**Вариант Е. Предметная область 21**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате. Средняя зарплата должна быть округлена до 2 знака после запятой (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений; для округления необходимо использовать функцию* [*https://docs.python.org/3/library/functions.html#round*](https://docs.python.org/3/library/functions.html#round)).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.

**Текст программы**

class Department:

def \_\_init\_\_(self, id: int, name: str):

self.\_id = id

self.\_name = name

@property

def id(self):

return self.\_id

@property

def name(self):

return self.\_name

class Employee:

def \_\_init\_\_(self, id: int, name: str, salary: float):

self.\_id = id

self.\_name = name

self.\_salary = salary

@property

def id(self):

return self.\_id

@property

def name(self):

return self.\_name

@property

def salary(self):

return self.\_salary

class DepartmentEmployee:

def \_\_init\_\_(self, department\_id: int, employee\_id: int):

self.\_department\_id = department\_id

self.\_employee\_id = employee\_id

@property

def department\_id(self):

return self.\_department\_id

@property

def employee\_id(self):

return self.\_employee\_id

def task1\_modified(Departments, Employees, DepartmentEmployees):

print("Запрос 1 ")

data = [(d, [e for de in DepartmentEmployees for e in Employees if de.department\_id == d.id and de.employee\_id == e.id])

for d in Departments if "отдел" in d.name.lower()]

for department, employees in data:

print(department.name)

for employee in employees:

print(f" {employee.name}")

def task2\_modified(Departments, Employees, DepartmentEmployees):

print("Запрос 2 ")

data = {}

for department in Departments:

employees = [e for de in DepartmentEmployees for e in Employees if de.department\_id == department.id and de.employee\_id == e.id]

if employees:

avg\_salary = round(sum(e.salary for e in employees) / len(employees), 2)

data[department.name] = avg\_salary

for department, avg\_salary in sorted(data.items(), key=lambda x: x[1]):

print(f"{department}: {avg\_salary}")

def task3\_modified(Departments, Employees, DepartmentEmployees):

print("Запрос 3 ")

data = [(e, [d for de in DepartmentEmployees for d in Departments if de.department\_id == d.id and de.employee\_id == e.id])

for e in Employees if e.name.startswith("А")]

for employee, departments in data:

print(f"{employee.name}:")

for department in departments:

print(f" {department.name}")

def main():

# Создаем экземпляры классов для заполнения данными

departments = [

Department(1, "Отдел разработки"),

Department(2, "Отдел маркетинга"),

]

employees = [

Employee(1, "Иванов Иван Иванович", 50000),

Employee(2, "Петрова Мария Петровна", 60000),

]

department\_employees = [

DepartmentEmployee(1, 1), # Иванов в отделе разработки

DepartmentEmployee(2, 2), # Петрова в отделе маркетинга

]

# Вызываем функции для выполнения задач

task1\_modified(departments, employees, department\_employees)

task2\_modified(departments, employees, department\_employees)

task3\_modified(departments, employees, department\_employees)

if \_\_name\_\_ == "\_\_main\_\_":

main()

