

מטרה: ליצור אפליקציית FULLSTACK שמנהלת תחקור של מאגר מידע של קורסים.

חלקי האפליקציה:

קליינט: טרמינל, מציג תפריט של פעולות. כל פעולה שנבחרת מפעילה פונקציה שמשתמשת בחיבור DB.

DB: חיבור לDB של MySQL, מרימים בעזרת XAMPP, מתחברים בעזרת CONNECTOR.

התפריט צריך להיות בדיוק כך:

1. Load CSV into DB

2. Search records by institution name

3. Search records by course name

4. Find most/least common course

5. Show course count per district

6. Free SQL query

7. Exit

הפרויקט משתמש ב-:

- `mysql-connector-python` (או `PyMySQL`)

- טבלה אחת בלבד: `courses`

- רק פקודות SQL גולמיות (raw SQL)

- ללא ORM וללא מחלקות מודלים

- ארכיטקטורה נקייה (קבצים נפרדים לפי אחריות)

📁 מבנה תיקיות מומלץ

```
soldier_courses_explorer/  
  app.py           # תפריט + הדפסה יפה של תוצאות  
  config.py        # פרטי חיבור למסד הנתונים  
  db/  
    connection.py  # get_connection()
```

```
load_csv.py      # CSV-לוגיקה לטעינת ה
queries.py       # SQL-כל שאילתות ה
data/
courses.csv      # CSV-קובץ ה שסופק
```

קובץ ה-courses.csv CSV מכיל כותרת עמודות בפורמט:

```
institution,city,address,course,district,telephone,email
```

דוגמה לשורה:

```
יאיר אסודי מקצועות המים הצלה, יסוד המעלה, יסוד המעלה ת.ד. 99 מיקוד
50121, הצלה בבריכות שחיה סוג 1, חיפה
yairassodi@gmail.com, 6937152, והצפון
```

שלב 1 — מסד נתונים וטבלה ב-MYSQL

לפני הרצת פייתון, צרו מסד נתונים וטבלה:

```
CREATE DATABASE IF NOT EXISTS soldier_courses_db
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;

USE soldier_courses_db;

CREATE TABLE IF NOT EXISTS courses (
  id INT AUTO_INCREMENT PRIMARY KEY,
  institution VARCHAR(255) NOT NULL, -- שם הגוף המכשיר
  city VARCHAR(255) NOT NULL,        -- ישוב
  address VARCHAR(255),               -- כתובת מלאה
  course VARCHAR(255) NOT NULL,       -- שם הקורס
  district VARCHAR(255),              -- ('מחוז (חיפה והצפון וכו')
  telephone VARCHAR(50),              -- טלפון / שלוחה
  email VARCHAR(255)                  -- כתובת מייל
);
```

שלב 2 — דרישות פונקציונליות לכל אופציה בתפריט

להלן מה שכל סעיף בתפריט חייב לעשות.

Load CSV into DB¹

מטרה: להכניס את כל השורות מ-`courses.csv` לתוך הטבלה `courses`.

משימות:

- לקרוא את ה-CSV בעזרת המודול `csv` של פייתון.
- לדלג על שורת הכותרת.
- פרמטריית `INSERT` על כל שורה להריץ פקודת

```
INSERT INTO courses (  
    institution, city, address, course,  
    district, telephone, email  
)  
VALUES (%s, %s, %s, %s, %s, %s, %s);
```

העזרו בפקודות שהופיעו במצגת.

Search Records by Institution Name²

קלט מהמשתמש: מילת חיפוש (או חלק משם הגוף/המוסד).

שאלתה לדוגמה:

```
SELECT id, institution, city, course, district, telephone, email  
FROM courses  
WHERE institution LIKE CONCAT('%', %s, '%')  
LIMIT 50;
```

פלט:

הדפסת רשימה של תוצאות, כולל לפחות:

- `id`
- `institution`
- `city`

- course

- telephone / email (לבחירתכם)

Search Records by Course Name³

קלט מהמשתמש: מילת חיפוש בשם הקורס (לדוגמה: "הצלה").

שאלתה לדוגמה:

```
SELECT id, institution, city, course, district, telephone, email
FROM courses
WHERE course LIKE CONCAT('%', %s
;LIMIT 50
```

פלט:

הצגת רשימת קורסים מתאימים, כולל שם הקורס, המוסד, העיר והמחוז.

Find Most/Least Common Course⁴

המטרה: למצוא איזה קורס מופיע הכי הרבה פעמים ואיזה הכי מעט פעמים בטבלה.

:Most common course

```
SELECT course, COUNT(*) AS num
FROM courses
GROUP BY course
ORDER BY num DESC
LIMIT 1;
```

:Least common course

```
SELECT course, COUNT(*) AS num
FROM courses
GROUP BY course
ORDER BY num ASC
LIMIT 1;
```

דרישות:

- להציג גם את שם הקורס וגם את מספר הפעמים (`num`).
- אופציונלי (אתגר):
 - להציג TOP 5 קורסים הכי נפוצים.
 - להציג גם רשימה של מוסדות שמפעילים את הקורס הנפוץ ביותר.

Show Course Count per District 5

מטרה: לראות כמה קורסים יש בכל מחוז.

שאלתה לדוגמה:

```
SELECT district, COUNT(*) AS num_courses
FROM courses
GROUP BY district
ORDER BY num_courses DESC;
```

פלט:

- טבלה קטנה של: `district | num_courses`
- אפשר גם להציג רק המחוז עם הכי הרבה קורסים והמחוז עם הכי מעט.

אתגר קטן:

- אפשר להוסיף אופציה משנה:
 - "Show courses for specific district" עם פרמטר מהמשתמש:

```
SELECT institution, city, course, telephone, email
FROM courses
WHERE district = %s;
```

Free SQL Query 6

מטרה: לתת למשתמש להריץ כל שאלת `SELECT` שירצה על הטבלה `courses`.

משימות:

1. לקרוא מן המשתמש מחרוזת SQL.
 2. לוודא שהיא מתחילה ב-`SELECT` (לא לאפשר `UPDATE`, `DELETE`, `DROP`, וכו').
 - בדיקה פשוטה: `"sql.strip().upper().startswith("SELECT"`
 3. אם השאילתה תקינה — להריץ אותה עם `cursor.execute(sql)` (ללא פרמטרים).
 4. להדפיס את כל השורות שחזרו.
 5. אם השאילתה לא מתחילה ב-`SELECT` — להדפיס הודעת שגיאה מתאימה.
-

הפרדת קוד (ארכיטקטורה נקייה) 🍷

`app.py`

- מכיל את לולאת התפריט:

```
while True:
    print_menu()
    choice = input("Choose option: ")
    # queries / load_csv קריאה לפונקציה המתאימה ב
```

- לא מבצע שאילתות בעצמו – רק קורא לפונקציות ב-`db/queries.py` וב-`db/load_csv.py`.
-

`db/connection.py`

```
import mysql.connector
from config import DB_CONFIG

def get_connection():
    return mysql.connector.connect(
        host=DB_CONFIG["host"],
        user=DB_CONFIG["user"],
        password=DB_CONFIG["password"],
        database=DB_CONFIG["database"]
    )
```

db/load_csv.py

- פונקציה שמקבלת connection או יוצרת אחד בעצמה.
- קוראת את `.data/courses.csv`.
- מריצה `INSERT` פרמטרי לכל שורה.
- מדפיסה כמה שורות נטענו בהצלחה.

db/queries.py

כל פונקציה מקבלת `conn` ופרמטרים, למשל:

```
def search_by_institution(conn, keyword):
    sql = """
    SELECT id, institution, city, course, district, telephone, email
    FROM courses
    WHERE institution LIKE CONCAT('%', %s, '%')
    LIMIT 50;
    """

    cur = conn.cursor()
    cur.execute(sql, (keyword,))
    return cur.fetchall()
```

באופן דומה:

- `search_by_course(conn, keyword)`
 - `get_most_common_course(conn)`
 - `get_least_common_course(conn)`
 - `get_course_count_per_district(conn)`
 - `run_free_query(conn, sql)`
-

