



**Таблица customers**

customer_id	first_name	last_name	phone	country
1	John	Doe	817-646-8833	USA
2	Robert	Luna	412-862-0502	USA
3	David	Robinson	208-340-7906	UK
4	John	Reinhardt	307-242-6285	UK
5	Betty	Taylor	806-749-2958	UAE



## Вступ до ORM

ORM (Object-Relational Mapping) — це спосіб взаємодії з базами даних через об'єктно-орієнтований підхід.

Дозволяє працювати з даними у вигляді класів і об'єктів.

Зменшує потребу в написанні SQL-запитів вручну.



```
pip install sqlalchemy
```

```
pip install psycopg2-binary
```

```
from sqlalchemy import create_engine
```

```
engine = create_engine('sqlite:///example.db')
```



Table — таблиці бази даних.

Column — стовпці таблиці.

Session — сесія взаємодії з базою.

Mapper — відповідність класів Python таблицям.



```
from sqlalchemy import Table, Column, Integer, String, MetaData
```


```
metadata = MetaData()
```

```
users_table = Table(  
    'users',  
    metadata,  
    Column('id', Integer, primary_key=True),  
    Column('name', String(50)),  
    Column('email', String(100), unique=True)  
)
```



## Методи та атрибути Table:

- .columns — отримання всіх стовпців таблиці.
- .primary\_key — отримання первинного ключа.
- .foreign\_keys — отримання зовнішніх ключів.
- .insert() — створення SQL INSERT запиту.
- .update() — створення SQL UPDATE запиту.
- .delete() — створення SQL DELETE запиту.



```
Column('id', Integer, primary_key=True)
Column('name', String(50), nullable=False)
Column('created_at', String, default='CURRENT_TIMESTAMP')
```

- `primary_key=True` — встановлення первинного ключа.
- `nullable=False` — заборона значень NULL.
- `default=<значення>` — встановлення значення за замовчуванням.
- `unique=True` — унікальність стовпця.
- `ForeignKey('<таблиця>.<стовпець>')` — зв'язок із зовнішнім ключем.



```
from sqlalchemy.orm import sessionmaker
from sqlalchemy import create_engine
```

```
engine = create_engine("sqlite:///example.db")
Session = sessionmaker(bind=engine)
session = Session()
```

Основні методи Session:

- `.add(instance)` — додати новий об'єкт у базу.
- `.add_all([instance1, instance2])` — додати список об'єктів.
- `.commit()` — збереження змін.
- `.rollback()` — відкат змін.
- `.query(Model)` — створення SQL SELECT запиту.
- `.delete(instance)` — видалення об'єкта.
- `.close()` — закриття сесії.





```
from sqlalchemy.ext.declarative import declarative_base
```

```
Base = declarative_base()
```

```
class User(Base):
```

```
    __tablename__ = 'users'
```

```
    id = Column(Integer, primary_key=True)
```

```
    name = Column(String(50), nullable=False)
```

```
    email = Column(String(100), unique=True)
```

```
    def __repr__(self):
```

```
        return f"<User(id={self.id}, name={self.name}, email={self.email})>"
```

## Методи моделі (ORM):

.query — дозволяє робити SQL-запити через ORM.


.filter() — додає умови запити.

.all() — повертає всі записи.

.first() — повертає перший запис.

.get(<id>) — отримує запис за первинним ключем.

.delete() — видаляє об'єкт.



```
from sqlalchemy.orm import sessionmaker
Session = sessionmaker(bind=engine)
session = Session()
```

# Додавання


```
new_user = User(name='John Doe', age=30)
session.add(new_user)
session.commit()
```

# Оновлення

```
user = session.query(User).filter_by(name='John Doe').first()
user.age = 31
session.commit()
```

# Видалення

```
session.delete(user)
session.commit()
```



# Вибірка всіх користувачів  
`users = session.query(User).all()`

# Фільтрація  
`young_users = session.query(User).filter(User.age < 30).all()`

# Сортювання  
`sorted_users = session.query(User).order_by(User.age).all()`



```
from sqlalchemy import ForeignKey
from sqlalchemy.orm import relationship
```

```
class Department(Base):
    __tablename__ = 'departments'
    id = Column(Integer, primary_key=True)
    name = Column(String)
    employees = relationship("Employee", back_populates="department")
```

```
class Employee(Base):
    __tablename__ = 'employees'
    id = Column(Integer, primary_key=True)
    name = Column(String)
    department_id = Column(Integer, ForeignKey('departments.id'))
    department = relationship("Department", back_populates="employees")
```




```
from pony.orm import Database
```

```
db = Database(provider='sqlite', filename='example.db', create_db=True)
```

- Entity — таблиці бази даних.
- PrimaryKey — первинний ключ.
- Required — обов'язкові поля.

```
from pony.orm import Required
```

```
class User(db.Entity):  
    name = Required(str)  
    age = Required(int)
```



```
db = Database()
class User(db.Entity):
    name = Required(str)
    email = Required(str, unique=True)
    age = Optional(int)
db.bind(provider='sqlite', filename=':memory:', create_db=True)
db.generate_mapping(create_tables=True)
```



```
with db_session:
```

```
user1 = User(name="John Doe", email="john@example.com", age=30)
```

```
user2 = User.create(name="Jane Doe", email="jane@example.com", age=25)
```

```
with db_session:
```

```
    users = select(u for u in User)[:]
```

```
    print("Усі користувачі:", [(u.name, u.email, u.age) for u in users])
```

```
    user = User.get(email="john@example.com")
```


```
    if user:
```

```
        print(f"Знайдено: {user.name}, {user.email}, {user.age}")
```

```
    young_users = User.select(lambda u: u.age < 30)[:]
```

```
    print("Молоді користувачі:", [u.name for u in young_users])
```





```
with db_session:
    user = User.get(email="john@example.com")
    if user:
        user.age = 35
        print(f"Оновлено: {user.name}, {user.email}, {user.age}")

    User.select(lambda u: u.age < 30).update(age=30)
```

```
with db_session:
    user = User.get(email="john@example.com")
    if user:
        user.delete()
        print(f"Користувач {user.email} видалений")

    User.select(lambda u: u.age > 50).delete()
```



```
from pony.orm import Database
```

```
db = Database()
```

```
db.bind(provider='postgres', user='user', password='password',  
host='localhost', database='dbname')
```

```
for table_name in db.get_tables():
```

```
    print(f"Таблиця: {table_name}")
```

```
    primary_keys = [col.name for col in db.get_columns(table_name) if  
col.is_pk]
```


```
    print(f"Первинний ключ: {primary_keys}")
```

```
for entity in db.entities.values():
```

```
    for attr in entity._attrs_:
```

```
        if attr.is_relation and attr.reverse:
```

```
            print(f"Таблиця {entity._table_}: Зовнішній ключ {attr.name} ->  
{attr.reverse.entity._table_}({attr.reverse.name})")
```



```
from sqlalchemy import create_engine, MetaData
engine = create_engine("postgresql://user:password@localhost/dbname")
metadata = MetaData()
metadata.reflect(bind=engine)
```

```
# Отримуємо первинні ключі
for table_name, table in metadata.tables.items():
    print(f"Таблиця: {table_name}")
    print(f"Первинний ключ: {[col.name for col in table.primary_key]}")
```

```
# Отримуємо зовнішні ключі
for table_name, table in metadata.tables.items():
    for col in table.columns:
        for fk in col.foreign_keys:
            sprint(f"Таблиця {table_name}: Зовнішній ключ {col.name} ->
{fk.column.table.name}({fk.column.name})")
```