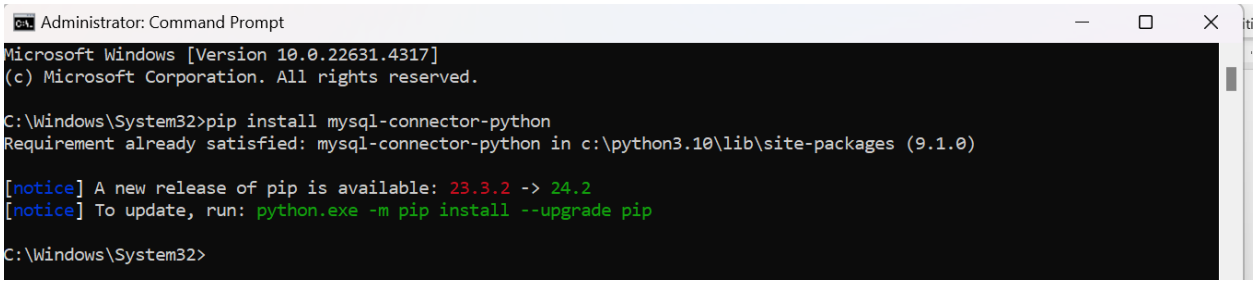
Name: Archana Kalathiya Department: SmartOps Fusion

EMPLOYEE MANAGEMENT SYSTEM

Command Prompt Code and Output:



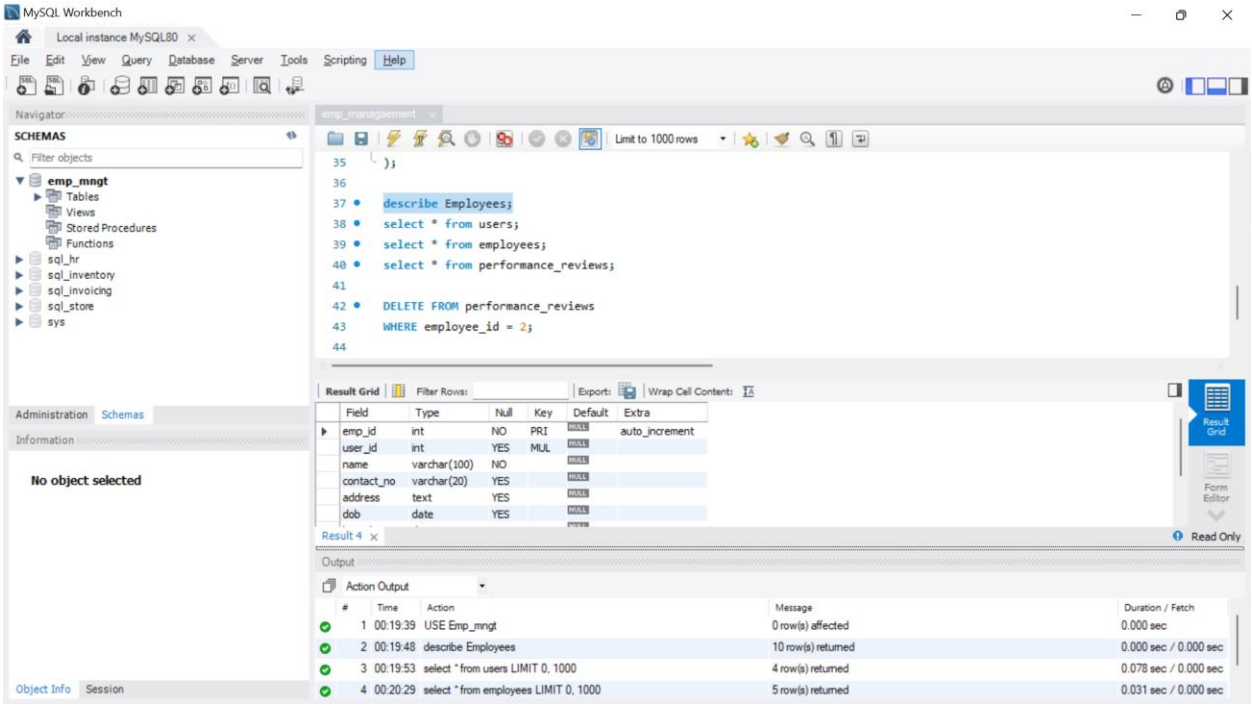
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>pip install mysql-connector-python
Requirement already satisfied: mysql-connector-python in c:\python3.10\lib\site-packages (9.1.0)

[notice] A new release of pip is available: 23.3.2 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Windows\System32>
```

MySQL Workbench Code:

```
CREATE DATABASE Emp_mngt;

USE Emp_mngt;

-- Table for storing user login information
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);

-- Table for storing employee details
CREATE TABLE employees (
    emp_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    name VARCHAR(100) NOT NULL,
    contact_no VARCHAR(20),
    address TEXT,
    dob DATE,
    hire_date DATE,
    position VARCHAR(100) DEFAULT NULL, -- Position can be NULL if not assigned
    salary DECIMAL(10, 2) DEFAULT NULL,  -- Salary can be NULL if not assigned
    experience INT DEFAULT NULL,         -- Experience can be NULL initially
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- Table for storing performance reviews
CREATE TABLE performance_reviews (
    review_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    employee_id INT,

    review_date DATE,

    rating FLOAT CHECK (rating >= 1 AND rating <= 5), -- Rating between 1 and 5

    feedback TEXT,

    FOREIGN KEY (employee_id) REFERENCES employees(emp_id)

);


describe Employees;

describe users;

describe performance_reviews;


select * from users;

select * from employees;

select * from performance_reviews;


DELETE FROM performance_reviews

WHERE employee_id = 2;


DELETE FROM employees

WHERE emp_id = 6;


DELETE FROM users

where id = 3;
```

MYSQL Workbench Output:



MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- emp_mngt
  - Tables
  - Views
  - Stored Procedures
  - Functions
- sql_hr
- sql_inventory
- sql_invoicing
- sql_store
- sys

Administration Schemas

Information

No object selected

Object Info Session

emp_managaement*

Limit to 1000 rows

```
34      FOREIGN KEY (employee_id) REFERENCES employees(emp_id)
35  );
36
37  describe Employees;
38  describe users;
39
40  select * from users;
41  select * from employees;
42  select * from performance_reviews;
43
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
email	varchar(100)	NO	UNI	NULL	
password	varchar(255)	NO		NULL	

Result Grid Form Editor

Result 5 Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:19:39	USE Emp_mngt	0 row(s) affected	0.000 sec
2	00:19:48	describe Employees	10 row(s) returned	0.000 sec / 0.000 sec
3	00:19:53	select * from users LIMIT 0, 1000	4 row(s) returned	0.078 sec / 0.000 sec
4	00:20:29	select * from employees LIMIT 0, 1000	5 row(s) returned	0.031 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- emp_mngt
  - Tables
  - Views
  - Stored Procedures
  - Functions
- sql_hr
- sql_inventory
- sql_invoicing
- sql_store
- sys

Administration Schemas

Information

No object selected

Object Info Session

emp_managaement*

Limit to 1000 rows

```
34      FOREIGN KEY (employee_id) REFERENCES employees(emp_id)
35  );
36
37  describe Employees;
38  describe users;
39  describe performance_reviews;
40  select * from users;
41  select * from employees;
42  select * from performance_reviews;
43
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
review_id	int	NO	PRI	NULL	auto_increment
employee_id	int	YES	MUL	NULL	
review_date	date	YES		NULL	
rating	float	YES		NULL	
feedback	text	YES		NULL	

Result Grid Form Editor

Result 6 Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5	00:20:35	describe Employees	10 row(s) returned	0.000 sec / 0.000 sec
6	00:21:14	describe users	3 row(s) returned	0.000 sec / 0.000 sec
7	00:21:52	describe perfomance_reviews	5 row(s) returned	0.000 sec / 0.000 sec

id	email	password
4	manager@example.com	$2b$12$hDaCdSwE0rdapIZen3d9SOxV2npr/VL...
5	hr@example.com	$2b$12$mE3TpAeF/fo.ZJvmV5kEtOQiDFoC/eTi...
6	teamlead@example.com	$2b$12$TZojeZS/BAXaDa1/uldOpOsyaJnweAt0...
7	manager1@example.com	$2b$12$.vhA.2rIWmaCM1jiVVAgcu2sDVfTW1m...
NULL	NULL	NULL

Users Table

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

emp_id	user_id	name	contact_no	address	dob	hire_date	position	salary	experience
8	5	Frank Black	4561237890	London	1991-11-10	2023-01-20	Reliability Engineer	70000.00	2
10	5	1	Rachel Green	Los Angles	1990-01-02	2018-05-22	Data Analyst	85000.00	7
11	5	Rosy Black	55231298321	Chicago	1990-06-01	2021-05-22	Realiability Engineer	80000.00	4
12	5	Sarah Brown	0987654321	Canada	1985-04-12	2016-01-15	Product Manager	90000.00	8
13	7	Joey	9876543210	London	1990-12-02	2021-02-01	DevOps Engineer	65000.00	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Employee Table

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

review_id	employee_id	review_date	rating	feedback
2	8	2024-10-28	4.5	Creats new builds and maintains code well
3	8	2024-10-28	4.5	Write code well
NULL	NULL	NULL	NULL	NULL

Performance reviews Table

Python Code:

```
import mysql.connector
import bcrypt
from datetime import datetime

# Establishing MySQL connection
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root ",
    database="Emp_mngt"
)
cursor = db.cursor()

# Password hashing functions
def hash_pw(pw):
    return bcrypt.hashpw(pw.encode('utf-8'), bcrypt.gensalt())

def check_pw(hashed_pw, pw):
    return bcrypt.checkpw(pw.encode('utf-8'), hashed_pw)

# Registration function
def register():
    name = input("Enter Name: ")
    email = input("Enter Email ID: ")
    contact = input("Enter Contact No.: ")
    address = input("Enter Address: ")
    dob = input("Enter Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Hire Date (YYYY-MM-DD): ")
    pw = input("Enter Password: ")
    confirm_pw = input("Confirm Password: ")

    if pw != confirm_pw:
        print("Passwords do not match.")
        return

    hashed_pw = hash_pw(pw)

    try:
        # Insert into users table (no employee table insertion here)
        cursor.execute("INSERT INTO users (email, password) VALUES (%s, %s)", (email, hashed_pw))
        db.commit()

        print("Registration successful! You can now log in.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

# Login function
def login():
    email = input("Enter Email ID: ")
```

```
    pw = input("Enter Password: ")

    cursor.execute("SELECT id, password FROM users WHERE email = %s", (email,))
    result = cursor.fetchone()

    if result and check_pw(result[1].encode('utf-8'), pw):
        print("Login successful! ")
        return result[0]  # Returns user ID (Manager's ID)
    else:
        print("Login failed. Check your credentials.")
        return None

# Function to list all employees
def list_employees():
    cursor.execute("SELECT emp_id, name, position FROM employees")
    employees = cursor.fetchall()
    if employees:
        print("¥nEmployees:")
        for emp in employees:
            print(f" Employee ID: {emp[0]}, Name: {emp[1]}, Position: {emp[2]}")
    else:
        print("No employees found.")

# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
        return

    try:
        experience = int(input("Enter Employee Experience in Years: "))
    except ValueError:
        print("Please enter a valid number of years for experience.")
        return

    try:
        cursor.execute(
            "INSERT INTO employees (user_id, name, contact_no, address, dob, hire_date, position, salary, experience)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",
            (manager_id, name, contact, address, dob, hire_date, position, salary, experience)
        )
        db.commit()
```

```
        print("Employee created successfully!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")


# CRUD: Read employee details
def read_employee():
    list_employees()
    emp_id = input("Enter Employee ID to Read: ")
    cursor.execute("SELECT * FROM employees WHERE emp_id = %s", (emp_id,))
    emp = cursor.fetchone()
    if emp:
        print(f"ID: {emp[0]}, Name: {emp[2]}, Contact: {emp[3]}, Address: {emp[4]}, DOB: {emp[5]}, Hire Date: {emp[6]}, Position:
{emp[7]}, Salary: {emp[8]}, Experience: {emp[9]} years")
    else:
        print("Employee not found.")

# CRUD: Update employee details
def update_employee():
    list_employees()
    emp_id = input("Enter Employee ID to Update: ")
    new_contact = input("Enter new Contact No.: ")
    new_address = input("Enter new Address: ")
    cursor.execute("UPDATE employees SET contact_no = %s, address = %s WHERE emp_id = %s",
                   (new_contact, new_address, emp_id))
    db.commit()
    print("Employee details updated successfully.")

# CRUD: Delete employee
def delete_employee():
    list_employees()
    emp_id = input("Enter Employee ID to Delete: ")
    cursor.execute("DELETE FROM employees WHERE emp_id = %s", (emp_id,))
    db.commit()
    print("Employee deleted successfully.")

# CRUD: Add performance review
def add_review():
    list_employees()
    emp_id = input("Enter Employee ID to Add Review: ")
    cursor.execute("SELECT * FROM employees WHERE emp_id = %s", (emp_id,))
    employee = cursor.fetchone()
    if not employee:
        print("No employee found with the given ID.")
        return
    rating = float(input("Enter Rating (1.0-5.0): "))
    feedback = input("Enter Feedback: ")
    review_date = datetime.now().date()

    cursor.execute(
        "INSERT INTO performance_reviews (employee_id, review_date, rating, feedback) VALUES (%s, %s, %s, %s)",
```

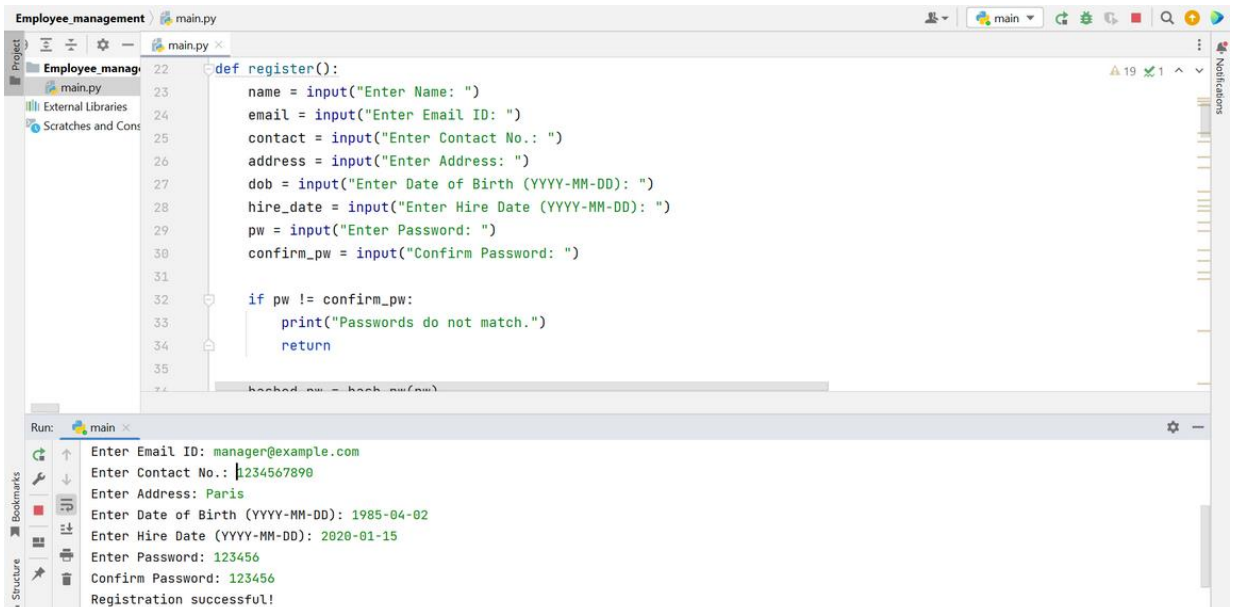
```
        (emp_id, review_date, rating, feedback))
    db.commit()
    print("Performance review added successfully.")

# CRUD: View performance reviews
def view_reviews():
    list_employees()
    emp_id = input("Enter Employee ID to View Reviews: ")
    cursor.execute("SELECT * FROM performance_reviews WHERE employee_id = %s", (emp_id,))
    reviews = cursor.fetchall()
    print("¥nPerformance Reviews:")
    for review in reviews:
        print(f"Date: {review[2]}, Rating: {review[3]}, Feedback: {review[4]}")

# CRUD: Update performance review
def update_review():
    list_employees()
    review_id = input("Enter Review ID to Update: ")
    new_rating = float(input("Enter new Rating (1.0-5.0): "))
    new_feedback = input("Enter new Feedback: ")
    cursor.execute("UPDATE performance_reviews SET rating = %s, feedback = %s WHERE review_id = %s",
                   (new_rating, new_feedback, review_id))
    db.commit()
    print("Review updated successfully.")

# CRUD: Delete performance review
def delete_review():
    list_employees()
    review_id = input("Enter Review ID to Delete: ")
    cursor.execute("DELETE FROM performance_reviews WHERE review_id = %s", (review_id,))
    db.commit()
    print("Review deleted successfully.")

def promote_employee():
    list_employees()  # Show all employees
    emp_id = input("Enter Employee ID to Promote: ")

    cursor.execute("SELECT name, position, salary, experience FROM employees WHERE emp_id = %s", (emp_id,))
    employee = cursor.fetchone()

    if not employee:
        print("No employee found with the given ID.")
        return

    name, current_position, current_salary, experience = employee
    print(f"Current Position: {current_position}, Current Salary: ${current_salary}, Years of Experience: {experience}")

    positions = {
        1: ("DevOps Engineer", 60000),
        2: ("Build Engineer", 65000),
        3: ("Reliability Engineer", 70000),
```

```
        4: ("Release Manager", 80000),
        5: ("Data Analyst", 85000),
        6: ("Product Manager", 90000)
    }

    print("Available Positions for Promotion:")
    for num, (pos, sal) in positions.items():
        print(f"{num}. {pos} - ${sal}")

    # Get user input as a number
    while True:
        try:
            choice = int(input("Select new position by entering the number: "))
            if choice in positions:
                break
            else:
                print("Invalid choice. Please select a valid number.")
        except ValueError:
            print("Please enter a valid number.")

    new_position, new_salary = positions[choice]

    # Update the employee's position and salary
    cursor.execute("UPDATE employees SET position = %s, salary = %s WHERE emp_id = %s",
                   (new_position, new_salary, emp_id))
    db.commit()

    # Show the promotion details
    print(f"\nEmployee {name} promoted!")
    print(f"Previous Position: {current_position}, Previous Salary: ${current_salary}")
    print(f"New Position: {new_position}, New Salary: ${new_salary}")

def main():
    while True:
        print("\n1. Register")
        print("2. Login")
        print("3. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            register()
        elif choice == '2':
            user_id = login()
            if user_id:
                while True:
                    print("\n1. Create Employee")
                    print("2. Read Employee")
                    print("3. Update Employee")
                    print("4. Delete Employee")
                    print("5. Add Performance Review")
                    print("6. View Performance Reviews")
```

```
            print("7. Update Performance Review")
            print("8. Delete Performance Review")
            print("9. Promote Employee")
            print("10. Logout")
            action = input("Enter your action: ")

            if action == '1':
                create_employee(user_id)
            elif action == '2':
                read_employee()
            elif action == '3':
                update_employee()
            elif action == '4':
                delete_employee()
            elif action == '5':
                add_review()
            elif action == '6':
                view_reviews()
            elif action == '7':
                update_review()
            elif action == '8':
                delete_review()
            elif action == '9':
                promote_employee()
            elif action == '10':
                break
            else:
                print("Invalid action. Please select again.")
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please select again.")

if __name__ == "__main__":
    main()
```

PyCharm Output:

Registration



Login

```python
# Login function
def login():
    email = input("Enter Email ID: ")
    pw = input("Enter Password: ")

    cursor.execute("SELECT id, password FROM users WHERE email = %s", (email,))
    result = cursor.fetchone()

    if result and check_pw(result[1].encode('utf-8'), pw):
        print("Login successful! ")
        return result[0]  # Returns user ID (Manager's ID)
    else:
        print("Login failed. Check your credentials.")
        return None

# Function to list all employees
```

```
1. Register
2. Login
3. Exit
Enter your choice: 2
Enter Email ID: hr@example.com
Enter Password: 12345678
Login successful!
```

Add Employee

```python
# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
        return

    try:
```

```
Enter Employee Name: Frank Black
Enter Employee Contact No.: 1234567890
Enter Employee Address: Paris
Enter Employee Date of Birth (YYYY-MM-DD): 1991-11-01
Enter Employee Hire Date (YYYY-MM-DD): 2023-01-20
Enter Employee Position: Build Engineer
Enter Employee Salary: 65000
Enter Employee Experience in Years: 2
Employee created successfully!
```

Read/View

```python
# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
        return

    try:
```

```
Employees:
 Employee ID: 7, Name: Frank Black, Position: Build Engineer
Enter Employee ID to Read: 7
ID: 7, Name: Frank Black, Contact: 1234567890, Address: Paris, DOB: 1991-11-01, Hire Date: 2023-01-20, Position: Build Engineer, Salary:
 65000.00, Experience: 2 years

1. Create Employee
2. Read Employee
```

Update details

```python
# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
        return

    try:
```

```
Employees:
 Employee ID: 7, Name: Frank Black, Position: Build Engineer
Enter Employee ID to Update: 7
Enter new Contact No.: 4561237890
Enter new Address: New York
Employee details updated successfully.

1. Create Employee
```

Delete employee

```python
# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
        return

    try:
```

```
Enter your action: 4

Employees:
 Employee ID: 7, Name: Frank Black, Position: Build Engineer
Enter Employee ID to Delete: 7
Employee deleted successfully.

1. Create Employee
2. Read Employee
```

Add Performance review

```python
def add_review():
    list_employees()
    emp_id = input("Enter Employee ID to Add Review: ")
    cursor.execute("SELECT * FROM employees WHERE emp_id = %s", (emp_id,))
    employee = cursor.fetchone()
    if not employee:
        print("No employee found with the given ID.")
        return
    rating = float(input("Enter Rating (1.0-5.0): "))
    feedback = input("Enter Feedback: ")
    review_date = datetime.now().date()

    cursor.execute(
        "INSERT INTO performance_reviews (employee_id, review_date, rating, feedback) VALUES (%s, %s, %s, %s)",
        (emp_id, review_date, rating, feedback))
    db.commit()
```

```
 Employee ID: 8, Name: Frank Black, Position: Build Engineer
 Employee ID: 9, Name: David Wilson, Position: DevOps Engineer
 Employee ID: 10, Name: 1, Position: Data Analyst
 Employee ID: 11, Name: Rosy Black, Position: Realiability Engineer
 Employee ID: 12, Name: Sarah Brown, Position: Product Manager
Enter Employee ID to Add Review: 8
Enter Rating (1.0-5.0): 4.5
Enter Feedback: Creats new builds and maintains code well
Performance review added successfully.
```

View Performance Review

```python
    db.commit()
    print("Performance review added successfully.")

# CRUD: View performance reviews
def view_reviews():
    list_employees()
    emp_id = input("Enter Employee ID to View Reviews: ")
    cursor.execute("SELECT * FROM performance_reviews WHERE employee_id = %s", (emp_id,))
    reviews = cursor.fetchall()
    print("\nPerformance Reviews:")
    for review in reviews:
        print(f"Date: {review[2]}, Rating: {review[3]}, Feedback: {review[4]}")

# CRUD: Update performance review
def update_review():
    list_employees()
```

```
Employee ID: 10, Name: 1, Position: Data Analyst
Employee ID: 11, Name: Rosy Black, Position: Realiability Engineer
Employee ID: 12, Name: Sarah Brown, Position: Product Manager
Enter Employee ID to View Reviews: 8

Performance Reviews:
Date: 2024-10-28, Rating: 4.5, Feedback: Creats new builds and maintains code well

1. Create Employee
```

Update Performance review

```python
    print("\nPerformance Reviews:")
    for review in reviews:
        print(f"Date: {review[2]}, Rating: {review[3]}, Feedback: {review[4]}")

# CRUD: Update performance review
def update_review():
    list_employees()
    review_id = input("Enter Review ID to Update: ")
    new_rating = float(input("Enter new Rating (1.0-5.0): "))
    new_feedback = input("Enter new Feedback: ")
    cursor.execute("UPDATE performance_reviews SET rating = %s, feedback = %s WHERE review_id = %s",
                   (new_rating, new_feedback, review_id))
    db.commit()
    print("Review updated successfully.")

# CRUD: Delete performance review
```

```
Employee ID: 8, Name: Frank Black, Position: Build Engineer
Employee ID: 9, Name: David Wilson, Position: DevOps Engineer
Employee ID: 10, Name: 1, Position: Data Analyst
Employee ID: 11, Name: Rosy Black, Position: Realiability Engineer
Employee ID: 12, Name: Sarah Brown, Position: Product Manager
Enter Review ID to Update: 8
Enter new Rating (1.0-5.0): 3.9
Enter new Feedback: manages code well but needs more improvement.
Review updated successfully.
```

Delete Performance Review

```python
    db.commit()
    print("Review updated successfully.")

# CRUD: Delete performance review
def delete_review():
    list_employees()
    review_id = input("Enter Review ID to Delete: ")
    cursor.execute("DELETE FROM performance_reviews WHERE review_id = %s", (review_id,))
    db.commit()
    print("Review deleted successfully.")

def promote_employee():
    list_employees()  # Show all employees
    emp_id = input("Enter Employee ID to Promote: ")

    cursor.execute("SELECT name, position, salary, experience FROM employees WHERE emp_id = %s", (emp_id,))
```

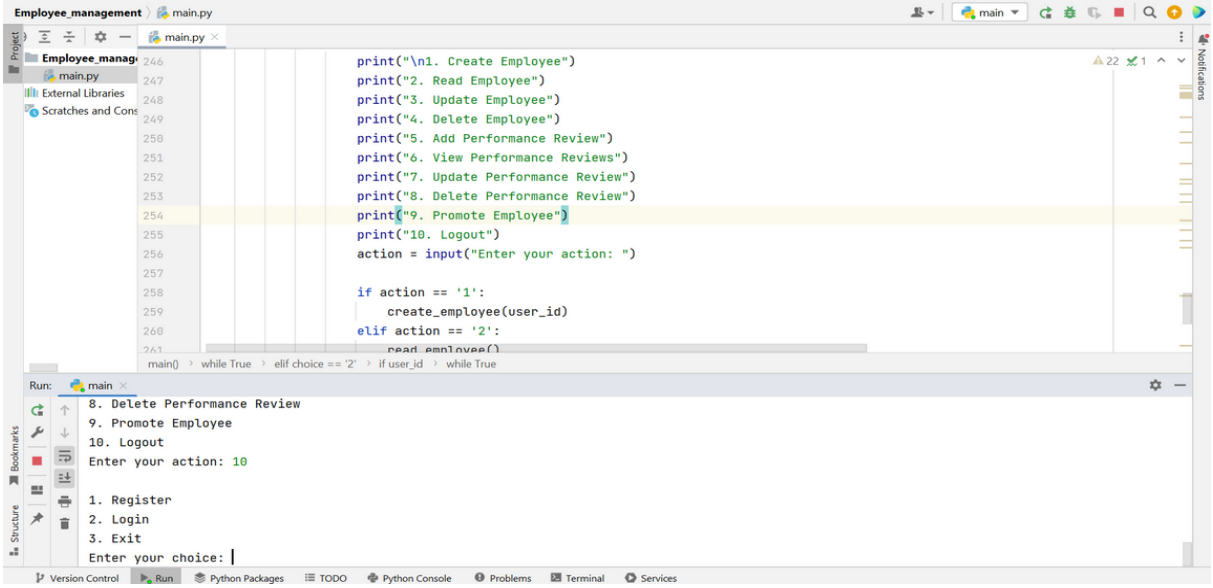
```
Employees:
 Employee ID: 8, Name: Frank Black, Position: Build Engineer
 Employee ID: 9, Name: David Wilson, Position: DevOps Engineer
 Employee ID: 10, Name: 1, Position: Data Analyst
 Employee ID: 11, Name: Rosy Black, Position: Realiability Engineer
 Employee ID: 12, Name: Sarah Brown, Position: Product Manager
Enter Review ID to Delete: 8
Review deleted successfully.
```

Promote Employee

```python
def promote_employee():
    list_employees()  # Show all employees
    emp_id = input("Enter Employee ID to Promote: ")

    cursor.execute("SELECT name, position, salary, experience FROM employees WHERE emp_id = %s", (emp_id,))
    employee = cursor.fetchone()

    if not employee:
        print("No employee found with the given ID.")
        return

    name, current_position, current_salary, experience = employee
    print(f"Current Position: {current_position}, Current Salary: ${current_salary}, Years of Experience: {experience}")

    positions = {
        1: ("DevOps Engineer", 60000),
```

```
3. Reliability Engineer - $70000
4. Release Manager - $80000
5. Data Analyst - $85000
6. Product Manager - $90000
Select new position by entering the number: 2

Employee David Wilson promoted!
Previous Position: DevOps Engineer, Previous Salary: $60000.00
New Position: Build Engineer, New Salary: $65000
```

Logout


Employee_management
main.py
main
Project
Employee_manag
main.py
External Libraries
Scratches and Cons
print("\n1. Create Employee")
print("2. Read Employee")
print("3. Update Employee")
print("4. Delete Employee")
print("5. Add Performance Review")
print("6. View Performance Reviews")
print("7. Update Performance Review")
print("8. Delete Performance Review")
print("9. Promote Employee")
print("10. Logout")
action = input("Enter your action: ")
if action == '1':
create_employee(user_id)
elif action == '2':
read_employee()
main() › while True › elif choice == '2' › if user_id › while True
Run: main
8. Delete Performance Review
9. Promote Employee
10. Logout
Enter your action: 10
1. Register
2. Login
3. Exit
Enter your choice:
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services