```
// src/hooks/useCards.js
import { useState, useEffect } from 'react';
import { supabase } from '../lib/supabase';

export const useCards = () => {
  const [cards, setCards] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const fetchCards = async () => {
    try {
      setLoading(true);
      const { data, error } = await supabase
        .from('cards')
        .select('*')
        .order('created_at', { ascending: false });

      if (error) throw error;
      setCards(data || []);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    fetchCards();

    // Subscribe to real-time changes
    const channel = supabase
      .channel('cards-changes')
      .on(
        'postgres_changes',
        { event: '*', schema: 'public', table: 'cards' },
        () => {
          fetchCards();
        }
      )
      .subscribe();

    return () => {
      supabase.removeChannel(channel);
    };
  }, []);

  const getCardById = async (cardId) => {
    const { data, error } = await supabase
      .from('cards')
      .select('*')
      .eq('card_id', cardId)
      .single();
    if (error) throw error;
    return data;
  };

  const addCard = async (cardData) => {
    const { data, error } = await supabase
      .from('cards')
      .insert([cardData])
      .select()
      .single();

    if (error) throw error;
    return data;
  };

  const deleteCard = async (id) => {
    const { error } = await supabase
      .from('cards')
      .delete()
      .eq('id', id);

    if (error) throw error;
  };

  const generateCardId = () => {
    return Math.random().toString(36).substring(2, 8).toUpperCase();
  };

  return {
    cards,
    loading,
    error,
    getCardById,
    addCard,
    deleteCard,
    generateCardId,
    refetch: fetchCards,
  };
};
```