Computer Science

Project:-

Employee Database

Management

using

MySQL connector:-

Submitted by:

Aditya Singh(05) | XII Science-B (2025-26)

**Introduction:**

This program is a console-based Employee Management System developed in Python using the mysql.connector module for database operations. It allows users to store, manage, and retrieve employee details efficiently through a MySQL database named “employee\_db”.  
  
The system automatically creates the database and the required details table (if they do not already exist) at the time of execution. The table stores essential information about each employee, including:  
  
- Employee ID (Primary Key)  
- Name  
- Basic Salary  
- House Rent Allowance (HRA)  
- Conveyance Allowance (CA)

Through an interactive menu, the program provides the following key functionalities:  
  
1. Add Employee: Add new employee records with validation checks for unique IDs and proper input formats.  
  
2. Search with ID: Quickly find and display details of an employee using their Employee ID.  
  
3. Search with Name: Search for employees by name and view potential matches.  
  
4. Calculate Total Salary: Automatically compute the total salary (Basic Salary + HRA + CA) for a specific employee.  
  
5. Calculate Total Allowance: Compute the combined allowances (HRA + CA) for an employee.  
  
6. Exit: Safely close the database connection and end the program.

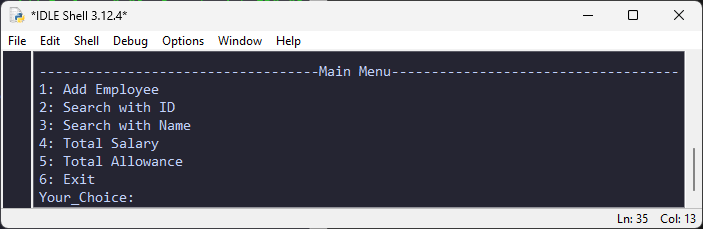
**Source code:**

employee\_menu\_3-v2.py

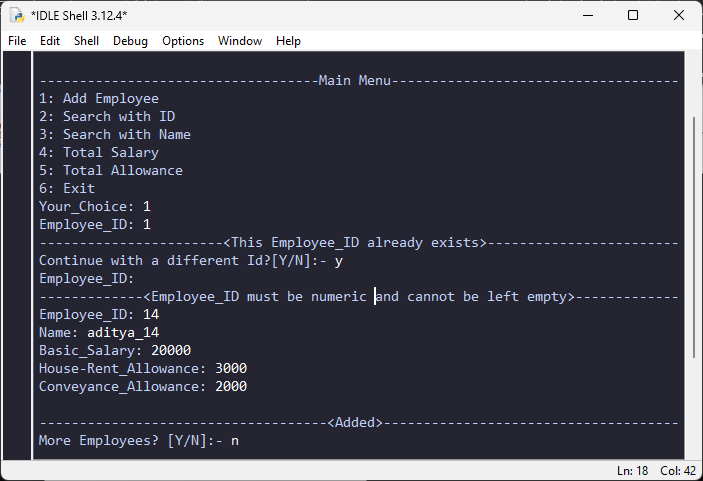
|  |  |  |
| --- | --- | --- |
| 1 | **import** mysql.connector **as** m | |
| 2 | **from** time **import** sleep | |
| 3 | con=m.connect(user="root",password="258000",host="localhost") | |
| 4 | cur=con.cursor() | |
| 5 | cur.execute('show databases like "employee\_db";') | |
| 6 | **if** cur.fetchall()==[]: | |
| 7 | cur.execute("create database employee\_db;use aditya0923;create table details(id int primary key,name varchar(50) not null,basic\_salary int default 0,HRA int default 0,CA int default 0);") | |
| 8 | con=m.connect(user="root",password="258000",host="localhost",database="employee\_db") | |
| 9 | cur=con.cursor() | |
| 10 | **else**: | |
| 11 | con=m.connect(user="root",password="258000",host="localhost",database="employee\_db") | |
| 12 | cur=con.cursor() | |
| 13 | **def validate\_id**(): | |
| 14 | **while** True: | |
| 15 | **try**: | |
| 16 | e\_id=int(input("Employee\_ID: ")) | |
| 17 | cur.execute(f"select id from details where id={e\_id};") | |
| 18 | **if** cur.fetchall()==[]: | |
| 19 | **return** False,e\_id | |
| 20 | **else**: | |
| 21 | **return** True,e\_id | |
| 22 | **except**: | |
| 23 | print("<Employee\_ID must be numeric and cannot be left empty>".center(80,"-")) | |
| 24 | **def add**(): | |
| 25 | **while** True: | |
| 26 | f,e\_id=validate\_id() | |
| 27 | **if** f: | |
| 28 | print("<This Employee\_ID already exists>".center(80,"-")) | |
| 29 | **while** True: | |
| 30 | choice=input("Continue with a different Id?[Y/N]:-").lower() | |
| 31 | **if** choice=="y": | |
| 32 | xd=0 | |
| 33 | **break** | |
| 34 | **elif** choice=="n": | |
| 35 | xd=1 | |
| 36 | **break** | |
| 37 | **else**: | |
| 38 | print("<Invalid Choice>".center(80,"-")) | |
| 39 | **continue** | |
| 40 | **if** xd==1: | |
| 41 | **break** | |
| 42 | **else**: | |
| 43 | **while** True: | |
| 44 | name=input("Name: ").title() | |
| 45 | **if not** name=="": | |
| 46 | **break** | |
| 47 | | **else**: | |
| 48 | | print("<Name cannot be left empty>".center(80,"-")) | |
| 49 | | **while** True: | |
| 50 | | **try**: | |
| 51 | | basic\_salary=int(input("Basic\_Salary: ")) | |
| 52 | | **break** | |
| 53 | | **except**: | |
| 54 | | print("<Basic\_Salary must be numeric and cannot be left  empty>".center(80,"-")) | |
| 55 | | **while** True: | |
| 56 | | **try**: | |
| 57 | | HRA=int(input("House-Rent\_Allowance: ")) | |
| 58 | | **break** | |
| 59 | | **except**: | |
| 60 | | print("<House-Rent\_Allowance must be numeric and cannot be left empty>".center(80,"-")) | |
| 61 | | **while** True: | |
| 62 | | **try**: | |
| 63 | | CA=int(input("Conveyance\_Allowance: ")) | |
| 64 | | **break** | |
| 65 | | **except**: | |
| 66 | | print("<Conveyance\_Allowance must be numeric and cannot be left empty>".center(80,"-")) | |
| 67 | | cur.execute(f"insert into details values({e\_id},'{name}',{basic\_salary},{HRA},  {CA});") | |
| 68 | | con.commit() | |
| 69 | | print('\n',"<Added>".center(80,"-"),sep="") | |
| 70 | | **while** True: | |
| 71 | | choice=input("More Employees? [Y/N]:- ").lower() | |
| 72 | | **if** choice=="y": | |
| 73 | | xd=0 | |
| 74 | | **break** | |
| 75 | | **elif** choice=="n": | |
| 76 | | xd=1 | |
| 77 | | **break** | |
| 78 | | **else**: | |
| 79 | | print("<Invalid Choice>".center(80,"-")) | |
| 80 | | **continue** | |
| 81 | | **if** xd==1: | |
| 82 | | **break** | |
| 83 | | **def search\_id**(): | |
| 84 | | f,e\_id=validate\_id() | |
| 85 | | **if** f: | |
| 86 | | cur.execute(f"select \* from details where id={e\_id}") | |
| 87 | | t=cur.fetchone() | |
| 88 | | print(f"{'ID':<6}{'Name':<20}{'Basic Salary':<20}{'HRA':<10}{'CA':<10}","-  ".center(65,"-"),sep="\n") | |
| 89 | | print(f"{t[0]:<6}{t[1]:<20}{t[2]:<20}{t[3]:<10}{t[4]:<10}") | |
| 90 | | **else**: | |
| 91 | | print("<This Employee ID does not exists>".center(80,"-")) | |
| 92 | | **def search\_name**(): | |
| 93 | | name=input("Name: ").title() | | |
| 94 | | cur.execute(f"select \* from details where name like '{name}';") | | |
| 95 | | t=cur.fetchall() | | |
| 96 | | **if not** t==[]: | | |
| 97 | | print(f"Potential Matches:-") | | |
| 98 | | print(f"{'ID':<6}{'Name':<20}{'Basic Salary':<20}{'HRA':<10}{'CA':<10}","-  ".center(65,"-"),sep="\n") | | |
| 99 | | t1=0 | | |
| 100 | | **for** i **in** t: | | |
| 101 | | print(f"{t[t1][0]:<6}{t[t1][1]:<20}{t[t1][2]:<20}{t[t1][3]:<10}{t[t1][4]:<10}") | | |
| 102 | | t1+=1 | | |
| 103 | | **else**: | | |
| 104 | | print("<No matches found>".center(80,"-")) | | |
| 105 | | **def total\_salary**(): | | |
| 106 | | f,e\_id=validate\_id() | | |
| 107 | | **if** f: | | |
| 108 | | cur.execute(f"select \* from details where id='{e\_id}';") | | |
| 109 | | t=cur.fetchone() | | |
| 110 | | print("Total Salary=",t[2]+t[3]+t[4]) | | |
| 111 | | **else**: | | |
| 112 | | print("<This Employee ID does not exists>".center(80,"-")) | | |
| 113 | | **def total\_allowance**(): | | |
| 114 | | f,e\_id=validate\_id() | | |
| 115 | | **if** f: | | |
| 116 | | cur.execute(f"select \* from details where id='{e\_id}';") | | |
| 117 | | t=cur.fetchone() | | |
| 118 | | print("Total allowance=",t[3]+t[4]) | | |
| 119 | | **else**: | | |
| 120 | | print("<This Employee ID does not exists>".center(80,"-")) | | |
| 121 | | **while** True: | | |
| 122 | | print('',"Main Menu".center(80,"-"),"1: Add Employee","2: Search with ID","3: Search with Name","4: Total Salary","5: Total Allowance","6: Exit",sep="\n") | | |
| 123 | | **while** True: | | |
| 124 | | **try**: | | |
| 125 | | choice=int(input("Your\_Choice: ")) | | |
| 126 | | **if** choice>6: | | |
| 127 | | print("<Invalid Choice>".center(80,"-")) | | |
| 128 | | **continue** | | |
| 129 | | **break** | | |
| 130 | | **except**: | | |
| 131 | | print("<Invalid Choice>".center(80,"-")) | | |
| 132 | | **continue** | | |
| 133 | | **if** choice==1: | | |
| 134 | | add() | | |
| 135 | | **elif** choice==2: | | |
| 136 | | search\_id() | | |
| 137 | | **elif** choice==3: | | |
| 138 | | search\_name() | | |
| 139 | | **elif** choice==4: | | |
| 140 | | total\_salary() | | |
| 141 | | **elif** choice==5: |
| 142 | | total\_allowance() |
| 143 | | **elif** choice==6: |
| 144 | | con.close() |
| 145 | | print('\n',"<Program has ended>".center(80,"-"),sep="") |
| 146 | | sleep(1) |
| 147 | | **break** |

**Output:**

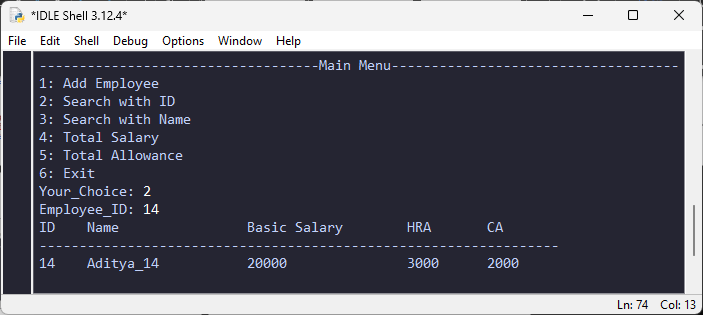
Main Menu:



Add Employee:



Search with ID:

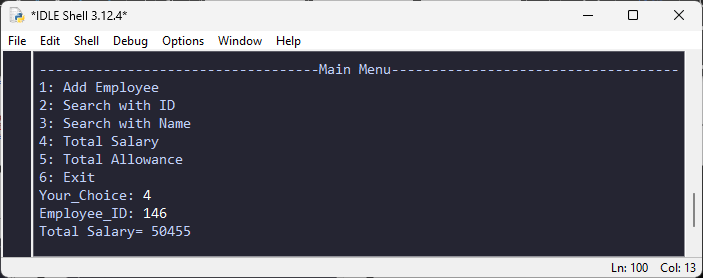


Search with Name:

A screenshot of a computer

AI-generated content may be incorrect.

Total Salary:



Total Allowance:

A screenshot of a computer

AI-generated content may be incorrect.

**Database and Table:**

Database= employee\_db

Table= details

Table Structure:

A screenshot of a computer program

AI-generated content may be incorrect.

Table Contents:

