| SQL Truy vấn | SQLAlchemy ORM (FastAPI) | Mô tả ngắn |
| --- | --- | --- |
| **CREATE TABLE** (không trực tiếp, dùng migration) | Base.metadata.create\_all(engine) hoặc dùng Alembic cho migration | Tạo bảng từ model definitions (e.g., class Blog(Base): ...) |
| **DROP TABLE** (không trực tiếp) | Base.metadata.drop\_all(engine) | Xóa bảng từ model definitions |
| **INSERT INTO blog (title, body) VALUES ('Title', 'Body');** | new\_blog = models.Blog(title='Title', body='Body'); db.add(new\_blog); db.commit(); db.refresh(new\_blog) | Thêm record mới và refresh để lấy ID mới |
| **INSERT INTO blog (title, body) VALUES ... (multiple rows)** | db.bulk\_insert\_mappings(models.Blog, [{'title': 'T1', 'body': 'B1'}, {'title': 'T2', 'body': 'B2'}]); db.commit() | Thêm nhiều record (bulk insert) |
| **SELECT \* FROM blog;** | db.query(models.Blog).all() | Lấy tất cả records |
| **SELECT \* FROM blog WHERE id=1;** | db.query(models.Blog).filter(models.Blog.id == 1).first() | Lấy một record hoặc None |
| **SELECT \* FROM blog WHERE title='Specific Title';** | db.query(models.Blog).filter\_by(title='Specific Title').all() | Filter đơn giản bằng keyword args (filter\_by) |
| **SELECT \* FROM blog WHERE id > 5 AND title LIKE '%A%';** | db.query(models.Blog).filter(models.Blog.id > 5, models.Blog.title.like('%A%')).all() | Filter phức tạp với AND (implicit) |
| **SELECT \* FROM blog WHERE id=1 OR title='Title';** | from sqlalchemy import or\_; db.query(models.Blog).filter(or\_(models.Blog.id == 1, models.Blog.title == 'Title')).all() | Filter với OR |
| **SELECT \* FROM blog WHERE id IN (1,2,3);** | db.query(models.Blog).filter(models.Blog.id.in\_([1,2,3])).all() | Filter với IN |
| **SELECT \* FROM blog WHERE id BETWEEN 5 AND 10;** | db.query(models.Blog).filter(models.Blog.id.between(5,10)).all() | Filter với BETWEEN |
| **SELECT \* FROM blog WHERE title IS NULL;** | db.query(models.Blog).filter(models.Blog.title.is\_(None)).all() | Filter với IS NULL |
| **SELECT \* FROM blog WHERE title IS NOT NULL;** | db.query(models.Blog).filter(models.Blog.title.isnot(None)).all() | Filter với IS NOT NULL |
| **SELECT COUNT(\*) FROM blog;** | db.query(models.Blog).count() | Đếm tổng records |
| **SELECT SUM(views) FROM blog;** | from sqlalchemy import func; db.query(func.sum(models.Blog.views)).scalar() | Tính tổng (aggregate SUM) |
| **SELECT AVG(views) FROM blog;** | db.query(func.avg(models.Blog.views)).scalar() | Tính trung bình (AVG) |
| **SELECT MAX(views) FROM blog;** | db.query(func.max(models.Blog.views)).scalar() | Tìm giá trị lớn nhất (MAX) |
| **SELECT MIN(views) FROM blog;** | db.query(func.min(models.Blog.views)).scalar() | Tìm giá trị nhỏ nhất (MIN) |
| **SELECT \* FROM blog ORDER BY views DESC;** | from sqlalchemy import desc; db.query(models.Blog).order\_by(desc(models.Blog.views)).all() | Sắp xếp giảm dần (ORDER BY DESC) |
| **SELECT \* FROM blog ORDER BY views ASC;** | db.query(models.Blog).order\_by(models.Blog.views).all() | Sắp xếp tăng dần (ORDER BY ASC, default) |
| **SELECT \* FROM blog LIMIT 10 OFFSET 20;** | db.query(models.Blog).offset(20).limit(10).all() | Phân trang (LIMIT và OFFSET) |
| **SELECT DISTINCT title FROM blog;** | db.query(models.Blog.title).distinct().all() | Lấy giá trị unique (DISTINCT) |
| **UPDATE blog SET title='New Title' WHERE id=1;** | blog = db.query(models.Blog).filter(models.Blog.id == 1).first(); blog.title = 'New Title'; db.commit() | Cập nhật single object |
| **UPDATE blog SET title='New' WHERE id=1; (batch)** | db.query(models.Blog).filter(models.Blog.id == 1).update({'title': 'New'}); db.commit() | Cập nhật batch |
| **DELETE FROM blog WHERE id=1;** | blog = db.query(models.Blog).filter(models.Blog.id == 1).first(); db.delete(blog); db.commit() | Xóa single object |
| **DELETE FROM blog WHERE id=1; (batch)** | db.query(models.Blog).filter(models.Blog.id == 1).delete(); db.commit() | Xóa batch |
| **SELECT \* FROM blog INNER JOIN author ON blog.author\_id = author.id;** | db.query(models.Blog, models.Author).join(models.Author, models.Blog.author\_id == models.Author.id).all() | Inner join hai bảng |
| **SELECT \* FROM blog LEFT JOIN author ON blog.author\_id = author.id;** | db.query(models.Blog, models.Author).outerjoin(models.Author, models.Blog.author\_id == models.Author.id).all() | Left join (outerjoin) |
| **SELECT \* FROM blog WHERE EXISTS (SELECT 1 FROM author WHERE author.id = blog.author\_id);** | subq = db.query(models.Author).filter(models.Author.id == models.Blog.author\_id).exists(); db.query(models.Blog).filter(subq).all() | Filter với EXISTS subquery |
| **SELECT \* FROM blog GROUP BY author\_id;** | db.query(models.Blog.author\_id).group\_by(models.Blog.author\_id).all() | Group by một cột |
| **SELECT author\_id, COUNT(*) FROM blog GROUP BY author\_id HAVING COUNT(*) > 1;** | db.query(models.Blog.author\_id, func.count()).group\_by(models.Blog.author\_id).having(func.count() > 1).all() | Group by với HAVING |
| **BEGIN TRANSACTION; ... COMMIT;** | with db.begin(): ... hoặc db.commit() | Quản lý transaction (commit thay đổi) |
| **BEGIN TRANSACTION; ... ROLLBACK;** | db.rollback() | Hủy transaction khi lỗi |
| **SELECT \* FROM blog FOR UPDATE;** | db.query(models.Blog).with\_for\_update().all() | Lock rows cho update (pessimistic locking) |
| **—** (Debug SQL) | print(str(query.statement.compile(compile\_kwargs={"literal\_binds": True}))) | Hiển thị câu SQL thực thi để debug |
| **ALTER TABLE blog ADD COLUMN new\_col VARCHAR(50);** (không trực tiếp) | Thay đổi model class và dùng Alembic migration | Thêm cột mới (qua migration tool) |
| **CREATE INDEX idx\_title ON blog(title);** (không trực tiếp) | Trong model: title = Column(String, index=True) rồi migrate | Tạo index trên cột |

**Ghi chú:**

* Giả sử db là session từ FastAPI dependency (e.g., db: Session = Depends(get\_db)).
* Model ví dụ: class Blog(Base): \_\_tablename\_\_ = 'blog'; id = Column(Integer, primary\_key=True); title = Column(String); body = Column(String); author\_id = Column(Integer, ForeignKey('author.id')); author = relationship('Author', back\_populates='blogs') (và tương tự cho Author).
* Với relationships: Truy cập như blog.author thay vì join thủ công nếu đã define relationship.
* Với SQL Server: Sử dụng mssql+pyodbc dialect; một số syntax như LIMIT/OFFSET có thể dùng TOP/OFFSET FETCH, nhưng SQLAlchemy xử lý tự động.
* Để bulk operations hiệu quả hơn, dùng db.bulk\_save\_objects([obj1, obj2]).
* Luôn commit sau thay đổi, và xử lý exception với rollback.