

# תרגיל: בניית מערכת ניהול עובדים עם MySQL

## מטרת התרגיל

MySQL ב-OOP תוך שימוש ב, MySQL בתרגיל זה תבנו מערכת מלאה לניהול עובדים במסד נתונים operations.

## דרישות מוקדמות

```
import mysql.connector
from mysql.connector import Error
```

הגדרות חיבור למסד הנתונים:

- Host: localhost
- User: root
- Password: (הסיסמה שלכם)
- Database: company\_db

## חלק 1: יצירת המחלקה הבסיסית (10 נקודות)

עם EmployeeDB צרו מחלקה בשם:

### 1.1 \_\_init\_\_ מתודת

מתודה שמקבלת את פרמטרי החיבור למסד הנתונים:

- host
- user
- password
- database

self.connection = None נוסף attribute וצרו, attributes-השמרו את הפרמטרים כ

רמז: 💡

```
def __init__(self, host, user, password, database):
    self.host = ...
    self.user = ...
    # המשיכו...
```

## חלק 2: התחברות למסד נתונים (15 נקודות)

### 2.1 connect() מתודת

כתבו מתודה שמתחברת למסד הנתונים:

- לטיפול בשגיאות try-except-השתמשו ב
- mysql.connector.connect() צרו חיבור עם
- "Connected to MySQL" ✓: אם החיבור הצליח, הדפיסו
- אם יש שגיאה, הדפיסו אותה

דוגמת פלט מצופה:

✓ Connected to MySQL

 **רמז:**

- לבדיקה `self.connection.is_connected()` השתמשו ב
- `mysql.connector.Error` תפסו את

## חלק 3 :יצירת טבלה (15 נקודות)

### 3.1 מתודת `create_table()`

עם `employees` כתבו מתודה שיוצרת טבלת

- `id` - `AUTO_INCREMENT`, מפתח ראשי -
- `name` - `VARCHAR(100)`, `NOT NULL`
- `department` - `VARCHAR(50)`
- `salary` - `DECIMAL(10, 2)`

`CREATE TABLE IF NOT EXISTS` השתמשו ב  
**דוגמת פלט**

✓ Table ready

## חלק 4 :הוספת עובד (20 נקודות)

### 4.1 מתודת `add_employee()`

שמוסיפה עובד חדש `add_employee(name, department, salary)` כתבו מתודה

- `%s` (עם `prepared statements` השתמשו ב
- לשמירת השינויים `commit()` בצעו
- של העובד החדש ID-החזירו את ה
- ID-הדפסו הודעת הצלחה עם השם וה

**דוגמת פלט**

✓ Added David Chen (ID: 1)

 **רמז:**

- `cursor.lastrowid` האחרון שנוצר ID-מחזיר את ה
- `cursor.execute(sql, (name, department, salary))` השתמשו ב

## חלק 5 :שליפת מידע (20 נקודות)

### 5.1 מתודת `get_all_employees()`

כתבו מתודה שמחזירה את כל העובדים

- `dictionaries` החזירו רשימה של
- טפלו בשגיאות

### 5.2 מתודת `get_employee_by_id()`

כתבו מתודה שמחזירה עובד ספציפי

- `dictionary` החזירו אחד
- `None` אם לא נמצא החזירו

 **רמז:**

`cursor = self.connection.cursor(dictionary=True)` # `dictionaries` מחזיר

## חלק 6: עדכון ומחיקה (15 נקודות)

### 6.1 מתודת update\_salary()

שם:עדכנת שכר (update\_salary(emp\_id, new\_salary) כתבו מתודה

- UPDATE statement-השתמשו ב
- commit בצעו
- הדפיסו הודעת הצלחה

### 6.2 מתודת delete\_employee()

שם:מחקת עובד (delete\_employee(emp\_id) כתבו מתודה

- DELETE statement-השתמשו ב
- commit בצעו
- הדפיסו הודעת הצלחה

דוגמת פלט

- ✓ Updated salary for employee 1
- ✓ Deleted employee 2

## חלק 7: חיפוש לפי מחלקה (10 נקודות)

### 7.1 מתודת get\_department\_employees()

שם:מחזירה את כל העובדים במחלקה (get\_department\_employees(department) כתבו מתודה מסוימת

- WHERE clause-השתמשו ב
- dictionaries החזירו רשימת

## חלק 8: סגירת חיבור (5 נקודות)

### 8.1 מתודת close()

כתבו מתודה שסוגרת את החיבור למסד הנתונים

- בדקו אם החיבור מחובר לפני הסגירה
- הדפיסו הודעת הצלחה

## בדיקת התוכנית

לאחר שסיימתם, בדקו את הקוד שלכם

```
# יצירת מופע
db = EmployeeDB('localhost', 'root', 'your_password', 'company_db')

# התחברות
db.connect()

# יצירת טבלה
db.create_table()
```

```

# הוספת עובדים
db.add_employee("David Cohen", "Development", 15000)
db.add_employee("Sarah Levi", "Marketing", 12000)
db.add_employee("Michael Ben-David", "Development", 16000)

# שליפת כל העובדים
employees = db.get_all_employees()
for emp in employees:
    print(emp)

# עדכון שכר
db.update_salary(1, 17000)

# חיפוש לפי מחלקה
dev_employees = db.get_department_employees("Development")
print(f"\nDevelopment employees: {len(dev_employees)}")

# סגירת חיבור
db.close()

```

## שאלות בונים 20 נקודות נוספות

- B1.** שמחזירה את השכר הממוצע לכל `get_average_salary_by_department()` הוסיפו מתודה מחלקה
- B2.** שמחזירה עובדים עם שכר מעל הסכום שהוזן `get_high_earners(min_salary)` הוסיפו מתודה
- B3.** שמאפשרת לעדכן כל שדה `update_employee(emp_id, **kwargs)` הוסיפו מתודה (name, department, או salary)
- B4.** `with`-כך שאפשר להשתמש ב (`__enter__` ו-`__exit__`) למחלקה context manager הוסיפו statement

## טיפים כלליים

- ✓ SQL injection למניעת (%s עם) prepared statements-תמיד השתמשו ב
- ✓ `commit()` תמיד בצעו INSERT/UPDATE/DELETE אחרי
- ✓ לטיפול בשגיאות try-except השתמשו ב
- ✓ (או context manager השתמשו ב) אחרי שימוש cursors סגרו
- ✓ בדקו את הקוד שלכם עם נתונים שונים

## הערכה

חלק	נקודות
מחלקה בסיסית	10
התחברות	15

חלק	נקודות
יצירת טבלה	15
הוספת עובד	20
שליפת מידע	20
עדכון ומחיקה	15
חיפוש לפי מחלקה	10
סגירת חיבור	5
<b>סה"כ</b>	<b>110</b>
בנוס	20

בהצלחה 