Manko:

Tapasztalni fogják, hogy a Python program készítését nem fogják tudni elkerülni.

A jegyzőkönyvben szerepeljen, hogy ki melyik rész elvégzésében vett részt, és a végén

foglalják össze, hogy milyen eredményre jutottak az adatok elemzése során.

Ez nem sablon feladat! Nem csak egy megoldása létezik. Különböző utakon is lehet jó

eredményre jutni. Legyenek kreatívak főleg az adattisztítás és a hiba azonosítás területén!

**A végén kérni fogom a jegyzőkönyvet (Word, vagy PDF), a teljes Python kódot, és az**

**adatbázisfájlt.**

Az alábbi 12 pont a tananyag 12 fejezetével van összhangban:

1. Miért érdemes a kapott adatokat Relációs adatbázisban kezelni

2. Tervezzék meg az adatbázist ER diagram segítségével. Lehet, hogy néhányszor újra

fogják tervezni, ezeknek a miértjét és eredményeit nyugodtan írják le.

3. Hozzák létre az adatbázist. A választott metódust indokolják meg.

4. Töltsék fel az adatokat a csv fájlokból a megfelelő táblákba. Ellenőrizzék az adatok és

az adattípusok helyességét!

5. DCL és TCL: SQLite-n belül nincs jogosultságkezelés.

a. Írják le, hogy melyik résztvevőnek milyen jogosultságra lett volna szüksége, a

munkájához.

b. Vizsgálják meg, hogy van-e, és ha van, akkor milyen esetben

tranzakciókezelésre.

c. 1 tranzakciót tegyenek bele akkor is, ha nem feltétlen szükséges csak a példa

kedvéért.

6. Lekérdezések

a. Végezzenek el lekérdezéseket, melyek összecsatolnak táblákat (JOIN).

b. Használjanak aggregációs függvényeket.c. Írják le, hogy miért végezték el ezeket a műveleteket, és milyen

következtetésekre jutottak.

7. Allekérdezések, UNION

a. Szükség esetén készítsenek egymásba ágyazott lekérdezéseket

b. A részeredményeiket INSERT és SELECT segítségével tegyék ki külön

táblákba

c. Vizsgálják meg, hogy mit lehetne kimutatni, vagy ellenőrizni az UNION

utasításokkal

8. Teljesítmény optimalizálás

a. Melyik mezőre milyen kulcsot, indexet alkalmaznának, és miért?

b. Milyen módszert javasolnának az adatműveletek gyorsítására, ha megkapnának

10 évnyi adatot és azt kellene feldolgozni?

9. Adattisztítás. Az adatok hibával terheltek.

a. Milyen módszerrel, illetve logika alapján kerestek, szűrtek hibás adatokat?

b. Milyen módszerrel kezelték azokat?

10. Milyen tudásuk alapján tudták azonosítani a hibákat?

a. Mi alapján tekintették a hibás adatokat hibás adatnak? pl. Fizika órán tanulták,

otthon tapasztalták, valamilyen logika alapján stb.

b. Adatbevitelhez milyen érvényesítési szabályokat, milyen kényszereket

javasolnának

11. Az adatok egy ipari adatgyűjtő program írja az adatbázisba.

a. Az adatok rögzítése és feldolgozása során milyen szerepköröket hoznának létre,

és azokat milyen jogosultsággal látnák el?

12. Biztonsági mentésre milyen stratégiát választanának?

Bevezetés

A feladatmankó pdf áttekintése után a megfelelő sorrend kialakítása a feladat. Kezdeti terv lépései:

1. Adatfájlok struktúrájának megállapítása
2. ER diagram elkészítése
3. Adatok betöltése pandas dataframe-be
4. Adattisztítás
5. Táblákra bontás
6. Lekérdezések
7. Teljesítmény optimalizálás
8. Jogosultságkezelés

A feladat megoldása közben kiderült, hogy ez a sorrend nem feltétlenül ideális, mert ha hamarabb készítjük el az ER-Diagrammot, akkor előfodulhat, hogy az adattisztítás során az adattáblák struktúrája változik, így az ER-Diagram módosítására lehet szükség.

# Adatfájl struktúra megállapítás

Adagok.csv oszlopnevek javítás, új oszlopnevek:

* ADAGSZÁM
* Kezdet\_DÁTUM
* Kezdet\_IDŐ
* Vége\_DÁTUM
* Vége\_IDŐ
* ADAGKÖZI\_IDŐ
* ADAGIDŐ

A kezdet\_DÁTUM+Kezdet\_IDŐ oszlopokból képezhető egy Kezdet oszlop, hasonlóan a vég adatokkal Vége néven, természetesen datetime formátumban.

Az Adagok.csv-ben csak 32 értékes sor van, a többit mely NULL értékeket tartalmaz eldobjuk.

A Hutopanelek.csv-ben lévő Time oszlopokat is formázzuk pandas dataframe datetime formátumra, hogy a tartalomnak megfelelően tudjuk az adatokat kezelni.

# Adattisztítás

Hiányzó adatok keresése

Szerencsére nincs hiányzó adat sor-szinten, de hiányzik a 7. szenzor teljes adata.

Nem lehetünk biztosak abban, hogy az Adagidő valóban helyesen felvitt értékeket tartalmaz, így kiszámoljuk és összehasonlítjuk a meglévő adatokkal.

Megállapítjuk, hogy van-e hiányzó adat a Hűtőpanelek.csv-ben Dataframe.isnull() segítségével. Nincs hiányzó adat.

Végül megjelenítjük a 2 dataframe statisztikáit a describe() függvény segítségével, hogy átfogó képet lássunk az adatokról. Látjuk, hogy a Hűtőpanelek.csv hőmérséklet érték oszlopainak statisztikái mind NaN értékűek. Megvizsgálva néhány értéket a fájlban, látszik, hogy a lebegőpontos értéknél a tizedesvessző valóban vessző, ahogy Európában szokás jelölni. A pandas viszont pontot vár, így át kell alakítanunk az értékeket.

Hiányzik az ADAGKÖZI\_IDŐ. A névből arra következtetünk, hogy a két adag között eltelt idő. Ezt könnyen kiszámíthatjuk és hozzáadhatjuk mint Adagkozi\_ido oszlop.

# ER diagram elkészítése

Megállapítjuk az entitásokat. Ez könnyű, mivel 2 táblánk van amik szignifikánsan elszeparálták az entitásokat, úgy mint Adagok és Hűtőpanelek.

A terv, hogy kapcsolatot keressünk a 2 entitás között. Felfedezzük, hogy minden mérés valamelyik adaghoz kapcsolódik, így ezt a kapcsolatot kell megteremteni. Mivel nincs kézenfekvő hozzárendelési szabály(tekintve, hogy nincs a méréskhez adag-id rendelve) létrehozunk egy új oszlopot a hűtőpanelek táblában, mely egyértelműen azonosítja, hogy melyik mérés melyik adaghoz tartozik.

A későbbi feladatok miatt(pl: hogy demonstrálni tudjuk a UNION műveleteket), szétbontjuk a Hűtőpanelek táblát 15 táblára, mindnek ugyanaz a struktúrája.

* adagok
  + Primary Key: adagszam
  + Fields: adagkozi\_ido, adagido, kezdet, vege
* hutopanel1
  + Foreign Key: adagszam (references adagok.adagszam)
  + Fields: Ido, Ertek
* hutopanel2
  + Foreign Key: adagszam (references adagok.adagszam)
  + Fields: Ido, Ertek

A diagram of a computer code

Description automatically generated

Ezen az ER diagrammon csak 2 hűtőpanel táblát ábrázoltunk, a többi 12-t(mivel a 7. hiányzik) ugyanilyen struktúrában töltjük be az adatbázisba.