

الاسم / اسلام السيد رمزي الغرباوي

سیکشن / 1

CHAPTER 7

Multiple Choice Questions (MCQs)

1. sqlite3

2. Saves changes permanently in the database

3. ?

4. fetchone()

5. Base

True / False Questions

1. False

2. True

3. True

4. True

5. False

Short Answer / Conceptual Questions

1. Difference between `fetchone()`, `fetchmany(n)`, and `fetchall()`.

- **`fetchone()`** → returns a single row (or `None` if no more rows)
- **`fetchmany(n)`** → returns up to `n` rows as a list of rows
- **`fetchall()`** → returns all remaining rows as a list

2. Why are parameterized queries preferred over string concatenation?

- Prevent SQL injection (user input is treated as data, not SQL code).
- ↗ Automatically escape values.
- ↗ Improve security and reliability.
- ↗ Improve performance by allowing query caching.

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

A transaction is a group of SQL operations that succeed or fail as a single unit.

It's important because it ensures ACID properties:

- Atomicity → all or nothing
- ↗ Consistency → database remains valid
- ↗ Isolation → transactions don't interfere

- ☑ Durability → committed changes are saved permanently

4. Steps to connect to SQLite and insert a row

- Import sqlite3
- import sqlite3
- Connect to the database
- conn = sqlite3.connect("example.db")
- Create a cursor
- cursor = conn.cursor()
- Write an INSERT query (parameterized)

- cursor.execute("INSERT INTO users (name, age) VALUES (?, ?)", ("Ali", 25))
- Commit the transaction
- conn.commit()
- Close the connection
- conn.close()

5. How ORM improves database handling in Python

ORM (Object Relational Mapping):

- Maps database tables to Python classes.
- Represents rows as Python objects.

- Lets you work with databases using Python code instead of SQL.
- Reduces errors and improves readability.
- Makes switching databases easier (SQLite → MySQL → PostgreSQL).

Example:

```
user = User(name="Ali", age=25)
```

```
session.add(user)
```

```
session.commit()
```

#Problem 1

```
import sqlite3
```

```
conn = sqlite3.connect("school.db")
```

```
cursor = conn.cursor()
```

```
cursor.execute("""
```

```
CREATE TABLE IF NOT EXISTS students(
```

```
    id INTEGER PRIMARY KEY,
```

```
    name TEXT,
```

```
    grade REAL
```

```
)
```

```
""")
```

```
students = [
```

```
(1, "Ali", 85.5),
```

```
(2, "Sara", 92.0),
```

```
(3, "Mohamed", 78.3)
```

```
]
```

```
cursor.executemany("INSERT OR REPLACE INTO  
students VALUES (?, ?, ?)",
```

```
students)
```

```
conn.commit()
```

```
cursor.execute("SELECT * FROM students")  
rows = cursor.fetchall()  
  
for row in rows:  
    print(row)
```

```
conn.close()
```

#Problem 2

```
import sqlite3
```

```
conn = sqlite3.connect("school.db")
```

```
cursor = conn.cursor()
```

```
name = input("Enter name: ")
```

```
grade = float(input("Enter grade:"))
```

```
cursor.execute("select * from students")

rows = cursor.fetchall()

id = rows[-1][0] + 1

cursor.execute("INSERT OR REPLACE INTO students
VALUES(?, ?, ?)" , (id , name ,
grade))

conn.commit()

cursor.execute("select * from students")

rows = cursor.fetchall()

print("--- Updated Records ---")

for row in rows:

    print(row)
```

```
conn.close()
```

#Problem 3

```
import sqlite3
```

```
conn = sqlite3.connect("school.db")
```

```
cursor = conn.cursor()
```

```
try:
```

```
    cursor.execute("BEGIN")
```

```
    print("Started Transaction...")
```

```
    cursor.execute("INSERT OR REPLACE INTO  
    students VALUES(4 , 'Ahmed' , 80.1)")
```

```
    cursor.execute("INSERT OR REPLACE INTO  
students VALUES(5 , 'Khaled' ,  
84.14)")
```

```
    cursor.execute("INSERT OR REPLACE INTO  
students VALUES(6 , 'Mark' , 95.7)")
```

```
x = 1 / 0
```

```
conn.commit()
```

```
except Exception as e:
```

```
    print("Error Happened..")
```

```
    conn.rollback()
```

```
    print("Rollback Executed...")
```

```
cursor.execute("SELECT * FROM students")
```

```
rows = cursor.fetchall()
```

```
for row in rows:
```

```
    print(row)
```

```
conn.close()
```

#Problem 4

```
import sqlite3
```

```
class Book:
```

```
    def __init__(self, id, title, author):
```

```
        self.id = id
```

```
        self.title = title
```

```
        self.author = author
```

```
connection = sqlite3.connect("books.db")
```

```
cursor = connection.cursor()
```

```
cursor.execute("""
```

```
CREATE TABLE IF NOT EXISTS books(
```

```
    id INTEGER PRIMARY KEY,
```

```
    title TEXT,
```

```
author TEXT  
)  
"""")  
  
books = [  
    Book(1, "The Alchemist", "Paulo Coelho"),  
    Book(2, "Atomic Habits", "James Clear")  
]  
  
  
  
  
book_tuples = [(b.id, b.title, b.author) for b in books]  
  
cursor.executemany("INSERT OR REPLACE INTO  
books VALUES(?, ?, ?)", book_tuples)  
  
connection.commit()  
  
cursor.execute("SELECT * FROM books")  
  
rows = cursor.fetchall()  
  
for row in rows:  
    print(row)
```

`connection.close()`