4.2. Student Handout

Flask-SQLAlchemy Student Handout

Introduction to Flask-SQLAlchemy

Flask-SQLAlchemy is an extension for Flask that integrates SQLAlchemy, a powerful Object Relational Mapper (ORM). An ORM allows you to interact with your database using Python objects instead of raw SQL queries.

Installing and Configuring Flask-SQLAlchemy

Installation

Install Flask-SQLAlchemy using pip:

```
pip install Flask-SQLAlchemy
```

Configuration

Set up Flask-SQLAlchemy in your Flask app:

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy


app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///mydatabase.db'
db = SQLAlchemy(app)
```

Creating Models and Mapping Them to Database Tables

Example Model

```
class User(db.Model):

id = db.Column(db.Integer, primary_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)
```

Additional Examples

1. **Product Model:**

```
class Product(db.Model):

id = db.Column(db.Integer, primary_key=True)

name = db.Column(db.String(100), nullable=False)

price = db.Column(db.Float, nullable=False)
```

2. **Order Model:**

```
class Order(db.Model):

id = db.Column(db.Integer, primary_key=True)

order_date = db.Column(db.DateTime, nullable=False)

customer_id = db.Column(db.Integer, nullable=False)
```

3. **Category Model:**

```
class Category(db.Model):

id = db.Column(db.Integer, primary_key=True)

name = db.Column(db.String(50), unique=True, nullable=False)
```

Defining Relationships Between Tables

One-to-Many Relationship

```
class Post(db.Model):

id = db.Column(db.Integer, primary_key=True)

title = db.Column(db.String(100), nullable=False)

content = db.Column(db.Text, nullable=False)

user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)


user = db.relationship('User', backref='posts')
```

Additional Examples

1. **Author and Book:**

```
class Author(db.Model):

id = db.Column(db.Integer, primary_key=True)

name = db.Column(db.String(100), nullable=False)
```

```
class Book(db.Model):

id = db.Column(db.Integer, primary_key=True)

title = db.Column(db.String(100), nullable=False)

author_id = db.Column(db.Integer, db.ForeignKey('author.id'),
nullable=False)



author = db.relationship('Author', backref='books')
```

2. **Customer and Order:**

```
class Customer(db.Model):

id = db.Column(db.Integer, primary_key=True)

name = db.Column(db.String(100), nullable=False)



class Order(db.Model):

id = db.Column(db.Integer, primary_key=True)

customer_id = db.Column(db.Integer, db.ForeignKey('customer.id'),
nullable=False)



customer = db.relationship('Customer', backref='orders')
```

3. **Department and Employee:**

```
class Department(db.Model):

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

```
    name = db.Column(db.String(100), nullable=False)



class Employee(db.Model):

    id = db.Column(db.Integer, primary_key=True)

    department_id = db.Column(db.Integer, db.ForeignKey('department.id'),
nullable=False)



    department = db.relationship('Department', backref='employees')
```

Performing CRUD Operations

Create

```
new_user = User(username='john_doe', email='john@example.com')

db.session.add(new_user)

db.session.commit()
```

Read

```
user = User.query.filter_by(username='john_doe').first()

print(user.email)
```

Update

```
user = User.query.filter_by(username='john_doe').first()
```

```
user.email = 'new_email@example.com'

db.session.commit()
```

Delete

```
user = User.query.filter_by(username='john_doe').first()

db.session.delete(user)

db.session.commit()
```

Additional Examples

1. **Create Product:**

```
new_product = Product(name='Laptop', price=999.99)

db.session.add(new_product)

db.session.commit()
```

2. **Read Order:**

```
order = Order.query.filter_by(id=1).first()

print(order.order_date)
```

3. **Update Category:**

```
category = Category.query.filter_by(name='Electronics').first()

category.name = 'Gadgets'
```

```
db.session.commit()
```

Querying Data from the Database

Example Queries

- **Get all users:** `User.query.all()`
- **Filter by condition:** `User.query.filter_by(username='john_doe').all()`
- **Order by a column:** `User.query.order_by(User.username).all()`

Additional Examples

1. **Get all products:**

```
products = Product.query.all()
```

2. **Filter orders by customer ID:**

```
orders = Order.query.filter_by(customer_id=1).all()
```

3. **Order categories by name:**

```
categories = Category.query.order_by(Category.name).all()
```

Handling Database Migrations Using Flask-Migrate

Installation

```
pip install Flask-Migrate
```

Initialization

```
from flask_migrate import Migrate


migrate = Migrate(app, db)
```

Creating and Applying Migrations

- **Create Migration:**

```
flask db migrate -m "Added new column to User"
```

- **Apply Migration:**

```
flask db upgrade
```

Displaying Data in Templates

Passing Data to Templates

```
@app.route('/users')

def users():

users = User.query.all()
```

```
return render_template('users.html', users=users)
```

Template Example

```
<ul>

{% for user in users %}

<li>{{ user.username }} - {{ user.email }}</li>

{% endfor %}

</ul>
```

Additional Examples

1. **Display Products:**

```
@app.route('/products')

def products():

products = Product.query.all()

return render_template('products.html', products=products)
```

2. **Display Orders:**

```
@app.route('/orders')

def orders():

orders = Order.query.all()

return render_template('orders.html', orders=orders)
```

3. **Display Categories:**

```
@app.route('/categories')

def categories():

categories = Category.query.all()

return render_template('categories.html', categories=categories)
```

Conclusion

Flask-SQLAlchemy simplifies database interactions in Flask applications by allowing you to work with databases using Python objects. Key topics covered include:

- Installing and configuring Flask-SQLAlchemy
- Creating models and mapping them to database tables
- Defining relationships between tables
- Performing CRUD operations
- Querying data
- Handling migrations with Flask-Migrate
- Displaying data in templates

ORM Workflow Diagram

```
Python Code (Models) ---> ORM (SQLAlchemy) ---> SQL Queries ---> Database
```

Feel free to explore further and practice with these examples to solidify your understanding!