نوشتن و خواندن

در *SQLAlchemy* برای درج داده‌ها، یک *session* در نظر گرفته می‌شود. در هر *session*، تغییراتی روی جداول پایگاه داده می‌تواند رخ دهد، اما همه‌ی این کوئری‌ها در یک تراکنش اجرا می‌شوند و تا زمانی که تراکنش *commit* نشود، تغییرات روی پایگاه‌داده اعمال نخواهند شد.

برای تعریف یک *session* از روی یک *engine*، به شیوه‌ی زیر می‌توان عمل کرد:

```
from sqlalchemy.orm import Session

session = Session(engine)
```

در ادامه، می‌توان اشیائی از مدل‌ها ایجاد کرد، آن‌ها را به *session* اضافه کرد و در نهایت، *session* را *commit* کرد.

درج داده‌ها

ایجاد یک نمونه از کلاس Book

برای ایجاد یک کتاب جدید، می‌توان یک نمونه از کلاس `Book` را به شکل زیر تعریف کرد:

```
book = Book(
    title='The Great Gatsby',
    author='F. Scott Fitzgerald',
    price=9.99
)
```

افزودن نمونه به session

برای درج نمونه‌ی ساخته‌شده در جدول `books`، باید از متد `add` روی *session* استفاده کنیم:

```
session.add(book)
```

اعمال تغییرات در پایگاه‌داده

تا کنون تغییرات ما روی سشنی که ایجاد کرده‌ایم ذخیره شده است. برای ذخیره‌ی تغییرات در پایگاه داده، از متد `commit` استفاده می‌شود.

```
session.commit()
```

فرایند ها در کل به این صورت می باشد:

```
from sqlalchemy.orm import Session
from models import Book

session = Session(engine)

book = Book(
    title='The Great Gatsby',
    author='F. Scott Fitzgerald',
    price=9.99
)

session.add(book)

session.commit()
```

خواندن داده ها

تابع اصلی‌ای که برای دریافت داده‌ها استفاده می‌شود، تابع `execute` است. تابع `execute` برای اجرای یک کوئری مشخص (مانند `select`) در پایگاه‌داده استفاده می‌شود. این تابع نتیجه‌ی اجرای کوئری را به‌صورت یک شیء *iterable* برمی‌گرداند که می‌توان مقادیر بازگشتی را از آن استخراج کرد.

دریافت تمام سطرها

برای دریافت اطلاعات تمام کتاب‌های موجود در جدول `books` ، می‌توان از متد `all` استفاده کرد:

```
from sqlalchemy import select

books = session.execute(select(Book)).scalars().all()
```

تابع `select`

تابع `select` برای تعریف یک کوئری جهت انتخاب داده‌ها از پایگاه‌داده استفاده می‌شود. این تابع به شما اجازه می‌دهد ستون‌ها یا جداول موردنظر را برای دریافت داده‌ها مشخص کنید.

متد `scalars`

متد `scalars` برای دریافت مقدار خام سطرها (شیء پایتونی) استفاده می‌شود. در حالت عادی و بدون استفاده از تابع `scalars` ، تاپل‌هایی برگردانده می‌شوند که عضو اول آن‌ها شیء موردنظر است. فرض کنید ۲ کتاب در پایگاه‌داده اضافه کرده‌ایم. تفاوت استفاده و عدم استفاده از `scalars` را ببینید:

```
>>> books = session.execute(select(Book)).all()
>>> print(books)
[(<Book object at 0x7771ba8163c0>,), (<Book object at 0x7771ba9a7610>,)]
>>> books = session.execute(select(Book)).scalars().all()
>>> print(books)
[<Book object at 0x7771ba8163c0>, <Book object at 0x7771ba9a7610>]
```

دریافت سطر خاص با استفاده از شناسه

برای دریافت اطلاعات کتابی با شناسه‌ی *(ID)* مشخص، از متد `get` استفاده می‌کنیم. مثلاً برای دریافت کتابی با شناسه‌ی `10` داریم:

```
book = session.get(Book, 10)
```

فیلتر کردن داده‌ها با شرط

برای دریافت کتاب‌هایی که نویسنده‌ی آن‌ها `J.K. Rowling` است، می‌توان از متد `where` به‌صورت زیر استفاده کرد:

```
stmt = select().where(Book.author == 'J.K. Rowling')
books_by_rowling = session.execute(stmt).scalars().all()
```

همچنین می‌توان چندین شرط را با استفاده از آرگومان‌های متد `where` اعمال کرد. مثلاً، برای دریافت کتاب‌هایی با عنوان `Harry Potter` و قیمت کمتر از `20` :

```
stmt = select(Book).where(Book.title == 'Harry Potter', Book.price < 20)
cheap_harry_potter_books = session.execute(stmt).scalars().all()
```

استفاده از عملگرهای OR و NOT

برای اعمال شرایط پیچیده‌تر، مانند دریافت کتاب‌هایی که عنوانشان `The Hobbit` است یا قیمت آن‌ها بیشتر از `50` نیست، از توابع `or_` و `not_` استفاده می‌کنیم:

```
from sqlalchemy import or_, not_

stmt = select(Book).where(
    (
        Book.title == 'The Hobbit',
        (Book.price > 50)
    )
)
books = session.execute(stmt).scalars().all()
```

دریافت اولین نتیجه

برای دریافت فقط اولین سطر از نتیجه، به‌جای `all` از متد `first` استفاده می‌کنیم. مثلاً، برای دریافت اولین کتابی که نویسنده‌ی آن `George Orwell` است:

```
stmt = select(Book).where(Book.author == 'George Orwell')
book = session.execute(stmt).scalars().first()
```

مرتب‌سازی داده‌ها

برای مرتب‌سازی نتایج، از متد `order_by` استفاده می‌کنیم. مثلاً، برای دریافت کتاب‌ها به‌ترتیب صعودی عنوان و ترتیب نزولی قیمت داریم:

```
stmt = select(Book).order_by(Book.title.asc(), Book.price.desc())
books = session.execute(stmt).scalars().all()
```

اگر بر اساس چند تا ستون بخواهیم مرتب کنیم ترتیب به ترتیب (مهم است) در بخش order_by مینویسیم

محدود کردن تعداد نتایج

برای دریافت تعداد محدودی از نتایج (معادل `LIMIT` در *SQL*)، از متد `limit` استفاده می‌کنیم. مثلاً، برای دریافت ۵ کتاب اول که به ترتیب صعودی قیمت مرتب شده‌اند داریم:

```
stmt = select(Book).order_by(Book.price.asc()).limit(5)
books = session.execute(stmt).scalars().all()
```