```
import streamlit as st
import sqlite3
import pandas as pd
from time import sleep

# Funkcja pomocnicza do połączenia z bazą danych
def get_db_connection():
    conn = sqlite3.connect("lista_zakupow.db")
    conn.row_factory = sqlite3.Row
    return conn

# Funkcja do wyświetlania wszystkich rekordów
def view_all_records():
    conn = get_db_connection()
    # Pobieranie wszystkich rekordów
    records = conn.execute("SELECT * FROM zakupy").fetchall()
    conn.close()

    df = pd.DataFrame(records, columns=records[0].keys()) if records else pd.DataFrame()
    st.title("Wszystkie Rekordy")
    st.dataframe(df)

    if not df.empty:
        # Przycisk do zmiany statusu "kupic" na "tak"
        record_id = st.selectbox("Wybierz ID rekordu, aby oznaczyć jako do kupienia", df['id'])
        if st.button("Oznacz jako do kupienia"):
            conn = get_db_connection()
            conn.execute("UPDATE zakupy SET kupic = 'tak' WHERE id = ?", (record_id,))
            conn.commit()
            conn.close()
            st.success(f"Rekord o ID {record_id} został oznaczony jako do kupienia")
            sleep(1)
            st.rerun()

# Funkcja do wyświetlania rekordów, które należy kupić
def view_buy_records():
    conn = get_db_connection()
    # Pobieranie tylko rekordów, gdzie "kupic" = "tak"
    records = conn.execute("SELECT * FROM zakupy WHERE kupic = 'tak'").fetchall()
    conn.close()

    df = pd.DataFrame(records, columns=records[0].keys()) if records else pd.DataFrame()
    st.title("Rekordy do Kupienia")
    st.dataframe(df)

    if not df.empty:
        # Przycisk do zmiany statusu "kupic" na "nie"
        record_id = st.selectbox("Wybierz ID rekordu do oznaczenia jako kupiony", df['id'])
        if st.button("Oznacz jako kupiony"):
            conn = get_db_connection()
            conn.execute("UPDATE zakupy SET kupic = 'nie' WHERE id = ?", (record_id,))
            conn.commit()
            conn.close()
```

```
            st.success(f"Rekord o ID {record_id} został oznaczony jako kupiony")
            sleep(1)
            st.rerun()

# Funkcja do dodawania nowego rekordu
def add_record():
    st.subheader("Dodaj Rekord")
    numer_grupy = st.number_input("Numer Grupy", min_value=0)
    nazwa_grupy = st.text_input("Nazwa Grupy")
    nazwa_towaru = st.text_input("Nazwa Towaru")
    ilosc_towaru = st.number_input("Ilość Towaru", min_value=0)
    jednostka = st.selectbox("Jednostka", ["sztuka", "opakowanie", "litr", "kg"])
    kupic = st.selectbox("Kupić", ["tak", "nie"])

    if st.button("Dodaj"):
        conn = get_db_connection()
        conn.execute(
            "INSERT INTO zakupy (numer_grupy, nazwa_grupy, nazwa_towaru, ilosc_towaru, jednostka, kupic) VALUES (?, ?, ?,
?, ?, ?)",
            (numer_grupy, nazwa_grupy, nazwa_towaru, ilosc_towaru, jednostka, kupic),
        )
        conn.commit()
        conn.close()
        st.success("Rekord został dodany")

# Funkcja do usuwania rekordu
def delete_record():
    st.subheader("Usuń Rekord")
    record_id = st.number_input("ID Rekordu", min_value=0)
    if st.button("Usuń"):
        conn = get_db_connection()
        conn.execute("DELETE FROM zakupy WHERE id = ?", (record_id,))
        conn.commit()
        conn.close()
        st.success(f"Rekord z ID {record_id} został usunięty")

# Funkcja do edytowania rekordu
def edit_record():
    st.subheader("Edytuj Rekord")
    record_id = st.number_input("ID Rekordu do Edycji", min_value=0)

    conn = get_db_connection()
    record = conn.execute("SELECT * FROM zakupy WHERE id = ?", (record_id,)).fetchone()
    conn.close()

    if record:
        numer_grupy = st.number_input("Numer Grupy", min_value=0, value=record["numer_grupy"])
        nazwa_grupy = st.text_input("Nazwa Grupy", value=record["nazwa_grupy"])
        nazwa_towaru = st.text_input("Nazwa Towaru", value=record["nazwa_towaru"])
        ilosc_towaru = st.number_input("Ilość Towaru", min_value=0, value=record["ilosc_towaru"])
          jednostka = st.selectbox("Jednostka", ["sztuka", "opakowanie", "litr", "kg"], index=["sztuka", "opakowanie",
"litr", "kg"].index(record["jednostka"]))
        kupic = st.selectbox("Kupić", ["tak", "nie"], index=["tak", "nie"].index(record["kupic"]))
```

```
        if st.button("Aktualizuj"):
            conn = get_db_connection()
            conn.execute(
                 "UPDATE zakupy SET numer_grupy = ?, nazwa_grupy = ?, nazwa_towaru = ?, ilosc_towaru = ?, jednostka = ?,
kupic = ? WHERE id = ?",
                (numer_grupy, nazwa_grupy, nazwa_towaru, ilosc_towaru, jednostka, kupic, record_id),
            )
            conn.commit()
            conn.close()
            st.success("Rekord został zaktualizowany")
    else:
        st.warning("Rekord nie istnieje")

# Główna funkcja aplikacji
def main():
    st.sidebar.title("Menu")
     choice = st.sidebar.selectbox("Wybierz opcję", ["Przeglądaj do kupienia", "Przeglądaj wszystkie", "Dodaj", "Usuń",
"Edytuj"])

    if choice == "Przeglądaj do kupienia":
        view_buy_records()
    elif choice == "Przeglądaj wszystkie":
        view_all_records()
    elif choice == "Dodaj":
        add_record()
    elif choice == "Usuń":
        delete_record()
    elif choice == "Edytuj":
        edit_record()

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

Wyjaśnienia:

```
import streamlit as st
import sqlite3
import pandas as pd
from time import sleep
```

W tym fragmencie importujemy niezbędne moduły: 'streamlit' do tworzenia interfejsu użytkownika, 'sqlite3' do połączeń z bazą danych SQLite, 'pandas' do manipulowania danymi oraz 'sleep' z modułu 'time' do sprawienia, że program przestaje działać na określony czas.

```
def get_db_connection():
    conn = sqlite3.connect("lista_zakupow.db")
    conn.row_factory = sqlite3.Row
    return conn
```

Funkcja 'get_db_connection' jest funkcją pomocniczą, służącą do nawiązania połączenia z bazą danych 'lista_zakupow.db'. W funkcji tej ustawiamy również opcję 'row_factory' na 'sqlite3.Row', aby wyniki z bazy danych były zwracane jako obiekty, które umożliwiają dostęp do danych za pomocą nazw kolumn.

```
def view_all_records():
```

```
    conn = get_db_connection()
    records = conn.execute("SELECT * FROM zakupy").fetchall()
    conn.close()
    df = pd.DataFrame(records, columns=records[0].keys()) if records else pd.DataFrame()
    st.title("Wszystkie Rekordy")
    st.dataframe(df)
```

Funkcja 'view_all_records' służy do wyświetlania wszystkich rekordów z tabeli 'zakupy' w bazie danych. Najpierw nawiązujemy połączenie z bazą danych, następnie pobieramy wszystkie rekordy i zamykamy połączenie. Następnie tworzymy ramkę danych DataFrame w pandas na podstawie pobranych rekordów i wyświetlamy ją w interfejsie użytkownika.

```
if not df.empty:
        record_id = st.selectbox("Wybierz ID rekordu, aby oznaczyć jako do kupienia", df['id'])
        ...
```

W tym fragmencie sprawdzamy, czy ramka danych nie jest pusta. Jeżeli nie jest, umożliwiamy użytkownikowi wybór rekordu do oznaczenia jako 'do kupienia'. Używamy do tego celu funkcji 'selectbox' z pakietu 'streamlit', który generuje pasek wyboru w interfejsie użytkownika.

```
if st.button("Oznacz jako do kupienia"):
            conn = get_db_connection()
            ...
            st.success(f"Rekord o ID {record_id} został oznaczony jako do kupienia")
```

Jeśli użytkownik naciśnie przycisk 'Oznacz jako do kupienia', to skrypt połączy się z bazą danych, zmieni status rekordu na 'do kupienia' i poinformuje o tym użytkownika, wyświetlając na stronie komunikat sukcesu.

```
def view_buy_records():
    conn = get_db_connection()
    records = conn.execute("SELECT * FROM zakupy WHERE kupic = 'tak'").fetchall()
    ...
```

Funkcja 'view_buy_records' podobna jest do funkcji 'view_all_records', ale wybiera tylko te rekordy, które są oznaczone jako 'do kupienia' w bazie danych.

```
def add_record():
    st.subheader("Dodaj Rekord")
    numer_grupy = st.number_input("Numer Grupy", min_value=0)
    ...
```

Funkcja 'add_record' pozwala na dodawanie nowych rekordów do bazy danych. Użytkownik ma możliwość wprowadzenia informacji o numerze grupy, nazwie grupy, nazwie towaru, ilości towarów, jednostce i czy trzeba to kupić. Informacje są następnie zapisywane w bazie danych.

```
def delete_record():
    st.subheader("Usuń Rekord")
    record_id = st.number_input("ID Rekordu", min_value=0)
    ...
```

Funkcja 'delete_record' pozwala na usuwanie rekordów z bazy danych. Użytkownik podaje ID rekordu, który chce usunąć, a po naciśnięciu przycisku 'Usuń' rekord ten jest usuwany z bazy danych.

```
def edit_record():
    st.subheader("Edytuj Rekord")
    record_id = st.number_input("ID Rekordu do Edycji", min_value=0)
    ...
```

Funkcja 'edit_record' pozwala na edycję istniejących rekordów w bazie danych. Użytkownik podaje ID rekordu, który

chce edytować, i może następnie zmienić dowolne z pól tego rekordu.

```
def main():
    st.sidebar.title("Menu")
       choice = st.sidebar.selectbox("Wybierz opcję", ["Przeglądaj do kupienia", "Przeglądaj wszystkie", "Dodaj", "Usuń", "Edytuj"])
    ...
```

Funkcja 'main' jest główną funkcją aplikacji. Wyświetla menu boczne iz wyborem opcji do wykonania przez użytkownika (przeglądanie, dodawanie, usuwanie, edycja rekordów). Wybrana przez użytkownika opcja jest następnie obsługiwana przez odpowiednią funkcję.