Отчет по РК 2

Вариант запросов - В

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

Листинг программы

«main.py»

```
from operator import itemgetter


class Faculty:
    def __init__(self, faculty_id, faculty_name):
        self.faculty_id = faculty_id
        self.faculty_name = faculty_name
        self.departments = []


class Department:
```

```
    def __init__(self, department_id, department_name,
faculty_id):
        self.department_id = department_id
        self.department_name = department_name
        self.faculty_id = faculty_id
        self.professors = []


class Professor:
    def __init__(self, professor_id, professor_name, salary,
department_id):
        self.professor_id = professor_id
        self.professor_name = professor_name
        self.salary = salary
        self.department_id = department_id


def get_one_to_many_association(professors, departments):
    return [(professor.professor_name,
department.department_name)
            for professor in professors
            for department in departments
            if professor.department_id ==
department.department_id]


def get_min_salary_per_department(departments, professors):
    result_2 = []
```

```
    for department in departments:
        department_professors = [professor.salary for
professor in professors if
                                 professor.department_id ==
department.department_id]
        if department_professors:
            min_salary = min(department_professors)
            result_2.append((department.department_name,
min_salary))
    result_2.sort(key=itemgetter(1))
    return result_2
```

```
def get_many_to_many_association(departments, professors):
    return [(department.department_name,
professor.professor_name)
            for department in departments
            for professor in professors
            if professor.department_id ==
department.department_id]
```

```
def filter_professors_by_name(starting_letter,
one_to_many_association):
    return [(professor_name, department_name) for
professor_name, department_name in one_to_many_association if
            professor_name.startswith(starting_letter)]
```

```
def sort_association_by_key(association, key_index):
    return sorted(association, key=itemgetter(key_index))


def main():
    faculties = [
        Faculty(1, "Информатика, искусственный интеллект и системы управления"),
        Faculty(2, "Робототехника и комплексная автоматизация"),
    ]

    departments = [
        Department(101, "Системы обработки информации и управления", 1),
        Department(102, "Программное обеспечение ЭВМ и информационные технологии", 1),
        Department(201, "Инженерная графика", 2),
    ]

    professors = [
        Professor(1001, "Белов Кирилл Иванович", 60000, 101),
        Professor(1002, "Морозов Дмитрий Александрович", 55000, 101),
        Professor(1003, "Андреев Егор Даниилович", 62000, 102),
        Professor(2001, "Кравцов Арсений Артемьевич", 58000, 201),
    ]
```

```
    one_to_many_association =
get_one_to_many_association(professors, departments)
    result_1 = filter_professors_by_name("A",
one_to_many_association)
    result_1 = sort_association_by_key(result_1, 0)
    print("Задание B1:")
    print(result_1)

    result_2 = get_min_salary_per_department(departments,
professors)
    result_2 = sort_association_by_key(result_2, 1)
    print("\nЗадание B2:")
    print(result_2)

    many_to_many_association =
get_many_to_many_association(departments, professors)
    result_3 =
sort_association_by_key(many_to_many_association, 0)
    print("\nЗадание B3:")
    print(result_3)


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

«test.py»

```
import unittest
```

```
from main import Faculty, Department, Professor,
get_one_to_many_association, get_min_salary_per_department, \
    get_many_to_many_association, filter_professors_by_name,
sort_association_by_key


class TestFacultyDepartmentProfessor(unittest.TestCase):

    def setUp(self):
        self.faculties = [
            Faculty(1, "Информатика, искусственный интеллект
и системы управления"),
            Faculty(2, "Робототехника и комплексная
автоматизация"),
        ]

        self.departments = [
            Department(101, "Системы обработки информации и
управления", 1),
            Department(102, "Программное обеспечение ЭВМ и
информационные технологии", 1),
            Department(201, "Инженерная графика", 2),
        ]

        self.professors = [
            Professor(1001, "Белов Кирилл Иванович", 60000,
101),
            Professor(1002, "Морозов Дмитрий Александрович",
55000, 101),
```

```
            Professor(1003, "Андреев Егор Даниилович", 62000,
102),
            Professor(2001, "Кравцов Арсений Артемьевич",
58000, 201),
        ]

    def test_filter_professors_by_name(self):
        one_to_many_association =
get_one_to_many_association(self.professors,
self.departments)
        result = filter_professors_by_name("A",
one_to_many_association)
        self.assertEqual(result,
                         [("Андреев Егор Даниилович",
"Программное обеспечение ЭВМ и информационные технологии")])

    def test_get_min_salary_per_department(self):
        result =
get_min_salary_per_department(self.departments,
self.professors)
        self.assertEqual(result, [('Системы обработки
информации и управления', 55000),
                                  ('Инженерная графика',
58000),
                                  ('Программное обеспечение
ЭВМ и информационные технологии', 62000)])

    def test_get_many_to_many_association(self):
        result =
get_many_to_many_association(self.departments,
self.professors)
```

```
        self.assertEqual(result, [("Системы обработки
информации и управления", "Белов Кирилл Иванович"),
                                  ("Системы обработки
информации и управления", "Морозов Дмитрий Александрович"),
                                  ("Программное обеспечение
ЭВМ и информационные технологии", "Андреев Егор Даниилович"),
                                  ("Инженерная графика",
"Кравцов Арсений Артемьевич")])
```

```
if __name__ == '__main__':
    unittest.main()
```

Результаты работы программы

```
Ran 3 tests in 0.001s

OK
```