Bilkul! Aapka diya gaya code SQLAlchemy ORM ka example hai jisme 2 tables (`User` aur `Address`) ban rahe hain Python classes ke zariye. Main aapko **har ek cheez** asaan Urdu mein samjhaata hoon — **line by line**.

---

1. Typing Module se Import

```
python from typing import List, Optional
```

- `List`: Python mein list ka type batata hai. Jaise `List[int]` ka matlab hai integer ki list.
- `Optional`: Yeh batata hai ke value ho bhi sakti hai ya `None` bhi ho sakti hai.

Example:

```
python fullname: Optional[str] # ya to string hoga ya None
```

---

2. SQLAlchemy Se Import

```
python from sqlalchemy import ForeignKey, String
```

- `ForeignKey`: 2 tables ke darmiyan relation banata hai. Jaise har address ek user se link hota hai.
- `String`: Database ke liye string type column define karta hai.

```
python from sqlalchemy.orm import DeclarativeBase, Mapped,
relationship, mapped_column
```

- `DeclarativeBase`: ORM system mein base class banata hai. Isse aap ke sab model classes derive hoti hain.
- `Mapped`: Batata hai yeh attribute database column se map hota hai.
- `relationship`: Do models ke darmiyan connection banata hai.
- `mapped_column`: Column define karne ka naya tareeqa hai (SQLAlchemy 2.0+ mein).

---

3. Base Class

```
python class Base(DeclarativeBase): pass
```

- Sab ORM models is class se inherit karte hain.
- Yeh batata hai ke `Base` se banne wali classes SQLAlchemy table hongi.

---

1. User Model (User Table)

```
python class User(Base): __tablename__ = "user_account"
```

- Yeh table ka naam `user_account` hoga.

➤ Columns:

```
python id: Mapped[int] = mapped_column(primary_key=True)
```

- `id` column banayega jo integer hoga aur primary key hoga (unique + auto-incremented).

```
python name: Mapped[str] = mapped_column(String(30))
```

- `name` column string hoga (30 characters max).

```
python fullname: Mapped[Optional[str]]
```

- `fullname` optional string hai. Agar user ne full name na diya to `None` ho sakta hai.

➤ Relationship:

```
python addresses: Mapped[List["Address"]] = relationship(
back_populates="user", cascade="all, delete-orphan" )
```

- **Yeh ek `User` ke kai `Address` store karta hai** (one-to-many relation).
- `back_populates="user"` ka matlab: `Address` table mein bhi `user` ka reference hoga.
- `cascade="all, delete-orphan"` ka matlab:
  - Agar user delete ho jaye, to uske addresses bhi delete ho jayein.
  - Agar address user se alag ho jaye to woh bhi delete ho jaye.

➤ repr method:

```
python def __repr__(self) -> str: return f"User(id={self.id!r},
name={self.name!r}, fullname={self.fullname!r})"
```

- Jab aap `print(user)` karo, to yeh readable string return karega.

Example Output:

```
python User(id=1, name='Ali', fullname='Ali Raza')
```

---

2. Address Model (Address Table)

```
python class Address(Base): __tablename__ = "address"
```

- Table ka naam `address` hoga.

➤ Columns:

```
python id: Mapped[int] = mapped_column(primary_key=True)
```

- `id` primary key hoga (unique address ID).

```
python email_address: Mapped[str]
```

- `email_address` string field hai, user ka email yahan store hota hai.

```
python user_id: Mapped[int] =
mapped_column(ForeignKey("user_account.id"))
```

- Yeh `user_id` column `user_account` table ke `id` column se link hoga (foreign key).

➤ Relationship:

```
python user: Mapped["User"] =
relationship(back_populates="addresses")
```

- Yeh batata hai ke har address ka aik user hota hai.
- `back_populates="addresses"` ka matlab hai `User` model mein `addresses` ke zariye link hai.

➤ repr method:

```
python def __repr__(self) -> str: return f"Address(id={self.id!
r}, email_address={self.email_address!r})"
```

- Jab `print(address)` karte ho, to yeh readable string return karega.

Example Output:

```
python Address(id=1, email_address='ali@example.com')
```

Final Summary Table

| Concept | Description | | ---------- | ---------- | | `Base` | Parent class for models | | `User` | Represents a user, contains `id`, `name`, `fullname`, and list of addresses | | `Address` | Represents an address, contains `id`, `email_address`,

and `user_id` | | `relationship()` | Links one model to another | | `ForeignKey` | Connects one table to another | | `__repr__()` | Print-friendly representation |