[사용자] Neon Postgres와 Drizzle ORM을 사용하여, 사용자가 좋아요를 누른 코인들의 데이터를 테이블로 출력해보기

구현 어려움 정도	⭐⭐⭐
구현 완료	☑
기간	@December 26, 2024
정리 완료	☑

Neon Postgres와 Drizzle ORM을 사용하여, 사용자가 좋아요를 누른 코인들의 데이터를 테이블로 출력하는 기능을 구현했습니다. 아래는 코드에 대한 설명입니다.

> [개발환경] Neon Postgres과 Drizzle ORM을 활용해 테이블 만들기

위 링크에 Neon Postgres를 활용해 테이블을 만드는 방법을 작성했습니다.

1.데이터베이스 스키마 정의

Drizzle ORM을 사용하여 `user` 테이블과 `like` 테이블을 정의해줍니다.

- user 테이블은 사용자 정보를 저장합니다.
- like 테이블은 각사용자가 좋아요를 누른 코인에 대한 정보(coinSymbol)을 저장하며 userId와 coinSymbol이 주요 데이터입니다.
- user와 like는 1:N 관계로 하나의 사용자는 여러 개의 좋아요를 가질 수 있습니다.
- like 테이블은 userId를 외래키로 user 테이블을 참조합니다. 사용자가 삭제될 경우 해당 사용자가 좋아요를 누른 모든 코인 기록도 자동으로 삭제됩니다.
  → `onDelete: 'cascade'` 로 설정했기 때문

```
// src/db/schema.ts
import { relations } from "drizzle-orm";
import { pgTable, timestamp, uuid, text, varchar } from "drizzle-orm/pg-core";

export const user = pgTable('users', {
    ...
});

export const userRelations = relations(user, ({ many }) => ({
  likes: many(like),
}));

export const like = pgTable("likes", {
  id: uuid('id').defaultRandom().notNull().primaryKey(),
  userId: uuid('userId').references(() => user.id, {onDelete: 'cascade'}).notNull(),
  coinSymbol: varchar("coinSymbol", { length: 50 }).notNull(),
  createdAt: timestamp("created_at").defaultNow().notNull(),
});

export const likeRelations = relations(like, ({ one }) => ({
  user: one(user, {
    fields: [like.userId],
    references: [user.id]
  }),
}));
```

2. 좋아요 데이터 조회 함수

사용자가 좋아요를 누른 모든 코인들을 조회하는 함수입니다.

- getUerLikes 함수는 userId를 인자로 받아 해당 사용자가 좋아요를 누른 모든 코인의 coinSymbol 값을 배열로 반환합니다.
- `db.query.like.findMany` 메서드를 사용해 like 테이블에서 사용자의 userId와 일치하는 모든 좋아요 데이터를 조회합니다.
- 반환된 데이터는 coinSymbol만 추출해 배열로 반환합니다.

```
// src/data/liked.ts

import { db } from "@/db";
import { like } from "@/db/schema"; // 'like' 테이블을 import 해줍니다.
import { eq } from "drizzle-orm";

// 사용자가 좋아요를 누른 모든 코인 데이터를 가져오는 함수
export const getUserLikes = async (userId: string): Promise<string[] | null> => {
  if (!userId || userId.trim() === "") return null;  // 유효하지 않은 userId를 처리합니다.

  try {
    // 'like' 테이블에서 userId에 해당하는 모든 좋아요 데이터를 가져오고
    // 해당하는 'coin' 데이터를 포함하여 반환
    const userLikes = await db.query.like.findMany({
      where: eq(like.userId, userId),  // 필터링 조건: userId가 일치하는 데이터를 가져옵니다.
    });

    // 좋아요한 코인들의 symbol을 반환
    return userLikes.map((likeRecord) => likeRecord.coinSymbol);
  } catch (error) {
    console.error("좋아요 데이터를 불러오는데 실패했습니다.", error);
  }
};
```

3. 좋아요 데이터를 reactQuery로 처리

reactQuery를 사용해 서버에서 데이터를 비동기적으로 가져오는 hook입니다.

- useLikedData 혹은 reactQuery를 사용해 서버에서 userId에 해당하는 좋아요 데이터를 가져옵니다.
- getUserLikes 함수를 호출해 userId에 해당하는 코인들의 coinSymbol을 비동기로 가져옵니다.
- 이 데이터는 data, isLoading, isError로 반환되며 data, 로딩, 에러 상태에 대한 처리를 할 수 있습니다.

```
// src/hooks/useLikedData.ts
import { useQuery } from '@tanstack/react-query';
import { getUserLikes } from "@/data/liked";

export const useLikedData = ({ userId }: {userId: string}) => {
  const { data, isLoading, isError } = useQuery({
    queryKey: ['likedCoins', userId],
    queryFn: async () => {
      const data = await getUserLikes(userId);

      return data;
    }
  })

  return {
    data,
    isLoading,
    isError,
```

```
  };
}
```

4. 좋아요 데이터를 테이블로 렌더링

사용자가 좋아요를 누른 코인들을 테이블 형식으로 출력하는 컴포넌트이며 tanstack/react-table를 사용해 table을 만들었습니다.

- `likedCoinsData`에서 좋아요한 코인들의 coinSymbol을 기반으로, data에서 해당 심볼을 가진 코인들만 필터링하여 `filteredData`에 저장합니다.
- `useReactTable`을 사용하여 테이블을 설정하고, 페이지네이션과 정렬 기능을 추가합니다.
- 좋아요한 코인이 없으면 예외 처리 메시지를 출력하고, 데이터가 있으면 필터링된 코인 데이터를 테이블에 렌더링해줍니다.

```
// LikedCoinsTable.tsx
import { useUserStore } from "@/stores/useUserStore";
import { useLikedData } from "@/hooks/useLikedData";
import { createColumns } from './Columns';
import { useCurrencyStore } from "@/stores/useCurrencyStore";
import { useMarketData } from "@/hooks/useMarketData";
import { getCoreRowModel, getFilteredRowModel, getPaginationRowModel, getSortedRowModel, PaginationState, S
import { useEffect, useMemo, useState } from "react";
import TableBody from "./TableBody";
import TableHeader from "./TableHeader";
import CoinPagination from "./CoinPagination";

const LikedCoinsTable = () => {
  const user = useUserStore((state) => state.user);
  const [filteredData, setFilteredData] = useState<any[]>([]);
  const { data: likedCoinsData } = useLikedData({ userId: user?.id ?? '' });
  const { currency } = useCurrencyStore();
  const { coinListAll: data } = useMarketData({ currency });

  const [sorting, setSorting] = useState<SortingState>([]);

  const columns = useMemo(() => createColumns(currency), [currency]);

  const [pagination, setPagination] = useState<PaginationState>({
    pageIndex: 0,
    pageSize: 10,
  });

  useEffect(() => {
    if (likedCoinsData && data) {
      setFilteredData(
        data.filter((coin: any) => likedCoinsData.includes(coin.symbol))
      );
    }
  }, [likedCoinsData, data]);

  const table = useReactTable({
    data: filteredData || [],
    columns,
    state: { sorting, pagination },
    onSortingChange: setSorting,
    onPaginationChange: setPagination,
    getCoreRowModel: getCoreRowModel(),
    getSortedRowModel: getSortedRowModel(),
    getFilteredRowModel: getFilteredRowModel(),
    getPaginationRowModel: getPaginationRowModel(),
  });
```

```
  const pageCount = table.getPageCount();

  const handlePageChange = (newPageIndex: number) => {
    setPagination((prev) => ({
      ...prev,
      pageIndex: newPageIndex,
    }));
  };

  return (
    <div className="pb-[120px]">
      <table className="w-full table-fixed">
        <thead>
          <TableHeader headerGroups={table.getHeaderGroups()} />
        </thead>
        <tbody>
          {filteredData.length === 0 ? (
            <tr>
              <td className="w-[1200px] flex flex-col items-center pt-[100px] pb-[160px] text-[20px] leadir
                관심 자산이 없습니다. 거래소에서 하트를 눌러 추가해보세요.
              </td>
            </tr>
          ) : (
            <TableBody rows={table.getRowModel().rows} />
          )}
        </tbody>
      </table>
      {filteredData.length !== 0 && (
        <CoinPagination
          pageCount={pageCount}
          pageIndex={pagination.pageIndex}
          onPageChange={handlePageChange}
        />
      )}
    </div>
  )
}

export default LikedCoinsTable
```

전체 코드 확인하러 가기

- commit: https://github.com/wwowww/crypto/commit/2b59877d86d3273c87782188ab03e61775e41bd5
- schema: https://github.com/wwowww/crypto/blob/main/src/db/schema.ts