Code for to do list application :

import sqlite3

from datetime import datetime

# Database setup

def setup\_database():

conn = sqlite3.connect('todo\_list.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS tasks (

id INTEGER PRIMARY KEY,

title TEXT NOT NULL,

description TEXT,

due\_date TEXT,

priority TEXT,

status TEXT DEFAULT 'incomplete'

)

''')

conn.commit()

conn.close()

# Add a new task

def add\_task(title, description, due\_date, priority):

conn = sqlite3.connect('todo\_list.db')

cursor = conn.cursor()

cursor.execute('''

INSERT INTO tasks (title, description, due\_date, priority)

VALUES (?, ?, ?, ?)

''', (title, description, due\_date, priority))

conn.commit()

conn.close()

print(f"Task '{title}' added successfully.")

# View tasks with optional filters

def view\_tasks(show\_completed=True, show\_incomplete=True, sort\_by\_due\_date=False):

conn = sqlite3.connect('todo\_list.db')

cursor = conn.cursor()

query = 'SELECT \* FROM tasks WHERE '

conditions = []

if show\_completed:

conditions.append("status = 'complete'")

if show\_incomplete:

conditions.append("status = 'incomplete'")

query += " OR ".join(conditions)

if sort\_by\_due\_date:

query += " ORDER BY due\_date ASC"

cursor.execute(query)

tasks = cursor.fetchall()

conn.close()

if not tasks:

print("No tasks found.")

return

print("\nTasks List:")

for task in tasks:

print(f"{task[0]}. {task[1]} [Priority: {task[4]}, Due: {task[3]}, Status: {task[5]}]")

print(f" Description: {task[2]}")

# Mark a task as completed

def mark\_task\_completed(task\_id):

conn = sqlite3.connect('todo\_list.db')

cursor = conn.cursor()

cursor.execute('''

UPDATE tasks SET status = 'complete' WHERE id = ?

''', (task\_id,))

conn.commit()

conn.close()

print(f"Task {task\_id} marked as completed.")

# Mark a task as incomplete

def mark\_task\_incomplete(task\_id):

conn = sqlite3.connect('todo\_list.db')

cursor = conn.cursor()

cursor.execute('''

UPDATE tasks SET status = 'incomplete' WHERE id = ?

''', (task\_id,))

conn.commit()

conn.close()

print(f"Task {task\_id} marked as incomplete.")

# Delete a task

def delete\_task(task\_id):

conn = sqlite3.connect('todo\_list.db')

cursor = conn.cursor()

cursor.execute('DELETE FROM tasks WHERE id = ?', (task\_id,))

conn.commit()

conn.close()

print(f"Task {task\_id} deleted.")

# Main function to run the to-do list manager

def main():

setup\_database()

while True:

print("\nTo-Do List Manager")

print("1. Add Task")

print("2. View Tasks")

print("3. Mark Task as Completed")

print("4. Mark Task as Incomplete")

print("5. Delete Task")

print("6. Exit")

choice = input("Enter your choice: ")

if choice == '1':

title = input("Enter task title: ")

description = input("Enter task description: ")

due\_date = input("Enter due date (YYYY-MM-DD): ")

priority = input("Enter priority (Low, Medium, High): ")

add\_task(title, description, due\_date, priority)

elif choice == '2':

show\_completed = input("Show completed tasks? (y/n): ").lower() == 'y'

show\_incomplete = input("Show incomplete tasks? (y/n): ").lower() == 'y'

sort\_by\_due\_date = input("Sort by due date? (y/n): ").lower() == 'y'

view\_tasks(show\_completed, show\_incomplete, sort\_by\_due\_date)

elif choice == '3':

try:

task\_id = int(input("Enter task ID to mark as completed: "))

mark\_task\_completed(task\_id)

except ValueError:

print("Please enter a valid task ID.")

elif choice == '4':

try:

task\_id = int(input("Enter task ID to mark as incomplete: "))

mark\_task\_incomplete(task\_id)

except ValueError:

print("Please enter a valid task ID.")

elif choice == '5':

try:

task\_id = int(input("Enter task ID to delete: "))

delete\_task(task\_id)

except ValueError:

print("Please enter a valid task ID.")

elif choice == '6':

print("Exiting To-Do List Manager.")

break

else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()