bachelor	Computer Engineering	16.75	22
2	Maryam	Hosseini	Master	Data Science	17.2	25
4	Mohammad Reza	Ghasemi	Bachelor	Physics	14.8	24
6	Amir Hossein	Farahmand	PhD	Biology	17.95	29
10	Amir Hassan	Saleh Nejad	Bachelor	Civil Engineering	14.6	24

جستجو برای کسانی که اسمشون با "sh" شروع بشه:

In [47]:
```
%%sql
SELECT * FROM students_en
WHERE first_name ILIKE 'sh%';
```

Running query in 'default'

1 rows affected.

Out[47]:

student_id	first_name	last_name	grade_level	major	gpa	age
7	Shayan	Jamshidi	Bachelor	Applied Mathematics	15.75	22

اسم‌هایی که حرف دومشون a باشه و بعدش هرچی بود مهم نیست

- اولین کاراکتر: هر چیزی (`_`)
- دومین کاراکتر: دقیقاً `a`
- بعدش: هرچندتا کاراکتر باشه (`%`)

In [53]:
```
%%sql
SELECT * FROM students_en
WHERE first_name ILIKE '_a%';
```

Running query in 'default'

2 rows affected.

Out[53]:

student_id	first_name	last_name	grade_level	major	gpa	age
2	Maryam	Hosseini	Master	Data Science	17.2	25
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23

اسم‌هایی که 4 کاراکتر دارن و سومین حرفشون m هست

- دو تا کاراکتر اول: هرچی (`__`)
- کاراکتر سوم: دقیقا `m`
- کاراکتر چهارم: هرچی (`_`)

In [64]:
```sql
%%sql
SELECT * FROM students_en
WHERE first_name ILIKE '__r_';
```

Running query in 'default'

1 rows affected.

Out[64]:

student_id	first_name	last_name	grade_level	major	gpa	age
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23

دستور `NOT LIKE`

عملگر NOT نتیجه عملگرهای LIKE و ILIKE را نفی می‌کند:

```sql
SELECT select_list
FROM table_name
WHERE column1 NOT LIKE pattern;

SELECT select_list
FROM table_name
WHERE column1 NOT ILIKE pattern;
```

پیدا کردن اسم‌هایی که سومین حرفشان a نیست

In [78]:
```sql
%%sql

SELECT *
FROM students_en
WHERE first_name NOT ILIKE '__a%';
```

Running query in 'default'

9 rows affected.

Out[78]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	Amir Ali	Karimi	Bachelor	Computer Engineering	16.75	22
2	Maryam	Hosseini	Master	Data Science	17.2	25
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23
4	Mohammad Reza	Ghasemi	Bachelor	Physics	14.8	24
5	Negar	Nikou Manesh	Master	Industrial Engineering	18.1	26
6	Amir Hossein	Farahmand	PhD	Biology	17.95	29
8	Elnaz	Bahrami	Master	Architecture	18.5	27
9	Reza	Jahan Ara	Bachelor	Chemistry	16.0	23
10	Amir Hassan	Saleh Nejad	Bachelor	Civil Engineering	14.6	24

پیدا کردن اسم‌هایی که دقیقا ۵ حرف دارند و دومین حرفشان l نیست

In [82]:
```sql
%%sql

SELECT *
FROM students_en
WHERE first_name NOT ILIKE '_l___';
```

Running query in 'default'

9 rows affected.

Out[82]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	Amir Ali	Karimi	Bachelor	Computer Engineering	16.75	22
2	Maryam	Hosseini	Master	Data Science	17.2	25
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23
4	Mohammad Reza	Ghasemi	Bachelor	Physics	14.8	24
5	Negar	Nikou Manesh	Master	Industrial Engineering	18.1	26
6	Amir Hossein	Farahmand	PhD	Biology	17.95	29
7	Shayan	Jamshidi	Bachelor	Applied Mathematics	15.75	22
9	Reza	Jahan Ara	Bachelor	Chemistry	16.0	23
10	Amir Hassan	Saleh Nejad	Bachelor	Civil Engineering	14.6	24

پیدا کردن اسم‌هایی که با a شروع نمی‌شوند

In [84]:
```
%%sql

SELECT *
FROM students_en
WHERE first_name NOT ILIKE 'a%';
```

Running query in 'default'

7 rows affected.

Out[84]:

student_id	first_name	last_name	grade_level	major	gpa	age
2	Maryam	Hosseini	Master	Data Science	17.2	25
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23
4	Mohammad Reza	Ghasemi	Bachelor	Physics	14.8	24
5	Negar	Nikou Manesh	Master	Industrial Engineering	18.1	26
7	Shayan	Jamshidi	Bachelor	Applied Mathematics	15.75	22
8	Elnaz	Bahrami	Master	Architecture	18.5	27
9	Reza	Jahan Ara	Bachelor	Chemistry	16.0	23

PostgreSQL تعدادی عملگر فراهم کرده است که عملکرد عملگرهای ILIKE، NOT LIKE، LIKE و NOT ILIKE را شبیه‌سازی می‌کنند.

Operator	Meaning
~~	LIKE
~~*	ILIKE
!~~	NOT LIKE
!~~*	NOT ILIKE

In [97]:
```
%%sql
SELECT * FROM students_en
WHERE first_name ~~* '_a%';
```

Running query in 'default'

2 rows affected.

Out[97]:

student_id	first_name	last_name	grade_level	major	gpa	age
2	Maryam	Hosseini	Master	Data Science	17.2	25
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23

In [98]:
```
%%sql

SELECT *
FROM students_en
WHERE first_name !~~* '__a%';
```

Running query in 'default'

9 rows affected.

Out[98]:

student_id	first_name	last_name	grade_level	major	gpa	age
1	Amir Ali	Karimi	Bachelor	Computer Engineering	16.75	22
2	Maryam	Hosseini	Master	Data Science	17.2	25
3	Sara	Mirzaei	Bachelor	Electrical Engineering	15.4	23
4	Mohammad Reza	Ghasemi	Bachelor	Physics	14.8	24
5	Negar	Nikou Manesh	Master	Industrial Engineering	18.1	26
6	Amir Hossein	Farahmand	PhD	Biology	17.95	29
8	Elnaz	Bahrami	Master	Architecture	18.5	27
9	Reza	Jahan Ara	Bachelor	Chemistry	16.0	23
10	Amir Hassan	Saleh Nejad	Bachelor	Civil Engineering	14.6	24

In [ ]:

پایگاه داده‌ها

دانشگاه شهید بهشتی، دانشکده علوم ریاضی، گروه‌های آموزشی آمار و علوم کامپیوتر

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

مدرس: **میلاد وزان**

دوشنبه ـ ۱۵ اردیبهشت ۱۴۰۴

Monday - 2025 05 May

مطالب بعد از میان‌ترم

In [1]:
```
%load_ext sql
```

Connecting to 'default'

In [2]:
```
%config SqlMagic.displaylimit = None
```

displaylimit: Value None will be treated as 0 (no limit)

In [16]:
```
%%sql
DROP TABLE students;
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR
);

INSERT INTO students (name) VALUES
('Niloofar'), ('Milad'), ('Artin'), ('Reza'), ('Arash'), ('Mehdi'),
('Hamed'), ('Fatemeh'), ('Mohammad'), ('Maryam'), ('Zahra');
```

Running query in 'default'

11 rows affected.

Out[16]:

In [17]:
```
%%sql
DROP TABLE IF EXISTS employees;
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50)
);

INSERT INTO employees (name) VALUES
('Hamed'), ('Zahra'), ('Reza'), ('Sina'), ('Ali'),
('Mohammad'), ('Leila'), ('Navid'), ('Elham'), ('Kian'),
('Niloofar'), ('Milad'), ('Artin'), ('Sorena');
```

Running query in 'default'

14 rows affected.

Out[17]:

In [18]:
```
%%sql
CREATE TABLE warehouse_a (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(50)
);

INSERT INTO warehouse_a VALUES
(1, 'Laptop'), (2, 'Mouse'), (3, 'Keyboard'), (4, 'Monitor'), (5, 'Printer'),
(6, 'Speaker'), (7, 'Webcam'), (8, 'Microphone'), (9, 'USB Drive'), (10, 'Charger');
```

Running query in 'default'

10 rows affected.

Out[18]:

In [19]:
```
%%sql

CREATE TABLE warehouse_b (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(50)
);
```

```
INSERT INTO warehouse_b VALUES
(3, 'Keyboard'), (4, 'Monitor'), (5, 'Printer'), (6, 'Speaker'), (11, 'Tablet'),
(12, 'Router'), (13, 'SSD'), (14, 'RAM'), (15, 'GPU'), (10, 'Charger');
```

Running query in 'default'

10 rows affected.

Out[19]:

عملگرهای مجموعه

در PostgreSQL، عملگرهای مجموعه (Set Operators) برای ترکیب نتایج چند کوئری SELECT استفاده می‌شوند، به شکلی که با مجموعه‌ای از داده‌ها مانند اجتماع، اشتراک یا تفاضل رفتار می‌کنند؛ مشابه نظریه مجموعه‌ها در ریاضیات.

عملگر `UNION`

عملگر `UNION` به شما این امکان را می‌دهد که مجموعه نتایج دو یا چند کوئری را در یک مجموعه نتیجه واحد ترکیب کنید (با حذف مقادیر تکراری).

```
SELECT column1, column2
FROM tableA;
UNION
SELECT column1, column2
FROM tableB;
```

عملگر `UNION ALL`

گر می‌خواهید تکرارها حفظ شوند، از UNION ALL استفاده کنید.

```
SELECT column1, column2
FROM tableA;
UNION ALL
SELECT column1, column2
FROM tableB;
```

عملگر `INTERSECT`

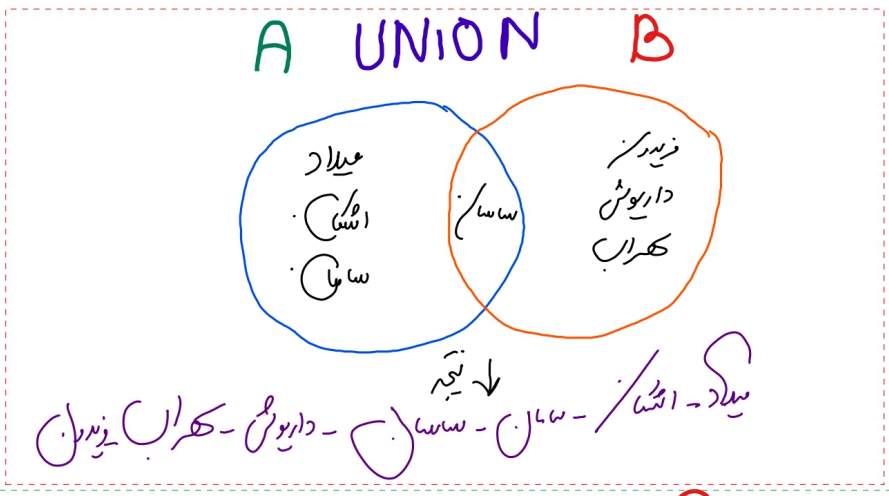
عملگر INTERSECT رکوردهای مشترک مجموعه نتایج دو کوئری را برمی‌گرداند. فقط ردیف‌هایی را برمی‌گردانند که در هر دو نتیجه وجود دارند (اشتراک).

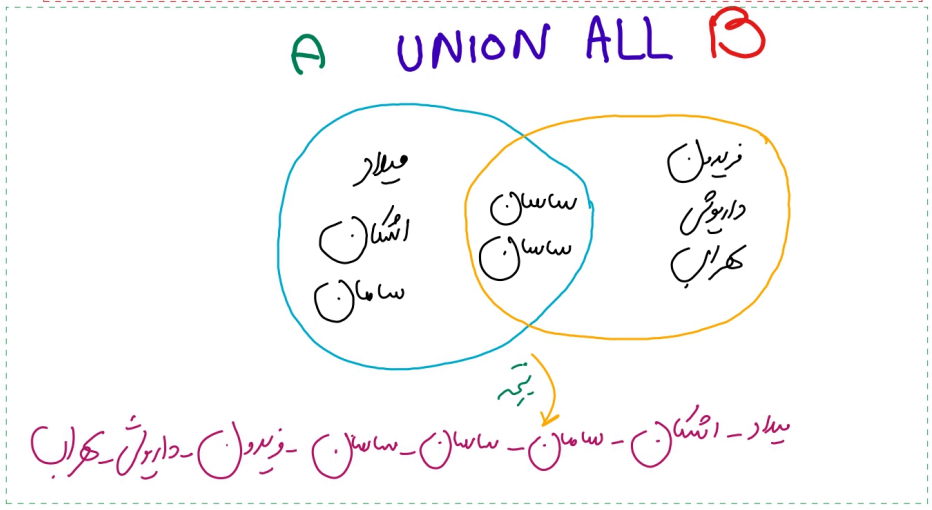
```
SELECT column1, column2
FROM table1;
INTERSECT
SELECT column1, column2
FROM table2;
```

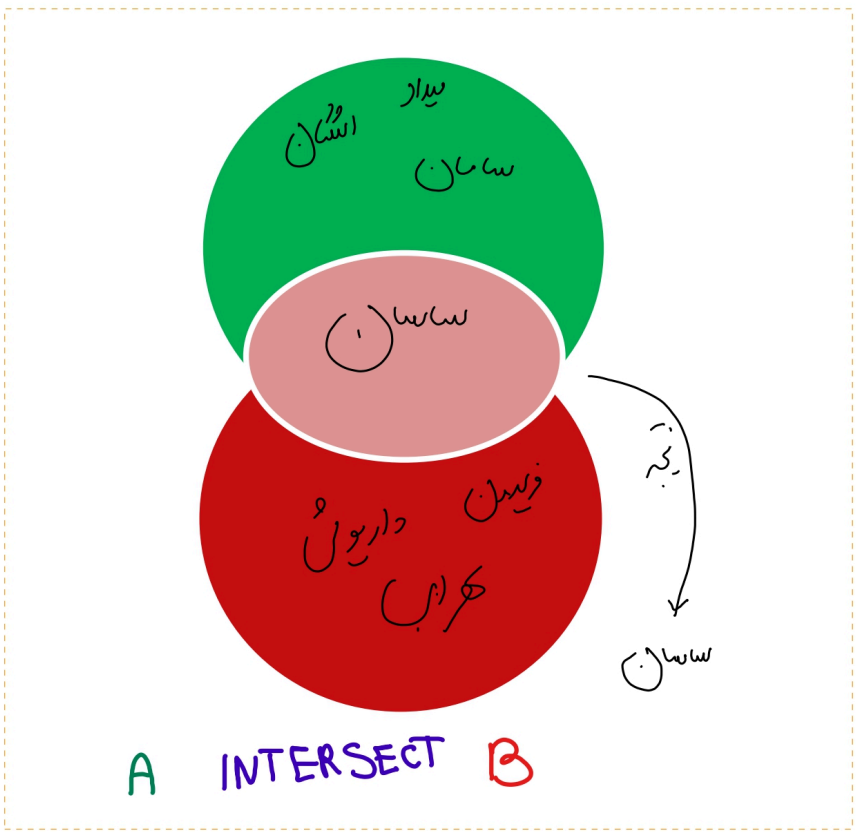
عملگر `EXCEPT`

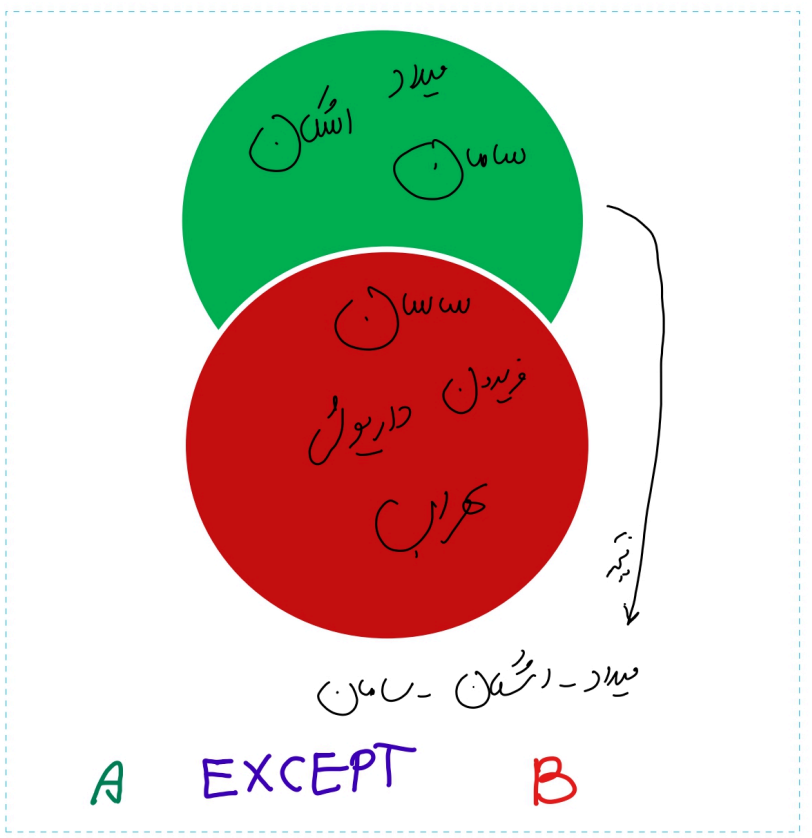
عملگر EXCEPT به شما امکان می‌دهد دو کوئری را ترکیب کنید و تفاضل بین مجموعه‌های نتیجه را پیدا کنید. عملگر EXCEPT سطرهایی را از مجموعه نتایج اول که در مجموعه دوم وجود ندارد برمی‌گرداند. نتیجه‌ی SELECT اول منهای موارد مشترک با SELECT دوم (تفاضل).

```
SELECT column1, column2
FROM tableA
EXCEPT
SELECT column1, column2
FROM tableB;
```









قوانین استفاده از عملگرهای مجموعه

1. دو `SELECT` باید:

- **تعداد ستون‌های برابر** داشته باشند.
- **نوع داده‌ی ستون‌ها** قابل مقایسه (compatible) باشد.

2. می‌توانند شامل `ORDER BY`, `LIMIT`, `OFFSET` باشند **اما فقط در آخر کل عملیات.**

مثال‌های `UNION`

لیست همه دانشجویان و کارمندان

In [29]:
```
%%sql
SELECT name FROM students
UNION
SELECT name FROM employees;
```

Running query in 'default'

18 rows affected.

Out[29]:

name
Zahra
Milad
Mohammad
Elham
Ali
Kian
Sina
Navid
Maryam
Hamed
Artin
Mehdi
Niloofar
Reza
Arash
Fatemeh
Leila
Sorena

In [30]:
```
%%sql
SELECT name FROM students
UNION ALL
SELECT name FROM employees;
```

Running query in 'default'

25 rows affected.

Out[30]:

name
Niloofar
Milad
Artin
Reza
Arash
Mehdi
Hamed
Fatemeh
Mohammad
Maryam
Zahra
Hamed
Zahra
Reza
Sina
Ali
Mohammad
Leila
Navid
Elham
Kian
Niloofar
Milad
Artin
Sorena

نام تمام محصولات موجود در دو انبار مختلف

In [31]:
```
%%sql
SELECT product_name FROM warehouse_a
UNION
SELECT product_name FROM warehouse_b;
```

Running query in 'default'

15 rows affected.

Out[31]:

product_name
Mouse
SSD
Tablet
Webcam
Monitor
Router
USB Drive
Speaker
Printer
GPU
RAM
Laptop
Keyboard
Microphone
Charger

مثال‌های INTERSECT

افرادی که هم دانشجو هستند و هم کارمند

In [33]:
```
%%sql
SELECT name FROM students
INTERSECT
SELECT name FROM employees;
```

Running query in 'default'

7 rows affected.

Out[33]:

name
Zahra
Hamed
Artin
Milad
Mohammad
Niloofar
Reza

محصولات مشترک در دو انبار

In [34]:
```
%%sql
SELECT product_name FROM warehouse_a
INTERSECT
SELECT product_name FROM warehouse_b;
```

Running query in 'default'

5 rows affected.

Out[34]:

product_name
Charger
Printer
Speaker
Keyboard
Monitor

مثال‌های EXCEPT

دانشجویانی که کارمند نیستند

In [35]:
```
%%sql
SELECT name FROM students
EXCEPT
SELECT name FROM employees;
```

Running query in 'default'

4 rows affected.

Out[35]:

name
Maryam
Mehdi
Arash
Fatemeh

محصولاتی که فقط در انبار A هستند

In [36]:
```
%%sql
SELECT product_name FROM warehouse_a
EXCEPT
SELECT product_name FROM warehouse_b;
```

Running query in 'default'

5 rows affected.

Out[36]:

product_name
Webcam
Mouse
Microphone
USB Drive
Laptop

استفاده از `ORDER BY`

In [41]:
```
%%sql
SELECT name FROM students
UNION
SELECT name FROM employees
ORDER BY name ASC
LIMIT 5 OFFSET 1;
```

Running query in 'default'

5 rows affected.

Out[41]:

name
Arash
Artin
Elham
Fatemeh
Hamed

In [43]:
```
%%sql

-- این باعث خطا می‌شود
SELECT name FROM students ORDER BY name
UNION
SELECT name FROM employees;
```

Running query in 'default'

```
RuntimeError: If using snippets, you may pass the --with argument explicitly.
For more details please refer: https://jupysql.ploomber.io/en/latest/compose.html#with-argument


Original error message from DB driver:
(psycopg2.errors.SyntaxError) syntax error at or near "UNION"
LINE 2: UNION
        ^

[SQL: SELECT name FROM students ORDER BY name
UNION
SELECT name FROM employees;]
(Background on this error at: https://sqlalche.me/e/20/f405)

If you need help solving this issue, send us a message: https://ploomber.io/community
```

کلید خارجی

در SQL، کلید خارجی ستونی یا مجموعه‌ای از ستون‌ها در یک جدول است که به کلید اصلی جدول دیگری اشاره می‌کند. جدولی که دارای کلید خارجی است، جدول فرزند یا جدول کلید خارجی یا جدول ارجاع‌دهنده (referencing) نامیده می‌شود. جدولی که ستون‌های کلید اصلی آن توسط یک کلید خارجی ارجاع داده می‌شوند، جدول والد یا جدول مرجع (reference) نامیده می‌شود. هدف اصلی کلید خارجی، ایجاد رابطه بین جداول فرزند و والد است. برای هر مقدار در کلید خارجی در جدول فرزند، همواره می‌توانید مقادیر متناظر را در کلید اصلی جدول والد پیدا کنید.

در زیر نحو پایه برای تعریف یک محدودیت کلید خارجی آمده است:

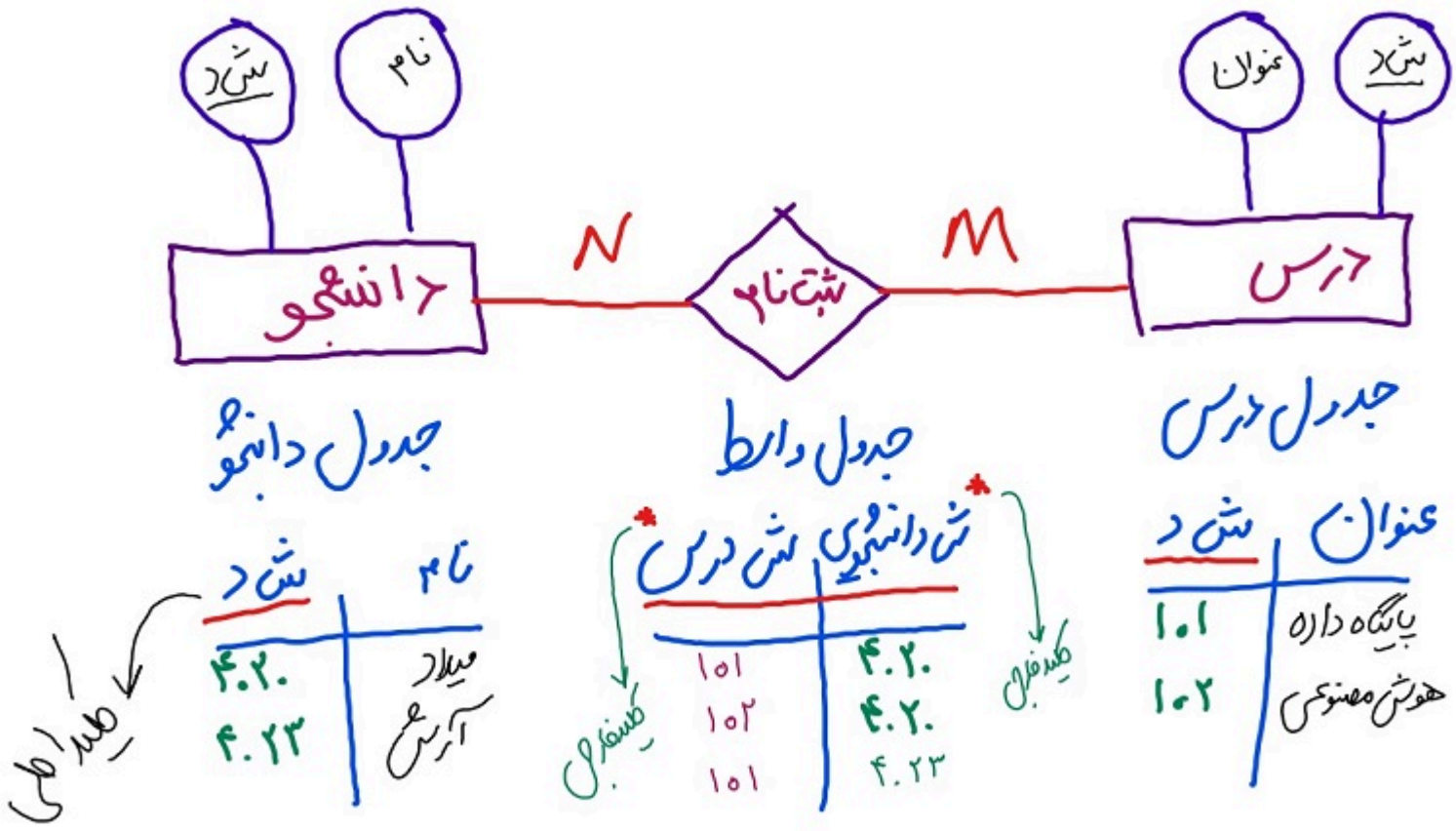
```
CONSTRAINT constraint_name
FOREIGN KEY (fk_column)
REFERENCES table(pk_column)
ON DELETE delete_action
ON UPDATE update_action;
```

مثال

رابطه‌ی چند به چند

یک دانشجو می‌تواند در چندین دوره (درس) ثبت‌نام کند.

یک دوره (درس) نیز می‌تواند چندین دانشجو داشته باشد.



In [49]:
```sql
%%sql
DROP TABLE IF EXISTS students;
DROP TABLE IF EXISTS courses;

-- جدول دانشجویان
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR NOT NULL
);

-- جدول دوره‌ها
CREATE TABLE courses (
    id SERIAL PRIMARY KEY,
    title VARCHAR NOT NULL
);

-- جدول واسط: ثبت‌نام‌ها
CREATE TABLE enrollments (
    student_id INTEGER,
    course_id INTEGER,
    PRIMARY KEY (student_id, course_id),

    FOREIGN KEY (student_id)
    REFERENCES students(id),

    FOREIGN KEY (course_id)
    REFERENCES courses(id)
);
```

Running query in 'default'

Out[49]:

In [50]:
```sql
%%sql

-- اضافه کردن دانشجوها
INSERT INTO students (name) VALUES
('Ali Rezaei'),
('Sara Mohammadi'),
('Nima Hosseini');

-- اضافه کردن دوره‌ها
INSERT INTO courses (title) VALUES
('Database Systems'),
('Operating Systems'),
('Web Development');

-- ثبت‌نام دانشجوها در دوره‌ها
INSERT INTO enrollments (student_id, course_id) VALUES
(1, 1),  -- Ali in Database Systems
(1, 3),  -- Ali in Web Development
(2, 1),  -- Sara in Database Systems
```

```
(2, 2),  -- Sara in Operating Systems
(3, 2),  -- Nima in Operating Systems
(3, 3);  -- Nima in Web Development
```

Running query in 'default'

3 rows affected.

3 rows affected.

6 rows affected.

Out[50]:

خطاهای ناشی از نقض محدودیت (قید) کلید خارجی در پایگاه‌داده‌ها

در پایگاه‌داده‌های رابطه‌ای مانند PostgreSQL، *محدودیت کلید خارجی (Foreign Key Constraint)* تضمین می‌کند که مقدار در یک ستون (یا مجموعه ستون‌ها) در یک جدول، **به مقدار موجود در ستون کلید اصلی جدول دیگر اشاره می‌کند.**

اگر تلاش شود تا داده‌ای درج یا به‌روزرسانی شود که این قید را نقض کند، سیستم پایگاه‌داده جلوی آن عملیات را گرفته و خطا صادر می‌کند. این رفتار تضمین می‌کند که *یکپارچگی مرجع (Referential Integrity)* حفظ شود.

یکی از رایج‌ترین خطاها زمانی رخ می‌دهد که **رکوردی در یک جدول فرعی (مثلاً `enrollments`) درج شود، در حالی که مقدار ارجاع‌شده در جدول والد (مثلاً `students` یا `courses`) وجود ندارد.**

مثال از کوئری‌ای که خطا تولید می‌کند:

In [53]:
```
%%sql
-- تلاش برای ثبت‌نام دانشجویی که وجود ندارد
INSERT INTO enrollments (student_id, course_id)
VALUES (999, 888);
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.ForeignKeyViolation) insert or update on table "enrollments" violates foreign key constraint "en
rollments_student_id_fkey"
DETAIL:  Key (student_id)=(999) is not present in table "students".

[SQL: INSERT INTO enrollments (student_id, course_id)
VALUES (999, 888);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
If you need help solving this issue, send us a message: https://ploomber.io/community
```

استفاده صحیح از کلیدهای خارجی نقش کلیدی در حفظ صحت و سازگاری داده‌ها در پایگاه‌داده دارد. هرگونه تلاش برای درج داده بدون وجود مقادیر مرجع معتبر، باعث ایجاد خطای "violates foreign key constraint" خواهد شد

مثال رابطه‌ی یک به چند

هر شرکت می‌تواند چندین کارمند داشته باشد.

هر کارمند فقط در یک شرکت کار می‌کند.

In [67]:
```
%%sql
```

```sql
-- جدول شرکت‌ها
CREATE TABLE companies (
    id SERIAL PRIMARY KEY,
    name VARCHAR NOT NULL
);

-- جدول کارمندان
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR NOT NULL,
    company_id INTEGER NOT NULL,
    FOREIGN KEY (company_id) REFERENCES companies(id)
);
```

Running query in 'default'

Out[67]:

In [68]:
```sql
%%sql

-- اضافه کردن شرکت‌ها
INSERT INTO companies (name) VALUES
('TechVision Co.'),
('GreenSoft Inc.'),
('FutureWare Ltd.');

-- اضافه کردن کارمندان
INSERT INTO employees (name, company_id) VALUES
('Ali Rezaei', 1),     -- کارمند شرکت TechVision
('Sara Mohammadi', 1), -- کارمند شرکت TechVision
('Nima Hosseini', 2),  -- کارمند شرکت GreenSoft
('Leila Ahmadi', 3),   -- کارمند شرکت FutureWare
('Reza Karimi', 2);    -- کارمند شرکت GreenSoft
```

Running query in 'default'

3 rows affected.

5 rows affected.

Out[68]:

Deletion actions

هنگامی که یک ردیف را در جدول والد حذف می‌کنید، سیستم پایگاه داده باید تصمیم بگیرد که با ردیف در جدول فرزند چه کاری انجام دهد. به طور پیش فرض، سیستم پایگاه داده به شما اجازه نمی‌دهد یک ردیف را در جدول والد حذف کنید، اگر ردیف‌های مرتبط با آن در جدول فرزند وجود داشته باشد. اگر سعی کنید این کار را انجام دهید، سیستم پایگاه داده حذف را رد می‌کند و با خطا مواجه می شود.

عمل حذف به شما اجازه می‌دهد تا زمانی که ردیف‌های جدول والد حذف می‌شوند، عملکردی را در ردیف‌های جداول فرزند مشخص کنید:

```sql
ON DELETE delete_action;
```

delete_action می‌تواند یکی از موارد زیر باشد:

NO ACTION – یک خطای نقض محدودیت صادر می‌کند (وقتی یک ردیف در جدول فرزند به مقدار موجود در جدول والد اشاره دارد، نمی‌توانی آن ردیف را از جدول والد حذف کنی). NO ACTION رفتار پیش‌فرض است.

SET NULL – مقادیر ستون کلید خارجی را برابر NULL قرار می‌دهد.

CASCADE – همه ردیف‌های مرتبط در جدول فرزند را حذف می‌کند.

SET DEFAULT – مقادیر پیش‌فرض را برای ستون‌های کلید خارجی در جدول فرزند تنظیم می‌کند.

RESTRICT – مانند NO ACTION عمل می‌کند.

ON DELETE SET NULL

In [84]:
```sql
%%sql
DROP TABLE IF EXISTS brands;
DROP TABLE IF EXISTS products;

CREATE TABLE brands (
    brand_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR NOT NULL
);

CREATE TABLE products (
  product_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  name VARCHAR NOT NULL,
```

```
  price DECIMAL(10, 2) NOT NULL,
  brand_id INT,
  FOREIGN KEY (brand_id) REFERENCES brands (brand_id)
  ON DELETE SET NULL
);
```

Running query in 'default'

Out[84]:

In [85]:
```
%%sql

INSERT INTO brands(name)
VALUES ('Apple'), ('Samsung')
RETURNING *;
```

Running query in 'default'

2 rows affected.

Out[85]:

brand_id	name
1	Apple
2	Samsung

In [86]:
```
%%sql

INSERT INTO products(name, price, brand_id)
VALUES
('iPhone 14 Pro', 999.99, 1),
('iPhone 15 Pro', 1299.99, 1),
('Galaxy S23 Ultra', 1299.99, 2)
RETURNING *;
```

Running query in 'default'

3 rows affected.

Out[86]:

product_id	name	price	brand_id
1	iPhone 14 Pro	999.99	1
2	iPhone 15 Pro	1299.99	1
3	Galaxy S23 Ultra	1299.99	2

In [87]:
```
%%sql

DELETE FROM brands WHERE brand_id = 1;
```

Running query in 'default'

1 rows affected.

Out[87]:

In [88]:
```
%%sql
SELECT * FROM products;
```

Running query in 'default'

3 rows affected.

Out[88]:

product_id	name	price	brand_id
3	Galaxy S23 Ultra	1299.99	2
1	iPhone 14 Pro	999.99	None
2	iPhone 15 Pro	1299.99	None

حذف جدول با محدودیت کلید خارجی

هنگامی که جدولی را که توسط جداول دیگر از طریق محدودیت‌های کلید خارجی به آن ارجاع می‌شود، حذف می‌کنید، PostgreSQL یک خطا صادر می‌کند.

به عنوان مثال، جدول `products` به جدول `brands` از طریق یک محدودیت کلید خارجی وابسته است. اگر جدول `brands` را حذف کنید، با یک خطا مواجه خواهید شد:

In [89]:
```
%%sql
DROP TABLE brands;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.DependentObjectsStillExist) cannot drop table brands because other objects depend on it
DETAIL:  constraint products_brand_id_fkey on table products depends on table brands
HINT:  Use DROP ... CASCADE to drop the dependent objects too.

[SQL: DROP TABLE brands;]
(Background on this error at: https://sqlalche.me/e/20/2j85)
If you need help solving this issue, send us a message: https://ploomber.io/community
```

در این مواقع از دستور DROP TABLE ... CASCADE استفاده می‌کنیم

In [90]:
```
%%sql
DROP TABLE brands CASCADE;
```

Running query in 'default'

Out[90]:

ON DELETE CASCADE

In [92]:
```
%%sql
DROP TABLE IF EXISTS brands CASCADE;
DROP TABLE IF EXISTS products;

CREATE TABLE brands (
    brand_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR NOT NULL
);
CREATE TABLE products (
  product_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  name VARCHAR NOT NULL,
  price DECIMAL(10, 2) NOT NULL,
  brand_id INT,
  FOREIGN KEY (brand_id) REFERENCES brands (brand_id)
  ON DELETE CASCADE
);
```

Running query in 'default'

Out[92]:

In [93]:
```
%%sql

INSERT INTO brands(name)
VALUES('Apple'), ('Samsung')
RETURNING *;
```

Running query in 'default'

2 rows affected.

Out[93]:

brand_id	name
1	Apple
2	Samsung

In [94]:
```
%%sql
INSERT INTO products(name, price, brand_id)
VALUES
('iPhone 14 Pro', 999.99, 1),
('iPhone 15 Pro', 1299.99, 1),
('Galaxy S23 Ultra', 1299.99, 2)
RETURNING *;
```

Running query in 'default'

3 rows affected.

Out[94]:

product_id	name	price	brand_id
1	iPhone 14 Pro	999.99	1
2	iPhone 15 Pro	1299.99	1
3	Galaxy S23 Ultra	1299.99	2

In [95]:
```
%%sql
DELETE FROM brands
WHERE brand_id = 1;
```

Running query in 'default'

1 rows affected.

Out[95]:

In [96]:
```
%%sql
SELECT * FROM products;
```

Running query in 'default'

1 rows affected.

Out[96]:

product_id	name	price	brand_id
3	Galaxy S23 Ultra	1299.99	2

ON DELETE SET DEFAULT

In [98]:
```
%%sql
DROP TABLE IF EXISTS brands CASCADE;
DROP TABLE IF EXISTS products;

CREATE TABLE brands (
    brand_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR NOT NULL
);
CREATE TABLE products (
  product_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  name VARCHAR NOT NULL,
  price DECIMAL(10, 2) NOT NULL,
  brand_id INT DEFAULT 1,
  FOREIGN KEY (brand_id) REFERENCES brands (brand_id)
    ON DELETE SET DEFAULT
);
```

Running query in 'default'

Out[98]:

In [99]:
```
%%sql
INSERT INTO brands(name)
VALUES('Unknown'),
      ('Apple'),
      ('Samsung')
RETURNING *;
```

Running query in 'default'

3 rows affected.

Out[99]:

brand_id	name
1	Unknown
2	Apple
3	Samsung

In [100…
```
%%sql
INSERT INTO products(name, price, brand_id)
VALUES
('iPhone 14 Pro', 999.99, 2),
('iPhone 15 Pro', 1299.99, 2),
('Galaxy S23 Ultra', 1299.99, 3)
RETURNING *;
```

Running query in 'default'

3 rows affected.

Out[100…

product_id	name	price	brand_id
1	iPhone 14 Pro	999.99	2
2	iPhone 15 Pro	1299.99	2
3	Galaxy S23 Ultra	1299.99	3

In [101…
```
%%sql
DELETE FROM brands
WHERE brand_id = 2;
```

Running query in 'default'

1 rows affected.

Out[101…

In [102…
```
%%sql
SELECT * FROM products;
```

Running query in 'default'

3 rows affected.

Out[102…

product_id	name	price	brand_id
3	Galaxy S23 Ultra	1299.99	3
1	iPhone 14 Pro	999.99	1
2	iPhone 15 Pro	1299.99	1

ON DELETE SET NO ACTION

In [104…

```
%%sql
DROP TABLE IF EXISTS brands CASCADE;
DROP TABLE IF EXISTS products;

CREATE TABLE brands (
    brand_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR NOT NULL
);
CREATE TABLE products (
  product_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  name VARCHAR NOT NULL,
  price DECIMAL(10, 2) NOT NULL,
  brand_id INT,
  FOREIGN KEY (brand_id) REFERENCES brands (brand_id)
);
```

Running query in 'default'

Out[104…

In [105…

```
%%sql

INSERT INTO brands(name)
VALUES('Apple'), ('Samsung')
RETURNING *;
```

Running query in 'default'

2 rows affected.

Out[105…

brand_id	name
1	Apple
2	Samsung

In [106…

```
%%sql

INSERT INTO products(name, price, brand_id)
VALUES
('iPhone 14 Pro', 999.99, 1),
('iPhone 15 Pro', 1299.99, 1),
('Galaxy S23 Ultra', 1299.99, 2)
RETURNING *;
```

Running query in 'default'

3 rows affected.

Out[106…

product_id	name	price	brand_id
1	iPhone 14 Pro	999.99	1
2	iPhone 15 Pro	1299.99	1
3	Galaxy S23 Ultra	1299.99	2

In [107…

```
%%sql
DELETE FROM brands
WHERE brand_id = 1;
```

Running query in 'default'

```
RuntimeError: (psycopg2.errors.ForeignKeyViolation) update or delete on table "brands" violates foreign key constraint "product
s_brand_id_fkey" on table "products"
DETAIL:  Key (brand_id)=(1) is still referenced from table "products".

[SQL: DELETE FROM brands
WHERE brand_id = 1;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
If you need help solving this issue, send us a message: https://ploomber.io/community
```

In [ ]: