```
import sqlite3
import os
from datetime import datetime, timedelta

DB_FILE = "tasks.db"

def create_table():
    conn = sqlite3.connect(DB_FILE)
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS tasks (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            task_name TEXT NOT NULL,
            priority TEXT,
            due_date TEXT,
            completed INTEGER
        )
    ''')

    conn.commit()
    conn.close()

def add_task(task_name, priority, due_date):
```

```
    conn = sqlite3.connect(DB_FILE)
    cursor = conn.cursor()

    cursor.execute('''
        INSERT INTO tasks (task_name, priority, due_date, completed)
        VALUES (?, ?, ?, ?)
    ''', (task_name, priority, due_date, 0))

    conn.commit()
    conn.close()

def remove_task(task_id):
    conn = sqlite3.connect(DB_FILE)
    cursor = conn.cursor()

    cursor.execute('DELETE FROM tasks WHERE id = ?', (task_id,))

    conn.commit()
    conn.close()

def mark_task_completed(task_id):
    conn = sqlite3.connect(DB_FILE)
    cursor = conn.cursor()
```

```
    cursor.execute('UPDATE tasks SET completed = 1 WHERE id = ?', (task_id,))

    conn.commit()
    conn.close()

def display_tasks():
    conn = sqlite3.connect(DB_FILE)
    cursor = conn.cursor()

    cursor.execute('SELECT * FROM tasks')
    tasks = cursor.fetchall()

    if not tasks:
        print("No tasks found.")
    else:
        for task in tasks:
            task_id, task_name, priority, due_date, completed = task
            status = "Completed" if completed else "Pending"
            print(f"{task_id}. {task_name} - Priority: {priority}, Due Date: {due_date}, Status: {status}")

    conn.close()

def main():
    create_table()
```

```
while True:
    print("\n==== To-Do List ====")
    print("1. Add Task")
    print("2. Remove Task")
    print("3. Mark Task as Completed")
    print("4. Display Tasks")
    print("0. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
        task_name = input("Enter task name: ")
        priority = input("Enter priority (high/medium/low): ")
        due_date = input("Enter due date (YYYY-MM-DD): ")
        add_task(task_name, priority, due_date)
        print("Task added successfully.")

    elif choice == "2":
        task_id = input("Enter task ID to remove: ")
        remove_task(task_id)
        print("Task removed successfully.")

    elif choice == "3":
```

```
            task_id = input("Enter task ID to mark as completed: ")
            mark_task_completed(task_id)
            print("Task marked as completed.")

        elif choice == "4":
            display_tasks()

        elif choice == "0":
            print("Exiting program.")
            break

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

if __name__ == "__main__":
    main()

import os
import json
from datetime import datetime

TRANSACTIONS_FILE = "transactions.json"

def load_transactions():
```

```
    if os.path.exists(TRANSACTIONS_FILE):
        with open(TRANSACTIONS_FILE, "r") as file:
            return json.load(file)
    else:
        return {"income": [], "expenses": []}

def save_transactions(transactions):
    with open(TRANSACTIONS_FILE, "w") as file:
        json.dump(transactions, file)

def add_transaction(transactions, transaction_type, category, amount):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    transaction = {"timestamp": timestamp, "category": category, "amount": amount}

    transactions[transaction_type].append(transaction)
    save_transactions(transactions)

def calculate_budget(transactions):
    income_total = sum(transaction["amount"] for transaction in transactions["income"])
    expense_total = sum(transaction["amount"] for transaction in transactions["expenses"])
    budget_remaining = income_total - expense_total

    return budget_remaining
```

```
def analyze_expenses(transactions):
    expense_categories = set(transaction["category"] for transaction in
transactions["expenses"])

    print("\n=== Expense Analysis ===")
    for category in expense_categories:
        category_total = sum(transaction["amount"] for transaction in transactions["expenses"]
if transaction["category"] == category)
        print(f"{category}: ${category_total:.2f}")

def main():
    transactions = load_transactions()

    while True:
        print("\n==== Budget Tracker ====")
        print("1. Add Income")
        print("2. Add Expense")
        print("3. View Budget")
        print("4. Expense Analysis")
        print("0. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            category = input("Enter income category: ")
```

```
        amount = float(input("Enter income amount: "))
        add_transaction(transactions, "income", category, amount)
        print("Income added successfully.")

    elif choice == "2":
        category = input("Enter expense category: ")
        amount = float(input("Enter expense amount: "))
        add_transaction(transactions, "expenses", category, amount)
        print("Expense added successfully.")

    elif choice == "3":
        budget_remaining = calculate_budget(transactions)
        print(f"\n=== Budget Overview ===\nRemaining Budget: ${budget_remaining:.2f}")

    elif choice == "4":
        analyze_expenses(transactions)

    elif choice == "0":
        print("Exiting program.")
        break

    else:
        print("Invalid choice. Please try again.")
```

```
if __name__ == "__main__":
    main()
```