



**GÁBOR
DÉNES
EGYETEM**

MÉRNÖKINFORMATIKUS

PROJEKTFELADAT

ADATBÁZISOK
GDE_GSM TEAM.

Debreceni Péter
D67OJV

Maurer Patrik
MMZX7Z

Nagy Gábor
D15MSW

Szondi Ákos
VD85Z8

Tartalomjegyzék

1. Kezdetek. - Adatbázis kiválasztás és tervezés	3
2. Adatbázis terv – ER diagram	3
3. Adatbázis létrehozása	3
4. Adatok feltöltése, adatok, adattípusok helyességének ellenőrzése.	5
5. Jogosultságkezelés	5
6. Lekérdezések.....	6
a. Hibás adatok keresése és szűrése – JOIN	6
b. Pelenkénti átlaghőmérséklet számítása: - GROUP BY, ORDER BY	6
c. Aggregációs függvények – adagidő statisztikák	6
d. Aggregációs függvények –Napi hőmérséklet ingadozás.....	6
7. Al-lekérdezések, UNION.....	7
8. Teljesítmény optimalizálás.....	8
9. Adattisztítás - Hibakezelés	9
a) Hibás adatok keresési módszerei	9
b) Hibakezelési módszerek	9
10. Hibazonosítás alapjai.....	10
a) Hiba felismerésének forrásai:.....	10
b) Érvényesítési szabályok:	10
11. Szerepkörök és jogosultságok	11
12. Biztonság és mentés	12
12.1 Többrétegű mentési stratégia	12
12.2 Mentési módszerek	12
12.3 Tárolási biztonság	12
12.4Helyreállítási terv.....	12
13. Python programkód feldolgozási folyamata	13
13.1. Előkészületek	13
13.2. Adatbázis Létrehozás	13
13.3. Import Dekódolás	13
13.4. Adattisztítás	13
13.5. NF3 Normalizálás.....	13
13.6. Adatbázis Betöltés	13
13.7 A kód főbb Jellemzői.....	13
14. A program adatszerkezete:.....	14
15. A program funkcionális vázlata:	14
16. A program mappa vázlata:	14
17. Felhasználói Útmutató – A python Program Használata	15
17.1 A program indítása	15
17.2 Mappák előkészítése	15
17.3 Adatbázis Táblák feladatok.....	15
17.4 Adatok kódolása	16
17.5 Adatidő Tábla Kérdése.....	17
17.6 Használati Tippek.....	17

1. Kezdetek. - Adatbázis kiválasztás és tervezés

Javasolt adatbázis típus:

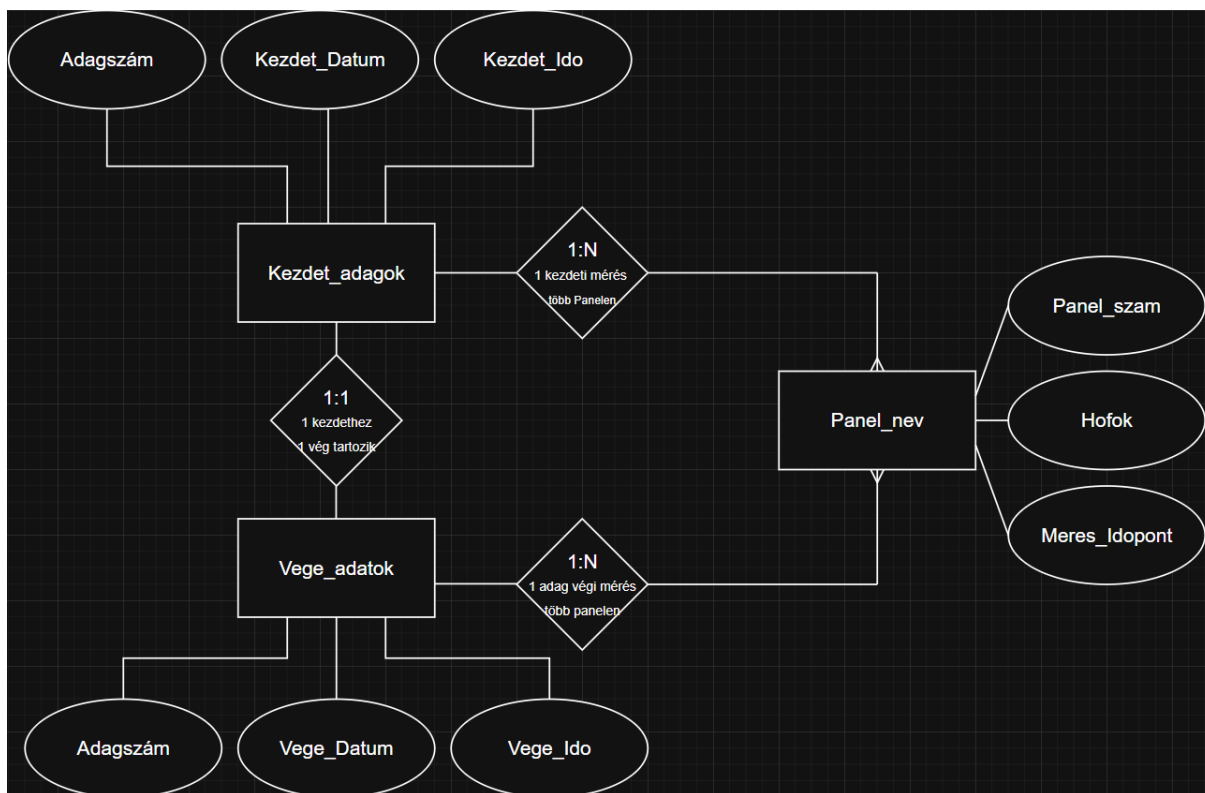
Az adatok tárolására és kezelésére a **relációs adatbázis-kezelő rendszer (RDBMS)**, például az **SQLite** a legalkalmasabb. SQLite egy könnyen használható, fájl alapú adatbázis, amely ideális kisebb és közepes méretű alkalmazásokhoz, ahol nincs szükség szerver alapú megoldásra. A választás oka az SQLite nyújtotta rugalmasság, illetve a feldolgozandó import adatokban található adatstruktúra.

Miért választottuk az SQLite-t?

- **Egyszerűség:** Az SQLite telepítése és karbantartása egyszerű, nem szükséges külön adatbázis szerver.
- **Relációs modell:** A táblák közötti kapcsolatok jól ábrázolhatók, így alkalmas az általunk megtervezett adatstruktúra kezelésére.
- **Teljesítmény:** Kis és közepes mennyiségű adat kezelésére gyors, egyszerű tranzakciókezeléssel.

2. Adatbázis terv – ER diagram

A táblák kapcsolatai az alábbi ER diagram alapján készültek:



ER Diagram:

- **Kezdet_adagok_NFdone** (ADAGSZÁM, Kezdet_DÁTUM, Kezdet_IDŐ)
- **Vege_adatok_NFdone** (ADAGSZÁM, Vége_DÁTUM, Vége_IDŐ)
- **Ido_ellenorzes_NFdone** (ADAGSZÁM, Örökölt_ADAGIDŐ, Számított_ADAGIDŐ, CRC_Error)
- **panel_szam_NFdone** (meres_idopont, hofok, panel_szam)

Kapcsolatok:

- 1:1 kapcsolat a **Kezdet_adagok** és **Vege_adatok** táblák között.
- 1:N kapcsolat a **Kezdet_adagok** és **panel_szam** között.

3. Adatbázis létrehozása

A tervnek megfelelően az adatbázis struktúráját az SQLite segítségével hozzuk létre python kódban, táblákat és kapcsolatokat definiálva.

SQLite választása:

Okok:

- **Könnyűség:** egyetlen fájl, telepítés nem szükséges
- **Projekt méret:** 10-100 MB adat → tökéletes SQLite-hoz
- **Fejlesztési sebesség:** gyors prototípus készítés
- **Kompatibilitás:** minden platformon működik

Létrehozási mód:

python

Automatikus séma generálás

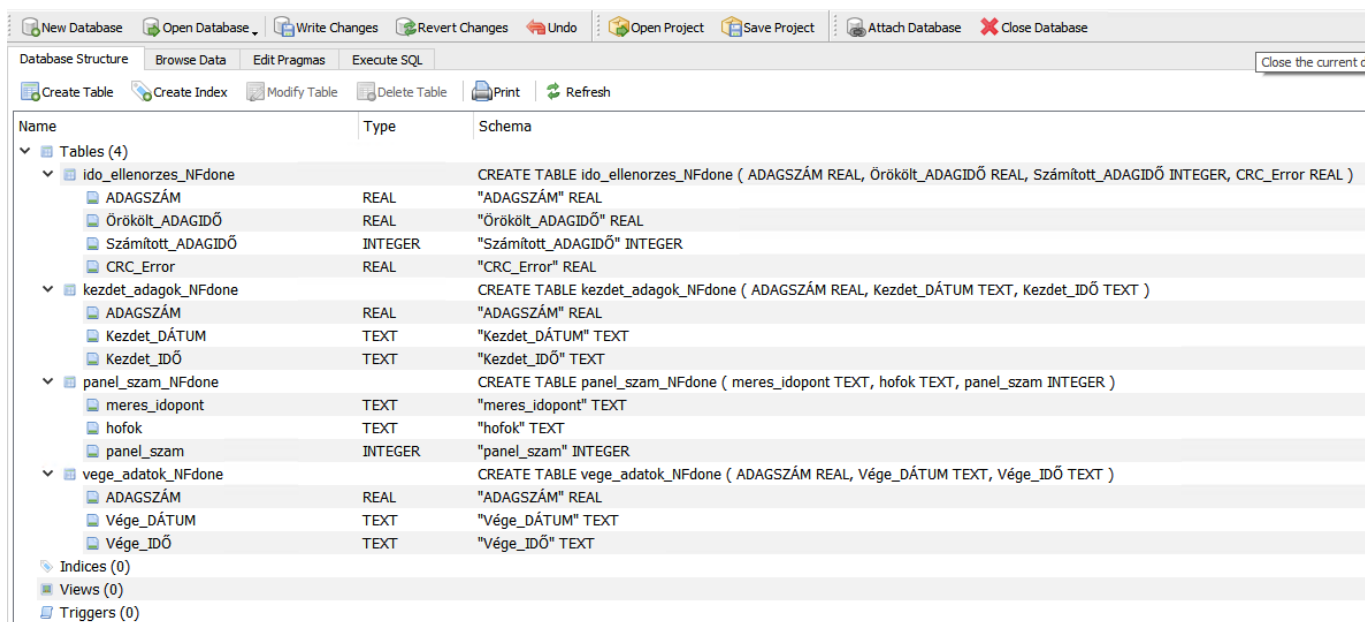
```
def create_database(db_path):
```

```
    conn = sqlite3.connect(db_path) # Fájl automatikus létrejön
```

```
    conn.close()
```

Előnyök:

- **Zero configuration:** nincs szerver beállítás
- **Portabilitás:** adatbázis fájl másolható
- **Alacsony kockázat:** kis projekthez megbízható



Name	Type	Schema
Tables (4)		
ido_ellenorzes_NFdone		CREATE TABLE ido_ellenorzes_NFdone (ADAGSZÁM REAL, Örökölt_ADAGIDŐ REAL, Számított_ADAGIDŐ INTEGER, CRC_Error REAL)
ADAGSZÁM	REAL	"ADAGSZÁM" REAL
Örökölt_ADAGIDŐ	REAL	"Örökölt_ADAGIDŐ" REAL
Számított_ADAGIDŐ	INTEGER	"Számított_ADAGIDŐ" INTEGER
CRC_Error	REAL	"CRC_Error" REAL
kezdet_adagok_NFdone		CREATE TABLE kezdet_adagok_NFdone (ADAGSZÁM REAL, Kezdet_DÁTUM TEXT, Kezdet_IDŐ TEXT)
ADAGSZÁM	REAL	"ADAGSZÁM" REAL
Kezdet_DÁTUM	TEXT	"Kezdet_DÁTUM" TEXT
Kezdet_IDŐ	TEXT	"Kezdet_IDŐ" TEXT
panel_szam_NFdone		CREATE TABLE panel_szam_NFdone (meres_idopont TEXT, hofok TEXT, panel_szam INTEGER)
meres_idopont	TEXT	"meres_idopont" TEXT
hofok	TEXT	"hofok" TEXT
panel_szam	INTEGER	"panel_szam" INTEGER
vege_adatok_NFdone		CREATE TABLE vege_adatok_NFdone (ADAGSZÁM REAL, Vége_DÁTUM TEXT, Vége_IDŐ TEXT)
ADAGSZÁM	REAL	"ADAGSZÁM" REAL
Vége_DÁTUM	TEXT	"Vége_DÁTUM" TEXT
Vége_IDŐ	TEXT	"Vége_IDŐ" TEXT
Indices (0)		
Views (0)		
Triggers (0)		

4. Adatok feltöltése, adatok, adattípusok helyességének ellenőrzése.

A tervnek megfelelően az adatbázis struktúráját az SQLite segítségével hozzuk létre python kódban, táblákat és kapcsolatokat definiálva.

Betöltési folyamat:

1. **Automatikus sémafelismerés** - pandas infers dtypes
2. **Kódolás konverzió** - UTF-8-BOM formátumra
3. **Adattisztítás** - duplikátumok, üres értékek
4. **NF3 normalizálás** - fájl típus szerinti szétbontás
5. **SQLite betöltés** - automatikus típuskonverzió

Ellenőrzési lépések:

```
python
# Adattípus ellenőrzés
print(df.dtypes)
# Oszlopnevek ellenőrzés
print(df.columns.tolist())
# Értéktartomány ellenőrzés
print(df['hofok'].min(), df['hofok'].max())
```

Ellenőrzési pontok:

- **Adattípusok:** integer/float/text helyessége
- **Értéktartományok:** fizikailag lehetséges értékek
- **Kapcsolatok:** külső kulcsok érvényessége
- **Hiányzó adatok:** kötelező mezők kitöltöttsége

5. Jogosultságkezelés

a. Jogosultságok a résztvevők számára:

Mivel SQLite-ban nincs külön jogosultságkezelés, az adatbázis fájl elérhetősége határozza meg, hogy ki férhet hozzá az adatokhoz. Az ipari környezetben a következő szerepköröket érdemes bevezetni:

- **Adminisztrátor:** Teljes hozzáférés az adatbázishoz, beleértve a táblák módosítását, adatkezelést.
- **Felhasználó:** Csak olvasási jogokkal rendelkezik, nem módosíthatja az adatokat.
- **Rendszeradminisztrátor:** A rendszer beállításainak módosítása, tranzakciók figyelemmel kísérése.

b. Tranzakciókezelés:

SQLite támogatja a tranzakciók kezelését. Tranzakcióra akkor van szükség, ha több adatot egyszerre szeretnénk menteni a táblákba, és biztosítani akarjuk, hogy minden adat helyesen legyen rögzítve, vagy ha valamilyen hiba történik, akkor az egész művelet visszavonjuk.

Tranzakció példa:

```
BEGIN TRANSACTION;
INSERT INTO Kezdet_adagok_NFdone (ADAGSZÁM, Kezdet_DÁTUM, Kezdet_IDŐ) VALUES (1, '2025-10-17', '08:00:00');
INSERT INTO Vege_adatok_NFdone (ADAGSZÁM, Vége_DÁTUM, Vége_IDŐ) VALUES (1, '2025-10-17', '09:00:00');
COMMIT;
```

Ez a tranzakció biztosítja, hogy ha bármelyik művelet hibát okozna, akkor az összes adat mentése visszavonásra kerül.

6. Lekérdezések

a. Hibás adatok keresése és szűrése – JOIN

Adagok és időellenőrzés összekapcsolása. Mivel a táblázatban rögzített értékek számítás alapján is előállíthatók, ellenőrizni szükséges, hogy ez valóban igaz –e. (A python kód az ellenőrzést előzetesen elvégezte!)

Alábbi lekérdezés ezen ellenőrzés igazolására készült.

Lekérdezés feltétele, hogy a ido_ellenorzes_NFdone tábla generálása előzetesen megtörténjen!:

```
SELECT k.ADAGSZÁM, k.Kezdet_DÁTUM, k.Kezdet_IDŐ, i.Örökölt_ADAGIDŐ,  
i.Számított_ADAGIDŐ, i.CRC_Error  
FROM kezdet_adagok_NFdone k  
JOIN ido_ellenorzes_NFdone i ON k.ADAGSZÁM = i.ADAGSZÁM  
WHERE i.CRC_Error = 1;b. Hibák kezelése:
```

Indoklás: CRC hibák azonosítására - összekapcsolom az adag kezdeti adatait az időellenőrzéssel, hogy lássam mely adagoknál van eltérés.

Amennyiben minden számított adat egyezik, az eredmény null értékkel tér(t) vissza.

b. Panelenkénti átlaghőmérséklet számítása: - GROUP BY, ORDER BY

```
SELECT p.panel_szam, COUNT(*) as meres_szam, AVG(p.hofok) as atlag_hofok  
FROM panel_szam_NFdone p  
GROUP BY p.panel_szam  
ORDER BY p.panel_szam;
```

Indoklás: Panelenkénti statisztikák - átlaghőmérséklet és mérésszám megállapításához.

c. Aggregációs függvények – adagidő statisztikák

```
SELECT  
COUNT(*) as osszes_adag,  
AVG(Örökölt_ADAGIDŐ) as atlag_adagido,  
MIN(Örökölt_ADAGIDŐ) as legrovidebb_adag,  
MAX(Örökölt_ADAGIDŐ) as leghosszabb_adag,  
SUM(CASE WHEN CRC_Error = 1 THEN 1 ELSE 0 END) as hibas_adagok  
FROM ido_ellenorzes_NFdone;
```

Indoklás: Panelenkénti statisztikák - átlaghőmérséklet és mérésszám megállapításához.

d. Aggregációs függvények –Napi hőmérséklet ingadozás

```
SELECT  
DATE(meres_idopont) as nap,  
MIN(hofok) as min_hofok,  
MAX(hofok) as max_hofok,  
MAX(hofok) - MIN(hofok) as ingadozas  
FROM panel_szam_NFdone  
GROUP BY DATE(meres_idopont)  
ORDER BY nap;
```

Indoklás: Napi hőmérséklet-ingadozás monitorozása - a hőmérséklet stabilitásának ellenőrzéséhez.

7. Al-lekérdezések, UNION

Átlagos adagidőnél hosszabb adagok részletes adatai

```
SELECT k.ADAGSZÁM, k.Kezdet_DÁTUM, k.Kezdet_IDŐ, i.Örökölt_ADAGIDŐ
FROM kezdet_adagok_NFdone k
JOIN ido_ellenorzes_NFdone i ON k.ADAGSZÁM = i.ADAGSZÁM
WHERE i.Örökölt_ADAGIDŐ > (
    SELECT AVG(Örökölt_ADAGIDŐ)
    FROM ido_ellenorzes_NFdone
    WHERE CRC_Error = 0
);
```

Indoklás: Kiemeli az átlag feletti teljesítési idővel rendelkező adagokat, kiszűrve a hibás méréseket a referenciaérték számításánál.

Legmagasabb hőmérsékletű panelek naponta

```
SELECT DATE(meres_idopont) as nap, panel_szam, hofok
FROM panel_szam_NFdone p1
WHERE hofok = (
    SELECT MAX(hofok)
    FROM panel_szam_NFdone p2
    WHERE DATE(p1.meres_idopont) = DATE(p2.meres_idopont)
)
ORDER BY nap;
```

Indoklás: Naponta azonosítja a legmelegebb paneleket, ami a hőeloszlás és potenciális problémás panelek azonosítását segíti.

Extrém hőmérsékletű mérések naplózása

```
CREATE TABLE IF NOT EXISTS extrem_homersektek AS
SELECT panel_szam, meres_idopont, hofok,
    CASE
        WHEN hofok < 10 THEN 'Túl alacsony'
        WHEN hofok > 90 THEN 'Túl magas'
    END as allapot
FROM panel_szam_NFdone
WHERE hofok < 10 OR hofok > 90;

-- Statisztika az extrém értékekről
SELECT allapot, COUNT(*) as darab, AVG(hofok) as atlag_hofok
FROM extrem_homersektek
GROUP BY allapot;
```

Indoklás: Biztonsági kritikus hőmérsékleti tartományok naplózása és kategorizálása.

8. Teljesítmény optimalizálás

a. Kulcsok és indexek alkalmazása:

- **Primér kulcsok:** Az **ADAGSZÁM** a fő kulcs minden táblában, mivel minden mérés ezen az azonosítón keresztül kapcsolódik a többi adathoz.
- **Indexek:** Az **ADAGSZÁM** és **meres_idopont** mezők indexelése segíti a lekérdezések gyorsítását, mivel gyakran használjuk őket kereséshez és kapcsolatokhoz.

```
CREATE INDEX IF NOT EXISTS idx_adagszam ON kezdet_adagok_NFdone (ADAGSZÁM) ;  
CREATE INDEX IF NOT EXISTS idx_meres_idopont ON panel_szam_NFdone (meres_idopont) ;  
CREATE INDEX IF NOT EXISTS idx_crc_error ON ido_ellenorzes_NFdone (CRC_Error) ;
```

Indoklás: Gyakran használt keresési feltételek optimalizálása - adagszám, időbélyeg és hibaállapot szerinti gyors keresés.

b. Adatműveletek gyorsítása 10 évnyi adat esetén

- **Partitioning:** A nagy mennyiségű adatot év vagy hónap szerint partícionálhatjuk, hogy a lekérdezések gyorsabbak legyenek.
- **Indexelés:** Használjunk indexeket az oszlopokon, amelyek gyakran szerepelnek a lekérdezések szűrésében vagy csatlakozásaiban.

Particionálás szemléltetése (SQLite-ben nem támogatott, de logika)

Éves partíciók létrehozása és időalapú szűrés

```
SELECT * FROM panel_szam_NFdone  
WHERE meres_idopont BETWEEN '2024-01-01' AND '2024-12-31' ;
```

Indoklás: Időalapú szegmentálás - nagy adatmennyiség kezelésére éves partíciók használata ajánlott, így csak a releváns időszak adatait kell feldolgozni.

Adatszegmentálás

- **Időalapú partícionálás:** Éves/havi mappákba szétosztás
- **Táblák felosztása:** Külön táblák évek szerint (adagok_2024, adagok_2025)
- **Archiválás:** Régi (5+ éves) adatok külön adatbázisba

Technikai optimalizálások

- **Indexelés:** Időbélyeg, adagszám, panel_szam mezőkre
- **Batch feldolgozás:** 10.000 soros kötegekben történő feldolgozás
- **Memória használat:** Pandas `chunksize` paraméter, streamelés

Folyamatfejlesztés

- **Inkrementális betöltés:** Csak az új/ad módosult adatok feldolgozása
- **Párhuzamos feldolgozás:** Több fájl egyidejű feldolgozása
- **Adattisztítás előszűréssel:** Csak releváns időszak adatainak betöltése

Adatbázis optimalizálás

- **Időalapú WHERE feltételek:** `WHERE év BETWEEN 2020 AND 2024`
- **Ideiglenes táblák:** Részeredmények cache-elése
- **Verziózás:** Éves "snapshot" táblák létrehozása

Kulcselv: "Ne dolgozd fel egyszerre, amit szét is oszthatsz" - időalapú szegmentálás és párhuzamos feldolgozás.

9. Adattisztítás - Hibakezelés

a) Hibás adatok keresési módszerei

Fizikai korlátok alapján

- **Hőmérséklet:** -50°C és 200°C között (ipari környezet fizikai lehetőségei)
- **Időértékek:** dátumok logikai sorrendje (befejezés \geq kezdet)
- **Adagidő:** pozitív érték, maximum 8 óra (műszaki korlát)

Statisztikai alapú szűrés

- **Kiugró értékek:** átlag ± 3 szórás tartományon kívüli értékek
- **Időbeli inkonzisztenciák:** CRC ellenőrzés (számított vs mért idő)
- **Ismétlődések:** duplikált mérések azonos időpontban

Logikai ellenőrzések

- **Hiányzó értékek:** kötelező mezők (pl. adagszám, időbélyeg)
- **Formátum ellenőrzés:** dátum/idő formátum megfeleltetése

b) Hibakezelési módszerek

Automatikus korrekció

```
python
# Hiányzó értékek kitöltése
df.fillna({'hofok': 0, 'ADAGIDŐ': 0})

# Szöveges mezők normalizálása
df['ADAGSZÁM'] = df['ADAGSZÁM'].str.strip().str.upper()
```

Hibás rekordok kezelése

```
python
# 1. Fizikailag lehetetlen értékek szűrése
df_clean = df[(df['hofok'] >= -50) & (df['hofok'] <= 200)]

# 2. Logikai hibák flaggelése
df['CRC_Error'] = abs(df['Számított_ADAGIDŐ'] - df['Örökölt_ADAGIDŐ']) > 1

# 3. Duplikátumok eltávolítása
df_clean = df.drop_duplicates(subset=['ADAGSZÁM', 'meres_idopont'])
```

Stratégia válogatás szerint:

- **Kritikus hibák:** eltávolítás (pl. fizikailag lehetetlen értékek)
- **Gyanús értékek:** flaggelés és megtartás (pl. CRC hibák)
- **Hiányzó adatok:** alapértelmezett értékkel pótlás
- **Formátumhibák:** automatikus korrekció

10. Hibaazonosítás alapjai

a) Hiba felismerésének forrásai:

Fizikai ismeretek:

- Hőmérséklet -50°C alatt \rightarrow lehetetlen (abszolút nulla közelében)
- Hőmérséklet 200°C felett \rightarrow ipari környezetben irreális
- Negatív adagidő \rightarrow idő nem mehet visszafelé

Gyakorlati tapasztalat:

- 8 óránál hosszabb adag \rightarrow gyakorlatilag lehetetlen műszakban
- Azonos időpont dupla mérés \rightarrow műszaki hiba

Logikai következtetés:

- Befejezés időpontja kezdés előtt \rightarrow időutazás nem lehetséges
- CRC eltérés > 1 perc \rightarrow mérési vagy számítási hiba

b) Érvényesítési szabályok:

Alapvető kényszerek:

```
python
# Értéktartományok
hofok: -50 <= érték <= 200
adag_ido: 0 < érték <= 480 # 8 óra
datumok: kezdet <= vége

# Kötelező mezők
adag_szam: nem üres, egyedi
időbélyeg: érvényes dátum-idő formátum
```

Üzleti szabályok:

- Panel 7-es szám soha \rightarrow ismert hiányzó berendezés
- Időeltérés $> 1\%$ \rightarrow automatikus flaggelés
- Duplikált adagszám \rightarrow elutasítás

Elv: "A valós világ fizikai törvényei és az üzleti gyakorlat határozzák meg, mi a hiba"

11. Szerepkörök és jogosultságok

Szerepkörök és jogosultságok:

OPERÁTOR:

- SELECT saját műszak adataira
- INSERT új mérések rögzítése
- UPDATE saját, nem küldött adatok

MŰSZAKVEZETŐ:

- SELECT összes operátor adata
- INSERT/UPDATE hibajavítások
- Nem kap DELETE jogot

TECHNIKUS:

- SELECT összes műszaki adat
- UPDATE kalibrációs adatok
- CREATE ideiglenes táblák

ADATELEMZŐ:

- SELECT összes adat
- CREATE VIEW jelentésekhez
- Nincs módosítási jog

RENDSZERADMIN:

- Teljes jogosultság
- BACKUP/RESTORE műveletek
- Felhasználókezelés

Elv: "Minimális jogosultság elve" - mindenki csak azt lássa/módosíthassa, ami a munkájához szükséges.

12. Biztonság és mentés

12.1 Többrétegű mentési stratégia

Napi különbozeti mentés

- **Mentés:** Minden nap 18:00-kor
- **Tartalom:** Csak az aznapi változások
- **Megtartás:** 7 napig

Heti teljes mentés

- **Mentés:** Vasárnap 02:00-kor
- **Tartalom:** Teljes adatbázis
- **Megtartás:** 4 hétig

Havi archiválás

- **Mentés:** Hónap első napján
- **Tartalom:** Teljes adatbázis + statisztikák
- **Megtartás:** 12 hónapig

12.2 Mentési módszerek

sql

```
-- Napi export kritikus adatokból
.sqlite3 data.db ".backup daily_backup_$(date +%Y%m%d).db"

-- CSV export fontos táblákból
.csv -header -csv panel_szam_NFdone "SELECT * FROM panel_szam_NFdone WHERE
date(meres_idopont) = date('now', '-1 day')"
```

12.3 Tárolási biztonság

- **Helyi:** Gyors helyreállítás érdekében
- **Felhő:** Vészhelyreállítás céljából
- **Offline:** Kritikus adatok fizikai tárolón

12.4 Helyreállítási terv

1. **Adatvesztés esetén:** Legutóbbi teljes mentés + napi különbozetek
2. **Séma hiba:** Csak teljes mentésből visszaállítás
3. **Részleges hiba:** Egyedi táblák visszatöltése CSV-ből

Szlogen: "A mentés nem egy esemény, hanem egy folyamat - több rétegben, rendszeresen, ellenőrizve"

13. Python programkód feldolgozási folyamata

13.1. Előkészületek

- Munka mappák kiürítése, felkészítése (temp, export)
- Felhasználói kérdésekkel, megerősítéssel támogatott munkaművelet

13.2. Adatbázis Létrehozás

- SQLite fájl inicializálás
- Mappa struktúra ellenőrzés

13.3. Import Dekódolás

- Automatikus kódolás-felismerés
- UTF-8-BOM konverzió
- Temp mappába mentés

13.4. Adattisztítás

- Duplikátum szűrés
- Hiányzó értékek kezelése
- Oszlopnév normalizálás

13.5. NF3 Normalizálás

- Fájltypus-specifikus feldolgozás
- Többtáblás séma létrehozás
- CRC ellenőrzés (adagok esetén)

13.6. Adatbázis Betöltés

- Automatikus típusfelismerés
- Táblakészítés és adatimport
- Naplózás és hibakezelés

13.7 A kód főbb Jellemzői

- **Rugalmasság:** Több kódolás támogatása
- **Interaktivitás:** Felhasználói visszajelzés
- **Automatizálás:** Séma- és típusfelismerés
- **Modularitás:** Könnyű karbantarthatóság
- **Hibatűrés:** Átfogó exception kezelés

14. A program adatszerkezete:

Az adatok feldolgozása, tisztítása, normálformára történő átalakítás Python programban történik, azonosan az adatbázis létrehozásával, táblák létrehozásával, adatbetöltéssel

ROOT/CODE/

- └─ FŐPROGRAM ÉS VEZÉRLÉS
 - └─ main.py
- └─ FÁJLKEZELÉSI MODULOK
 - └─ browse.py
 - └─ decoding.py
- └─ ADATFELDOLGOZÓ MODULOK
 - └─ cleaning.py
 - └─ normalizer_prepare.py
- └─ NORMALIZÁLÓ MODULOK
 - └─ normalizer_adagok.py
 - └─ normalizer_homerseklet.py
- └─ ADATBÁZIS MODULOK
 - └─ create2db.py
 - └─ db_loader.py

15. A program funkcionális vázlata:

ADATFELDOLGOZÁSI FOLYAMAT

- └─ 0. ELŐKÉSZÜLETEK
 - └─ Temp mappa kiürítése
 - └─ Export mappa kiürítése
- └─ 1. ADATBÁZIS LÉTREHOZÁS
 - └─ create2db.py
- └─ 2. IMPORT DEKÓDOLÁS
 - └─ browse.py (fájlfeltárás)
 - └─ decoding.py (kódolás konverzió)
- └─ 3. IMPORTÁLT ADATOK TISZTÍTÁS
 - └─ cleaning.py (adattisztítás)
- └─ 4. NORMALIZÁLÁS: NF1-3
 - └─ normalizer_prepare.py (Normálformára hozatal előkészítése)
 - └─ normalizer_adagok.py (3NF adagok tábla)
 - └─ normalizer_homerseklet.py (3NF hőmérséklet tábla)
- └─ 5. ADATBÁZIS BETÖLTÉS
 - └─ db_loader.py (táblakészítés + adatbetöltés)

16. A program mappa vázlata:

PROJECT_ROOT/

- └─ ■ code/ (forráskód)
 - └─ main.py (futtatandó kód)
 - └─ browse.py
 - └─ decoding.py
 - └─ cleaning.py
 - └─ normalizer_prepare.py
 - └─ normalizer_adagok.py
 - └─ normalizer_homerseklet.py
 - └─ create2db.py
 - └─ db_loader.py
- └─ ■ import/ (bemeneti CSV fájlok)
- └─ ■ temp/ (átmeneti dekódolt fájlok)
- └─ ■ export/ (tisztított és normalizált fájlok)
- └─ ■ db/ (adatbázis fájlok)
 - └─ data.db
- └─ ■ requirements.txt (függőségek)

17. Felhasználói Útmutató – A python Program Használata

17.1 A program indítása

A program futása során néhány helyen döntéseket kell hoznia. Íme, hogy mikor mire számíthat és mit javasolunk. **Indítás a main.py modul futtatásával lehetséges.**

17.2 Mappák előkészítése

Mikor jelenik meg:

- Amikor a temp vagy export mappák már tartalmazznak fájlokat

Mit kérdez a program:

```
text
temp mappa már tartalmaz 5 fájlt:
  ■ file1.csv
  ■ file2.csv
...
Kiürítsem a temp mappát? (i/n):
```

Melyek a lehetséges válaszok:

- **i (igen)** – javasolt válasz, ha új adatokat szeretne feldolgozni
- **n (nem)** - ha a meglévő fájlokkal szeretne tovább dolgozni

Miért? Így biztosíthatja, hogy ne keveredjenek össze a régi és új adatok.

17.3 Adatbázis Táblák felladatok

Mikor jelenik meg:

- Ha az adatbázis már tartalmaz korábbi táblákat

Mit kérdez a program:

```
text
Adatbázis már tartalmaz 3 táblát:
  □ table1
  □ table2
...
Töröljem és hozzam létre újra ezeket a táblákat? (i/n):
```

Mi a legjobb válasz:

- **i (igen)** - mindig ezt válassza

Miért? Így biztos, hogy a legfrissebb adatok kerülnek be és a séma mindig naprakész marad.

17.4 Adatok kódolása

a, Alacsony megbízhatóságú kódolás

Mikor jelenik meg:

- Csak akkor, ha a program nem biztos a fájl kódolásában, azaz a kódolás sikeressége <95%

Mit kérdez:

text

△ Alacsony megbízhatóság (75%) → kézi választás szükséges
Szeretnéd használni az automatikusan felismert kódolást? (i/n):

Mi a legjobb válasz:

- **n (nem)** - válassza ezt, és a program felajánl alternatív, régebbi kódolásokat

Mi történik ha igennel válaszol? Akkor a program futása leáll és a felhasználóra bizza a inkább, hogy más programmal (pl. Notepad++) konvertálja a fájlt UTF-8-BOM formátumra, mielőtt folytatná.

b, Kézi kódolás választás

Mikor jelenik meg:

- Ha régebbi kódolásra van szükség, mert az automatikus nem megbízható

Mit kérdez:

text

Működő kódolások:

1. latin2
2. cp852
3. cp1250
4. ? Manuálisan oldom meg (kilépés)

Válassz kódolást (1-3): **2**

Mi a legjobb válasz:

- A program megmutatja, mely kódolás, milyen eredménnyel jár.
- **Nézze meg, melyik kódolás eredményez 100% -ban hibamentes, olvasható adatot és válassza a kódolásnak megfelelő, listában megjelenő számot. (2)**
- Ha egyik sem jeleníti meg helyesen az adatokat, válassza a **3**-ast és próbálja megoldani manuálisan a dekódolást.

17.5 Adatidő Tábla Kérdése

Mikor jelenik meg:

- Az "Adagok" fájl feldolgozása után, ha minden számított időadat pontos egyezést mutat az örökölt adatokkal

Mit kérdez:

text

KÉRDÉS: Minden adagidő pontos (150 adag)

Megtartsam az ellenőrző táblát tájékoztatás céljából? (i/n):

Mi a legjobb válasz:

- n (nem)** amennyiben további feladatot nem szeretne a táblával végezni. Mivel ez számítható adat, tárolása felesleges

Miért? A táblázat felesleges adatokat tartalmaz, tekintve azok a már más táblákban tárolt adatokból kiszámítható.

17.6 Használati Tippek

Első futtatáskor:

- Minden "Kiürítsem a mappát?" kérdésre válaszoljon **i**-vel
- Adatbázis táblák törlésére is **i**-vel válaszoljon
- Kódolás kérdéseknél próbálja a javasolt opciókat

Ismételt futtatáskor:

- Ugyanazt a mintát kövesse, hogy mindig friss adatokkal dolgozzon

Ha problémák merülnek fel:

- Kódolás kérdéseknél inkább lépjen ki és külsőleg oldja meg
- Mindig készítsen biztonsági másolatot a fontos adatairól

A program részletesen jelzi, hogy hol tart a feldolgozásban, így mindig tudni fogja, hogy éppen mi történik az adataival.