In [ ]:
```
CREATE VIEW 뷰이름 AS
SQL문 -> JOIN
프랜차이즈 -> 지점별 매출현황 -> JOIN(CITY, SUPPLIER, SELLS, PART)
```

In [ ]:
```
!pip install sqlalchemy
```

In [1]:
```
from sqlalchemy import create_engine
from sqlalchemy.schema import MetaData, Table, Column, ForeignKey
from sqlalchemy.types import Integer, String
```

In [4]:
```
# starting point
engine = create_engine('sqlite:///orm1.db', echo=True)
```

In [6]:
```
meta = MetaData() # bing 옵션 = engine 연결
```

In [8]:
```
meta.tables
```

Out[8]:
```
FacadeDict({})
```

In [9]:
```
Table('T_USER', meta,
      Column('PK', Integer, primary_key=True),
      Column('NAME', String, nullable=False))
```

Out[9]:
```
Table('T_USER', MetaData(), Column('PK', Integer(), table=<T_USER>, primary_key=Tr
ue, nullable=False), Column('NAME', String(), table=<T_USER>, nullable=False), sch
ema=None)
```

In [ ]:
```
# 테이블 객체가 하나 생성 -> meta 등록, 아직은 물리적 DB 반영 X
```

In [12]:
```
meta.bind = engine
```

In [13]:
```
meta.create_all()
```

```
2023-03-07 09:41:54,221 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-07 09:41:54,226 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_US
ER")
2023-03-07 09:41:54,227 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 09:41:54,229 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("T_US
ER")
2023-03-07 09:41:54,229 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 09:41:54,231 INFO sqlalchemy.engine.Engine
CREATE TABLE "T_USER" (
        "PK" INTEGER NOT NULL,
        "NAME" VARCHAR NOT NULL,
        PRIMARY KEY ("PK")
)


2023-03-07 09:41:54,232 INFO sqlalchemy.engine.Engine [no key 0.00047s] ()
2023-03-07 09:41:54,234 INFO sqlalchemy.engine.Engine COMMIT
```

In [18]:
```
from sqlalchemy.sql import select, insert
```

In [21]:
```
print(select(meta.tables['T_USER']))
```

```
SELECT "T_USER"."PK", "T_USER"."NAME"
FROM "T_USER"
```

In [22]:
```
print(meta.tables['T_USER'].insert())
```
```
INSERT INTO "T_USER" ("PK", "NAME") VALUES (?, ?)
```

In [23]:
```
User = meta.tables['T_USER']
```

In [24]:
```
print(insert(User), select(User), User.update())
```
```
INSERT INTO "T_USER" ("PK", "NAME") VALUES (?, ?) SELECT "T_USER"."PK", "T_USE
R"."NAME"
FROM "T_USER" UPDATE "T_USER" SET "PK"=?, "NAME"=?
```

In [27]:
```
print(meta.tables['T_USER'].insert().values(NAME='아무나'))
```
```
INSERT INTO "T_USER" ("NAME") VALUES (?)
```

In [34]:
```
print(User.insert().values(NAME='아무나').compile())
```
```
INSERT INTO "T_USER" ("NAME") VALUES (?)
```

In [33]:
```
print(User.insert().values(NAME='아무나').compile().params)
```
```
{'NAME': '아무나'}
```

In [36]:
```
con = engine.connect()
```

In [35]:
```
insert(User, bind=engine).values(NAME='아무나')
```

Out[35]:
```
<sqlalchemy.sql.dml.Insert object at 0x10d8606a0>
```

In [37]:
```
cur = con.execute(insert(User).values(NAME='아무나'))
```
```
2023-03-07 09:54:25,430 INFO sqlalchemy.engine.Engine INSERT INTO "T_USER" ("NAM
E") VALUES (?)
2023-03-07 09:54:25,441 INFO sqlalchemy.engine.Engine [generated in 0.01333s] ('아
무나',)
2023-03-07 09:54:25,445 INFO sqlalchemy.engine.Engine COMMIT
```

In [38]:
```
cur = con.execute(User.select())
```
```
2023-03-07 09:55:29,041 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK", "T_USE
R"."NAME"
FROM "T_USER"
2023-03-07 09:55:29,045 INFO sqlalchemy.engine.Engine [generated in 0.00449s] ()
```

In [39]:
```
cur.fetchall()
```

Out[39]:
```
[(1, '아무나')]
```

In [44]:
```
from sqlalchemy.sql import and_, or_, between
```

In [46]:
```
print(and_(User.c.PK == 1, User.c.NAME == 1))
```
```
"T_USER"."PK" = :PK_1 AND "T_USER"."NAME" = :NAME_1
```

In [47]:
```
Table('T_ADDRESS', meta,
      Column('PK', Integer, primary_key=True),
      Column('ADDR', String),
      Column('FK', Integer, nullable=False))
```

```
Out[47]: Table('T_ADDRESS', MetaData(bind=Engine(sqlite:///orm1.db)), Column('PK', Integer
         (), table=<T_ADDRESS>, primary_key=True, nullable=False), Column('ADDR', String(),
         table=<T_ADDRESS>), Column('FK', Integer(), table=<T_ADDRESS>, nullable=False), sc
         hema=None)
```

```
In [50]: len(meta.tables), meta.tables
```

```
Out[50]: (2,
          FacadeDict({'T_USER': Table('T_USER', MetaData(bind=Engine(sqlite:///orm1.db)), C
         olumn('PK', Integer(), table=<T_USER>, primary_key=True, nullable=False), Column
         ('NAME', String(), table=<T_USER>, nullable=False), schema=None), 'T_ADDRESS': Tab
         le('T_ADDRESS', MetaData(bind=Engine(sqlite:///orm1.db)), Column('PK', Integer(),
         table=<T_ADDRESS>, primary_key=True, nullable=False), Column('ADDR', String(), tab
         le=<T_ADDRESS>), Column('FK', Integer(), table=<T_ADDRESS>, nullable=False), schem
         a=None)}))
```

```
In [51]: meta.bind
```

```
Out[51]: Engine(sqlite:///orm1.db)
```

```
In [52]: meta.create_all()
```

```
2023-03-07 10:02:32,779 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-07 10:02:32,783 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_US
ER")
2023-03-07 10:02:32,784 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 10:02:32,786 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_AD
DRESS")
2023-03-07 10:02:32,786 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 10:02:32,787 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("T_AD
DRESS")
2023-03-07 10:02:32,788 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 10:02:32,790 INFO sqlalchemy.engine.Engine
CREATE TABLE "T_ADDRESS" (
        "PK" INTEGER NOT NULL,
        "ADDR" VARCHAR,
        "FK" INTEGER NOT NULL,
        PRIMARY KEY ("PK")
)


2023-03-07 10:02:32,794 INFO sqlalchemy.engine.Engine [no key 0.00386s] ()
2023-03-07 10:02:32,796 INFO sqlalchemy.engine.Engine COMMIT
```

```
In [56]: Addr = meta.tables['T_ADDRESS']
```

```
In [57]: type(User), type(Addr)
```

```
Out[57]: (sqlalchemy.sql.schema.Table, sqlalchemy.sql.schema.Table)
```

```
In [59]: print(select([User, Addr]))
```

```
SELECT "T_USER"."PK", "T_USER"."NAME", "T_ADDRESS"."PK" AS "PK_1", "T_ADDRESS"."AD
DR", "T_ADDRESS"."FK"
FROM "T_USER", "T_ADDRESS"
```

```
In [60]: from sqlalchemy.sql import join
```

```
In [62]: print(join(User, Addr, User.c.PK==Addr.c.FK))
```

```
"T_USER" JOIN "T_ADDRESS" ON "T_USER"."PK" = "T_ADDRESS"."FK"
```

In [69]:
```
print(select([User.c.NAME, Addr.c.ADDR])\
      .select_from(join(User, Addr, User.c.PK==Addr.c.FK)))
```

```
SELECT "T_USER"."NAME", "T_ADDRESS"."ADDR"
FROM "T_USER" JOIN "T_ADDRESS" ON "T_USER"."PK" = "T_ADDRESS"."FK"
```

In [70]:
```
sql = select([User.c.NAME, Addr.c.ADDR])\
      .select_from(join(User, Addr, User.c.PK==Addr.c.FK))
cur = con.execute(sql)
cur.fetchall()
```

```
2023-03-07 10:47:40,233 INFO sqlalchemy.engine.Engine SELECT "T_USER"."NAME", "T_A
DDRESS"."ADDR"
FROM "T_USER" JOIN "T_ADDRESS" ON "T_USER"."PK" = "T_ADDRESS"."FK"
2023-03-07 10:47:40,235 INFO sqlalchemy.engine.Engine [generated in 0.00204s] ()
```

Out[70]:
```
[]
```

In [73]:
```
print(select([User.c.NAME, Addr.c.ADDR])\
      .select_from(join(User, Addr, User.c.PK==Addr.c.FK))\
      .where(User.c.PK == 1))
```

```
SELECT "T_USER"."NAME", "T_ADDRESS"."ADDR"
FROM "T_USER" JOIN "T_ADDRESS" ON "T_USER"."PK" = "T_ADDRESS"."FK"
WHERE "T_USER"."PK" = ?
```

In [89]:
```
# print(User.select().where(User.c.PK==1))
# print(select(User))
FK = con.execute(User.select().where(User.c.PK==1)).fetchone()[0]
                 # 1. User 객체 - SELECT - WHERE => SQL문
                 # 2. engine의 connection pool conntion -> con.execute
                 # 3. 2.의 결과가 resultproxy -> cursor
                 # 4. 3.의 결과(DB의 실제 데이터)를 fetchone(1행)[0 => 1열]

cur = con.execute(insert(Addr).values(ADDR='아무거나', FK=FK))
cur.lastrowid
```

```
2023-03-07 10:53:39,133 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK", "T_USE
R"."NAME"
FROM "T_USER"
WHERE "T_USER"."PK" = ?
2023-03-07 10:53:39,142 INFO sqlalchemy.engine.Engine [cached since 41.77s ago]
(1,)
2023-03-07 10:53:39,143 INFO sqlalchemy.engine.Engine INSERT INTO "T_ADDRESS" ("AD
DR", "FK") VALUES (?, ?)
2023-03-07 10:53:39,143 INFO sqlalchemy.engine.Engine [generated in 0.00054s] ('아
무거나', 1)
2023-03-07 10:53:39,144 INFO sqlalchemy.engine.Engine COMMIT
```

Out[89]:
```
1
```

In [96]:
```
con.close()
meta.clear()
engine.dispose()
```

In [97]:
```
engine = create_engine('sqlite:///orm1.db', echo=True)
meta = MetaData(engine)
con = engine.connect()
```

In [98]:
```
meta.create_all()
```

```
2023-03-07 10:59:28,128 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-07 10:59:28,141 INFO sqlalchemy.engine.Engine COMMIT
```

In [99]:
```
meta.tables
```

Out[99]:
```
FacadeDict({})
```

In [100…
```
Table('T_USER', meta,
      Column('PK', Integer))
```

Out[100]:
```
Table('T_USER', MetaData(bind=Engine(sqlite:///orm1.db)), Column('PK', Integer(),
table=<T_USER>), schema=None)
```

In [101…
```
meta.create_all()
```

```
2023-03-07 10:59:43,426 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-07 10:59:43,435 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_US
ER")
2023-03-07 10:59:43,438 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 10:59:43,440 INFO sqlalchemy.engine.Engine COMMIT
```

In [102…
```
meta.tables
```

Out[102]:
```
FacadeDict({'T_USER': Table('T_USER', MetaData(bind=Engine(sqlite:///orm1.db)), Co
lumn('PK', Integer(), table=<T_USER>), schema=None)})
```

In [105…
```
con.execute(meta.tables['T_USER'].select()).fetchall()
```

```
2023-03-07 11:00:58,515 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK"
FROM "T_USER"
2023-03-07 11:00:58,522 INFO sqlalchemy.engine.Engine [generated in 0.00680s] ()
```

Out[105]:
```
[(1,)]
```

In [103…
```
User
```

Out[103]:
```
Table('T_USER', MetaData(bind=Engine(sqlite:///orm1.db)), Column('PK', Integer(),
table=<T_USER>, primary_key=True, nullable=False), Column('NAME', String(), table=
<T_USER>, nullable=False), schema=None)
```

In [109…
```
Table('T_USER', meta,
      Column('PK', Integer),
      Column('TEST', String),
      extend_existing=True)
```

Out[109]:
```
Table('T_USER', MetaData(bind=Engine(sqlite:///orm1.db)), Column('PK', Integer(),
table=<T_USER>), Column('TEST', String(), table=<T_USER>), schema=None)
```

In [128…
```
물리적DB <-------> in memory Class-Instance
         ORM-Core
                    MetaData
engine-dialect
T_USER                T_USER*(없는 Column)
```

```
2023-03-07 11:07:15,281 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-07 11:07:15,284 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_US
ER")
2023-03-07 11:07:15,284 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 11:07:15,286 INFO sqlalchemy.engine.Engine COMMIT
```

In [147…
```
con.close()
meta.clear()
```

```
engine.dispose()
```

In [144…

```
# 프랜차이즈
engine = create_engine('sqlite:///test1.db', echo=True)
meta = MetaData(engine)
con = engine.connect()
```

In [136…

```
# city, part, sells, supplier
# pk, name    fk1, fk2, price,   fk
Table('city', meta,
      Column('pk', Integer, primary_key=True),
      Column('name', String), extend_existing=True)
Table('part', meta,
      Column('pk', Integer, primary_key=True),
      Column('name', String), extend_existing=True)
Table('supplier', meta,
      Column('pk', Integer, primary_key=True),
      Column('name', String),
      Column('fk', Integer), extend_existing=True)
Table('sells', meta,
      Column('fk1', Integer),
      Column('fk2', Integer),
      Column('price', Integer), extend_existing=True)
```

Out[136]:

```
Table('sells', MetaData(bind=Engine(sqlite:///test1.db)), Column('fk1', Integer(),
table=<sells>), Column('fk2', Integer(), table=<sells>), Column('price', Integer
(), table=<sells>), schema=None)
```

In [137…

```
meta.create_all()
```

```
2023-03-07 11:13:35,875 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-07 11:13:35,886 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("pk")
2023-03-07 11:13:35,892 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 11:13:35,900 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("par
t")
2023-03-07 11:13:35,901 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 11:13:35,902 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("supp
lier")
2023-03-07 11:13:35,902 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 11:13:35,903 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("sell
s")
2023-03-07 11:13:35,903 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 11:13:35,904 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("cit
y")
2023-03-07 11:13:35,904 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 11:13:35,905 INFO sqlalchemy.engine.Engine COMMIT
```

In [138…

```
con.execute(select(meta.tables['sells'])).fetchall()
```

```
2023-03-07 11:14:00,624 INFO sqlalchemy.engine.Engine SELECT sells.fk1, sells.fk2,
sells.price
FROM sells
2023-03-07 11:14:00,626 INFO sqlalchemy.engine.Engine [generated in 0.00222s] ()
```

Out[138]:

```
[]
```

In [139…

```
city = meta.tables['city']
supplier = meta.tables['supplier']
part = meta.tables['part']
sells = meta.tables['sells']
```

In [140…
```
sql = select([city.c.name, supplier.c.name])\
.select_from(join(city, supplier, city.c.pk == supplier.c.fk))
con.execute(sql).fetchall()
```

```
2023-03-07 11:16:27,150 INFO sqlalchemy.engine.Engine SELECT city.name, supplier.n
ame AS name_1
FROM city JOIN supplier ON city.pk = supplier.fk
2023-03-07 11:16:27,154 INFO sqlalchemy.engine.Engine [generated in 0.00530s] ()
```

Out[140]:
```
[('성북구', '안암1호점'), ('성북구', '안암2호점'), ('성북구', '종암1호점')]
```

In [ ]:
```
supplier - sells - part 3개 조인
city - supplier - sells - part 4개 조인
select할 columns 지정도 해보고
where 조건도 표현해보고,
groupby, orderby => 할수있음 해보고(PPT)
버전 2인 사람들은 select에 리스트 빼고 나열형으로 값 전달!
```

In [141…
```
import sqlalchemy
```

In [142…
```
sqlalchemy.__version__
```

Out[142]:
```
'1.4.41'
```

In [148…
```
# 프랜차이즈
engine = create_engine('sqlite:///playlist.db', echo=True)
meta = MetaData(engine)
con = engine.connect()
```

In [149…
```
meta.reflect(engine)
```

```
2023-03-07 12:02:27,612 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_mast
er WHERE type='table' ORDER BY name
2023-03-07 12:02:27,615 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,616 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("alb
um")
2023-03-07 12:02:27,617 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,618 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,619 INFO sqlalchemy.engine.Engine [raw sql] ('album',)
2023-03-07 12:02:27,620 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("album")
2023-03-07 12:02:27,620 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,621 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("album")
2023-03-07 12:02:27,621 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,622 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,622 INFO sqlalchemy.engine.Engine [raw sql] ('album',)
2023-03-07 12:02:27,623 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("albu
m")
2023-03-07 12:02:27,623 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,624 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("albu
m")
2023-03-07 12:02:27,624 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,624 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("albu
m")
2023-03-07 12:02:27,625 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,625 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("albu
m")
2023-03-07 12:02:27,625 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,626 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,626 INFO sqlalchemy.engine.Engine [raw sql] ('album',)
2023-03-07 12:02:27,627 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("art
ist")
2023-03-07 12:02:27,627 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,628 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,628 INFO sqlalchemy.engine.Engine [raw sql] ('artist',)
2023-03-07 12:02:27,629 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("artist")
2023-03-07 12:02:27,629 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,629 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("artist")
2023-03-07 12:02:27,630 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,630 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,630 INFO sqlalchemy.engine.Engine [raw sql] ('artist',)
2023-03-07 12:02:27,631 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("arti
st")
2023-03-07 12:02:27,631 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,632 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("arti
st")
2023-03-07 12:02:27,632 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,633 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("arti
st")
```

```
2023-03-07 12:02:27,633 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,633 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("arti
st")
2023-03-07 12:02:27,634 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,634 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,634 INFO sqlalchemy.engine.Engine [raw sql] ('artist',)
2023-03-07 12:02:27,635 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("gen
re")
2023-03-07 12:02:27,635 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,636 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,636 INFO sqlalchemy.engine.Engine [raw sql] ('genre',)
2023-03-07 12:02:27,637 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("genre")
2023-03-07 12:02:27,637 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,637 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("genre")
2023-03-07 12:02:27,638 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,638 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,638 INFO sqlalchemy.engine.Engine [raw sql] ('genre',)
2023-03-07 12:02:27,639 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("genr
e")
2023-03-07 12:02:27,639 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,639 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("genr
e")
2023-03-07 12:02:27,640 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,640 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("genr
e")
2023-03-07 12:02:27,640 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,640 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("genr
e")
2023-03-07 12:02:27,641 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,641 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,641 INFO sqlalchemy.engine.Engine [raw sql] ('genre',)
2023-03-07 12:02:27,642 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("tra
ck")
2023-03-07 12:02:27,642 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,643 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,643 INFO sqlalchemy.engine.Engine [raw sql] ('track',)
2023-03-07 12:02:27,643 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("track")
2023-03-07 12:02:27,644 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,644 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("track")
2023-03-07 12:02:27,644 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,645 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,645 INFO sqlalchemy.engine.Engine [raw sql] ('track',)
2023-03-07 12:02:27,645 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("trac
k")
2023-03-07 12:02:27,646 INFO sqlalchemy.engine.Engine [raw sql] ()
```

```
2023-03-07 12:02:27,646 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("trac
k")
2023-03-07 12:02:27,646 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,647 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("trac
k")
2023-03-07 12:02:27,647 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,647 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("trac
k")
2023-03-07 12:02:27,647 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-07 12:02:27,648 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-07 12:02:27,649 INFO sqlalchemy.engine.Engine [raw sql] ('track',)
```

In [150…
```
meta.tables.keys()
```

Out[150]:
```
dict_keys(['album', 'artist', 'genre', 'track'])
```

In [151…
```
album = meta.tables['album']
artist = meta.tables['artist']
```

In [155…
```
# 버전 1대
sql = select([album.c.name, artist.c.name])\
.select_from(join(album, artist, album.c.fk==artist.c.pk))
# 버전 2대
# select(album.c.name, artist.c.name)
```

In [168…
```
con.execute(sql).fetchall()
```

```
2023-03-07 12:09:16,801 INFO sqlalchemy.engine.Engine SELECT album.name, artist.na
me AS name_1
FROM album JOIN artist ON album.fk = artist.pk
2023-03-07 12:09:16,811 INFO sqlalchemy.engine.Engine [cached since 268.7s ago] ()
```

Out[168]:
```
[('아무개', '가수1'), ('아무개1', '가수1'), ('아무개2', '가수2')]
```

In [159…
```
album.columns.keys(), artist.columns.keys()
```

Out[159]:
```
(['pk', 'name', 'fk'], ['pk', 'name'])
```

In [167…
```
con.execute(album.select()).fetchall()
```

```
2023-03-07 12:09:12,559 INFO sqlalchemy.engine.Engine SELECT album.pk, album.name,
album.fk
FROM album
2023-03-07 12:09:12,563 INFO sqlalchemy.engine.Engine [cached since 218.7s ago] ()
```

Out[167]:
```
[(1, '아무개', 1), (2, '아무개1', 1), (3, '아무개2', 2)]
```

In [165…
```
con.execute(insert(album), [{'name':'아무개1', 'fk':1},
                            {'name':'아무개2', 'fk':2}])
```

```
2023-03-07 12:08:47,682 INFO sqlalchemy.engine.Engine INSERT INTO album (name, fk)
VALUES (?, ?)
2023-03-07 12:08:47,690 INFO sqlalchemy.engine.Engine [generated in 0.00880s]
(('아무개1', 1), ('아무개2', 2))
2023-03-07 12:08:47,694 INFO sqlalchemy.engine.Engine COMMIT
```

Out[165]:
```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x1199dcfa0>
```

In [166…
```
con.execute(insert(artist), [{'name':'가수1'},
                             {'name':'가수2'}])
```
```
2023-03-07 12:09:06,817 INFO sqlalchemy.engine.Engine INSERT INTO artist (name) VA
LUES (?)
2023-03-07 12:09:06,821 INFO sqlalchemy.engine.Engine [generated in 0.00379s]
(('가수1',), ('가수2',))
2023-03-07 12:09:06,826 INFO sqlalchemy.engine.Engine COMMIT
```
Out[166]:
```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x1198dc460>
```

In [169…
```
# 버전 1대
sql = select([album.c.name, artist.c.name])\
.select_from(join(album, artist, album.c.fk==artist.c.pk))
# 버전 2대
# select(album.c.name, artist.c.name)
con.execute(sql).fetchall()
```
```
2023-03-07 12:10:05,211 INFO sqlalchemy.engine.Engine SELECT album.name, artist.na
me AS name_1
FROM album JOIN artist ON album.fk = artist.pk
2023-03-07 12:10:05,214 INFO sqlalchemy.engine.Engine [cached since 317.1s ago] ()
```
Out[169]:
```
[('아무개', '가수1'), ('아무개1', '가수1'), ('아무개2', '가수2')]
```

In [170…
```
print(insert(artist))
```
```
INSERT INTO artist (pk, name) VALUES (?, ?)
```

In [172…
```
print(insert(artist, {'name':'hjahaha'})),\
print(insert(artist, {'name':'hjahaha'}).compile().params)
```
```
INSERT INTO artist (name) VALUES (?)
{'name': 'hjahaha'}
```
Out[172]:
```
(None, None)
```

In [179…
```
con.execute(artist.update().values(name='가수10').where(artist.c.pk == 1))
```
```
2023-03-07 12:17:34,263 INFO sqlalchemy.engine.Engine UPDATE artist SET name=? WHE
RE artist.pk = ?
2023-03-07 12:17:34,264 INFO sqlalchemy.engine.Engine [cached since 29.16s ago]
('가수10', 1)
2023-03-07 12:17:34,266 INFO sqlalchemy.engine.Engine COMMIT
```
Out[179]:
```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x1199c7a60>
```

In [178…
```
print(artist.update().values(name='가수10').where(artist.c.pk == 1))
```
```
UPDATE artist SET name=? WHERE artist.pk = ?
```

In [180…
```
con.execute(artist.select()).fetchall()
```
```
2023-03-07 12:17:49,538 INFO sqlalchemy.engine.Engine SELECT artist.pk, artist.nam
e
FROM artist
2023-03-07 12:17:49,544 INFO sqlalchemy.engine.Engine [generated in 0.00665s] ()
```
Out[180]:
```
[(1, '가수10'), (2, '가수1'), (3, '가수1'), (4, '가수1'), (5, '가수1'), (6, '가수
1')]
```

In [183…
```
print(artist.join(album, artist.c.pk==album.c.fk))
```
```
artist JOIN album ON artist.pk = album.fk
```

In [184…
```
from sqlalchemy.ext.declarative import declarative_base
```

In [185…
```
base = declarative_base()
```

In [186…
```
base.metadata.tables
```

Out[186]:
```
FacadeDict({})
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: