HW5

5.1

A.

```
class Customer(SQLModel, table=True):
    id: int | None = Field(default=None, primary_key=True)
    first_name: str
    last_name: str
```

B.

```
class Product(SQLModel, table=True):
    id: int | None = Field(default=None, primary_key=True)
    name: str
    price: float
```

C.

```
class Order(SQLModel, table=True):
    id: int | None = Field(default=None, primary_key=True)
    customer_id: int = Field(foreign_key="customer.id")
    product_id: int = Field(foreign_key="product.id")
    status: str
    quantity: int
```

D.

```
from sqlmodel import SQLModel, Field, create_engine, Session

sqlite_url = "sqlite:///hw5.db"
engine = create_engine(sqlite_url, echo=True)
SQLModel.metadata.create_all(engine)

with Session(engine) as session:
    c1 = Customer(first_name="Daniel", last_name="Brown")
    c2 = Customer(first_name="Kevin", last_name="Anderson")
    c3 = Customer(first_name="Sophia", last_name="Hall")

    p1 = Product(name="Smartphone", price=170.26)
    p2 = Product(name="Laptop", price=360.82)
    p3 = Product(name="LED Television", price=665.00)
    p4 = Product(name="Gaming Console", price=460.67)
    p5 = Product(name="Smartwatch", price=892.68)

    session.add_all([c1, c2, c3, p1, p2, p3, p4, p5])
    session.commit()
    o1 = Order(customer_id=c1.id, product_id=p5.id, status="Shipped", quantity=2)
    o2 = Order(customer_id=c3.id, product_id=p1.id, status="Delivered", quantity=1)
    o3 = Order(customer_id=c1.id, product_id=p4.id, status="Delivered", quantity=2)
    o4 = Order(customer_id=c2.id, product_id=p5.id, status="Delivered", quantity=3)

    session.add_all([o1, o2, o3, o4])
    session.commit()
```

5.1

A.

```
@app.get("/customers/", response_model=list[CustomerRead])
def read_customers(session: Session = Depends(get_session)):
    return session.exec(select(Customer)).all()
```

B.

```
@app.get("/customers/{id}", response_model=CustomerRead)
def read_customer(id: int, session: Session = Depends(get_session)):
    customer = session.get(Customer, id)
    if customer is None:
        raise HTTPException(404, f"Customer {id} not found")
    return customer
```

C.

```
@app.post("/customers/", response_model=int)
def create_customer(customer: CustomerCreate, session: Session = Depends(get_session)):
    db_customer = Customer.model_validate(customer)
    session.add(db_customer)
    session.commit()
    session.refresh(db_customer)
    return db_customer.id
```

D.

```
@app.put("/customers/{id}", response_model=CustomerRead)
def update_customer(id: int, data: CustomerCreate, session: Session = Depends(get_session)):
    customer = session.get(Customer, id)
    if customer is None:
        raise HTTPException(404, f"Customer {id} not found")
    customer.sqlmodel_update(data.model_dump())
    session.add(customer)
    session.commit()
    session.refresh(customer)
    return customer
```

E.

```
@app.delete("/customers/{id}")
def delete_customer(id: int, session: Session = Depends(get_session)):
    customer = session.get(Customer, id)
    if customer is None:
        raise HTTPException(404, f"Customer {id} not found")
    session.delete(customer)
    session.commit()
```