import { useQuery, UseQueryOptions, QueryKey } from '@tanstack/react-query';

type UseCustomQueryProps<TData, TError> = {

queryKey: QueryKey;

queryFn: () => Promise<TData>;

options?: Omit<UseQueryOptions<TData, TError, TData>, 'queryKey' | 'queryFn'>;

};

export function useCustomQuery<TData = unknown, TError = unknown>({

queryKey,

queryFn,

options,

}: UseCustomQueryProps<TData, TError>) {

return useQuery({

queryKey,

queryFn,

...options

});

}

import { useMutation, UseMutationOptions } from '@tanstack/react-query';

type UseCustomMutationProps<TData, TVariables, TError> = {

mutationFn: (variables: TVariables) => Promise<TData>;

options?: Omit<UseMutationOptions<TData, TError, TVariables>, 'mutationFn'>;

};

export function useCustomMutation<TData = unknown, TVariables = void, TError = unknown>({

mutationFn,

options,

}: UseCustomMutationProps<TData, TVariables, TError>) {

return useMutation({

mutationFn,

...options,

});

}

const loginMutation = useCustomMutation({

mutationFn: api.login,

options: {

onSuccess: () => {

queryClient.invalidateQueries({ queryKey: ['fetchMe'] });

dispatch(setIsAuthOpen(false));

}

}

});

// вызвать:

loginMutation.mutate({ username: 'user', password: 'pass' });

const { data, status } = useCustomQuery({

queryKey: ['fetchMe'],

queryFn: api.fetchMe,

options: { retry: 0 }

});