```
import json
import os
import datetime


class Subtask:
    def __init__(self, description, completed=False):
        self.description = description
        self.completed = completed

    def __str__(self):
        status = "Completed" if self.completed else "Incomplete"
        return f"- {self.description} ({status})"


class Task:
    def __init__(self, description, category="", priority=1, due_date=None,
reminders=None, subtasks=None, completed=False):
        self.description = description
        self.category = category
        self.priority = priority
        self.due_date = due_date
        self.reminders = reminders if reminders else []
        self.subtasks = subtasks if subtasks else []
        self.completed = completed

    def add_subtask(self, subtask):
        self.subtasks.append(subtask)

    def remove_subtask(self, index):
        try:
            self.subtasks.pop(index)
        except IndexError:
            print("Invalid subtask index.")

    def __str__(self):
        status = "Completed" if self.completed else "Incomplete"
        priority_color = "\033[91m" if self.priority == 1 else "\033[93m" if self.priority == 2
else "\033[92m"
        desc_str = f"{priority_color}{self.description} (Priority: {self.priority}, Due Date:
{self.due_date}, Category: {self.category}, Status: {status})"
        subtasks_str = "\n".join([str(subtask) for subtask in self.subtasks])
        return f"{desc_str}\n{subtasks_str}"


class TodoList:
    def __init__(self):
        self.tasks = []
    def add_task(self, task):
        self.tasks.append(task)
        print(f"Task '{task.description}' added to the list.")

    def remove_task(self, index):
        try:
            task = self.tasks.pop(index)
            print(f"Task '{task.description}' removed from the list.")
        except IndexError:
            print("Invalid task index.")

    def mark_task_complete(self, index):
        try:
            self.tasks[index].completed = True
            print("Task marked as complete.")
        except IndexError:
            print("Invalid task index.")

    def add_subtask_to_task(self, task_index, subtask):
        try:
            self.tasks[task_index].add_subtask(subtask)
            print(f"Subtask '{subtask.description}' added to the task.")
        except IndexError:
            print("Invalid task index.")

    def remove_subtask_from_task(self, task_index, subtask_index):
        try:
            self.tasks[task_index].remove_subtask(subtask_index)
            print("Subtask removed from the task.")
        except IndexError:
            print("Invalid task index or subtask index.")

    def clear_completed_tasks(self):
        self.tasks = [task for task in self.tasks if not task.completed]
        print("Completed tasks cleared.")

    def display_tasks(self, category=None, priority=None, due_date=None,
completed=None):
        filtered_tasks = self.tasks
        if category:
            filtered_tasks = [task for task in filtered_tasks if task.category == category]
        if priority:
            filtered_tasks = [task for task in filtered_tasks if task.priority == priority]
        if due_date:
            filtered_tasks = [task for task in filtered_tasks if task.due_date == due_date]
        if completed is not None:
            filtered_tasks = [task for task in filtered_tasks if task.completed == completed]
        if not filtered_tasks:
            print("No tasks in the to-do list.")
            return

        print("Tasks in the to-do list:")
        for index, task in enumerate(filtered_tasks, start=1):
            print(f"{index}. {task}")

    def save_tasks(self, filename):
        tasks_data = [{"description": task.description, "category": task.category,
"priority": task.priority,
                       "due_date": task.due_date.strftime("%Y-%m-%d") if task.due_date
else None,
                       "reminders": [reminder.strftime("%Y-%m-%d %H:%M") for reminder in
task.reminders],
                       "subtasks": [subtask.description for subtask in task.subtasks],
                       "completed": task.completed} for task in self.tasks]
        with open(filename, "w") as file:
            json.dump(tasks_data, file)
        print(f"Tasks saved to '{filename}'.")

    def load_tasks(self, filename):
        try:
            with open(filename, "r") as file:
                tasks_data = json.load(file)
            self.tasks = []
            for task_data in tasks_data:
                due_date = datetime.datetime.strptime(task_data["due_date"],
"%Y-%m-%d") if task_data["due_date"] else None
                reminders = [datetime.datetime.strptime(reminder, "%Y-%m-%d %H:%M")
for reminder in task_data["reminders"]]
                subtasks = [Subtask(subtask_desc) for subtask_desc in
task_data["subtasks"]]
                self.tasks.append(Task(task_data["description"], task_data["category"],
task_data["priority"], due_date, reminders, subtasks, task_data["completed"]))
            print(f"Tasks loaded from '{filename}'.")
        except FileNotFoundError:
            print("File not found.")
        except json.JSONDecodeError:
            print("Error decoding JSON.")


def main():
    todo_list = TodoList()

    while True:
        print("\nOptions:")
        print("1. Add Task")
        print("2. Remove Task")
        print("3. Mark Task as Complete")
        print("4. Add Subtask to Task")
        print("5. Remove Subtask from Task")
        print("6. Clear Completed Tasks")
        print("7. Display Tasks")
        print("8. Save Tasks")
        print("9. Load Tasks")
        print("10. Quit")

        choice = input("Enter your choice: ")

        if choice == '1':
            description = input("Enter task description: ")
            category = input("Enter task category: ")
            priority = int(input("Enter priority (1: High, 2: Medium, 3: Low): "))
            due_date_str = input("Enter due date (YYYY-MM-DD) or leave empty: ")
            due_date = datetime.datetime.strptime(due_date_str, "%Y-%m-%d") if
due_date_str else None
            todo_list.add_task(Task(description, category, priority, due_date))
        elif choice == '2':
            index = int(input("Enter index of task to remove: ")) - 1
            todo_list.remove_task(index)
        elif choice == '3':
            index = int(input("Enter index of task to mark as complete: ")) - 1
            todo_list.mark_task_complete(index)
        elif choice == '4':
            task_index = int(input("Enter index of task to add subtask to: ")) - 1
            subtask_description = input("Enter subtask description: ")
            todo_list.add_subtask_to_task(task_index, Subtask(subtask_description))
        elif choice == '5':
            task_index = int(input("Enter index of task to remove subtask from: ")) - 1
            subtask_index =
```