SQLAlchemy Async ORM

##

```python
from sqlalchemy.ext.asyncio import create_async_engine, AsyncSession
from sqlalchemy.orm import sessionmaker, declarative_base

DATABASE_URL = "sqlite+aiosqlite:///./mydb.db"
engine = create_async_engine(DATABASE_URL, echo=True)
async_session = sessionmaker(bind=engine, class_=AsyncSession, expire_on_commit=False)
Base = declarative_base()
```

##

```python
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String)
    age = Column(Integer)
    posts = relationship("Post", back_populates="user")

    def to_dict(self):
        return {"id": self.id, "name": self.name, "age": self.age}

class Post(Base):
    __tablename__ = 'posts'
    id = Column(Integer, primary_key=True)
    title = Column(String)
    user_id = Column(Integer, ForeignKey("users.id"))
    user = relationship("User", back_populates="posts")
```

##

```python
async def create_all_tables():
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
```

##

```python
```

```
#
async def add_user(name: str, age: int):
    async with async_session() as session:
        session.add(User(name=name, age=age))
        await session.commit()

#
async def add_many_users(users: list[tuple[str, int]]):
    async with async_session() as session:
        session.add_all([User(name=name, age=age) for name, age in users])
        await session.commit()
```

##

```python
#
result = await session.execute(select(User))
users = result.scalars().all()

#
result = await session.execute(select(User).where(User.name == ""))
user = result.scalars().first()

#   (,  )
result = await session.execute(select(User).where(User.id == 1))
user = result.scalar_one()

#   None
user = result.scalar_one_or_none()
```

##

```python
async def update_user_name(user_id: int, new_name: str):
    async with async_session() as session:
        result = await session.execute(select(User).where(User.id == user_id))
        user = result.scalar_one_or_none()
        if user:
            user.name = new_name
            await session.commit()
```

##

```python
async def delete_user(user_id: int):
    async with async_session() as session:
```

```
        result = await session.execute(select(User).where(User.id == user_id))
        user = result.scalar_one_or_none()
        if user:
            await session.delete(user)
            await session.commit()
```

JOIN

```python
#
async def get_user_posts(user_id: int):
    async with async_session() as session:
        result = await session.execute(select(Post).where(Post.user_id == user_id))
        return result.scalars().all()
```

##

	`.scalars().all()`
	`.scalars().first()`
(None)	`.scalar_one_or_none()`
	`.scalar_one()`
	`result.all()`

`to_dict()`

```python
def to_dict(self):
    return {
        "id": self.id,
        "name": self.name,
        "age": self.age
    }
```

 : ChatGPT