الاسم : احمد السيد عبده على فرج | سكشن 1

Chapter 7

Multiple Choice Questions (MCQs)

1. Which Python module is included in the standard library for working with SQLite databases?

a) mysql.connector

b) psycopg2

c) sqlite3

d) sqlalchemy

2. What does conn.commit() do after an INSERT statement?

a) Closes the database connection

b) Saves changes permanently in the database

c) Rolls back the transaction

d) Executes the SQL query again

3. Which placeholder style is used in parameterized queries with sqlite3?

a) %s

b) :param

c) ?

d) $1

4. Which method is used to fetch only the first row of a query result?

a) fetchall()

b) fetchmany()

c) fetchone()

d) next()

5. In SQLAlchemy, which class is commonly used to define ORM models?

a) Base

b) Mapper

c) Session

d) Engine

True / False Questions

1. SQLite databases are stored in memory only and cannot be written to a file. Answer = F

2. Using parameterized queries helps prevent SQL injection attacks. Answer = T

3. The rollback() method can undo uncommitted changes in a transaction. Answer = T

4. SQLAlchemy provides both Core (SQL Expression Language) and ORM interfaces. Answer = T

5. cursor.execute() always returns a list of results. Answer = F

Short Answer Questions

1. What is the difference between fetchone(), fetchmany(n), and fetchall() in database cursors?

Answer: fetchone() return one row

fetchmany(n) return first n rows
fetchall() return all rows

2. Why are parameterized queries preferred over string concatenation when inserting user input into SQL statements?

Answer: They protect the database from SQL Injection attacks.

3. What is a transaction in databases, and why is it important?

Answer: It is a group of database operations executed as a single unit.

4. Write the steps (in order) to connect to an SQLite database and insert a row into a table.

Answer:
Create database
Create curson
Execute the insert statement
Save the changes by call commit()

5. Briefly explain how ORM (Object Relational Mapping) improves database handling in Python.

Answer: It allows developers to work with database tables as Python objects instead of writing SQL statments

Problem 1:

```
import pymysql
connect = pymysql.connect(
    host="localhost",
    user="root",
    password="your_password",
    database="school"
)
cur = connect.cursor()
cur.execute("""
CREATE TABLE IF NOT EXISTS students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    grade FLOAT
)
""")
students = [("Ali", 85.5),("Sara", 92.0),("Mohamed",
78.3)]
cur.executemany("INSERT INTO students (name, grade)
VALUES (%s, %s)", students)
connect.commit()
cur.execute("SELECT * FROM students")
rows = cur.fetchall()
for row in rows:
    print(row)
connect.close()
```

Problem 2:

```
import pymysql

connect = pymysql.connect(
    host="localhost",
    user="root",
    password="your_password",
    database="school"
)
curson = connect.cursor()
name = input("Enter name: ")
grade = float(input("Enter grade: "))
curson.execute("INSERT INTO students (name, grade)
VALUES (%s, %s)",(name, grade))
connect.commit()
curson.execute("SELECT * FROM students")
for row in curson.fetchall():
    print(row)
curson.close()
connect.close()
```

Problem 3:

```
import pymysql

connect = pymysql.connectect(
    host="localhost",
    user="root",
    password="your_password",
    database="school"
)
curson = connect.cursonsor()
try:
    connect.start_transaction()
    curson.execute("INSERT INTO students (name, grade)
VALUES (%s, %s)",("Omar", 90))
    curson.execute("INSERT INTO students (name, grade)
VALUES (%s, %s)",("Laila", 95))
    connect.commit()
```

```
except Exception as e:
    print("Error:", e)
    connect.rollback()
    print("Transaction rolled back!")
curson.execute("SELECT * FROM students")
for row in curson.fetchall():
    print(row)
curson.close()
connect.close()
```

Problem 4:

```
from sqlalchemy import create_engine, Column, Integer,
String, Float
from sqlalchemy.orm import declarative_base,
sessionmaker
Base = declarative_base()
class Student(Base):
    __tablename__ = "students"
    id = Column(Integer, primary_key=True)
    name = Column(String(100))
    grade = Column(Float)
    def __repr__(self):
        return f"Student(id={self.id},
name='{self.name}', grade={self.grade})"
engine =
create_engine("mysql+mysqlconnector://root:your_password
@localhost/school")
Base.metadata.create_all(engine)
Session = sessionmaker(bind=engine)
session = Session()
student1 = Student(name="Hassan", grade=88.5)
student2 = Student(name="Nada", grade=91.2)
session.add_all([student1, student2])
session.commit()
students = session.query(Student).all()
for s in students:
    print(s)
```